(ADAPTIVE) GROVER FIXED-POINT SEARCH FOR QUBO PROBLEMS

ÁKOS NAGY, JAIME PARK, CINDY ZHANG, ATITHI ACHARYA, AND ALEX KHAN

ABSTRACT. *to be completed later...*

1. INTRODUCTION

to be completed later...

Organization of the paper: In Section 2, we introduce the two central notions of the paper; QUBO problems and Quantum Dictionaries. In Section 2.1, we provide an outline of the construction of [3] of the oracle for QUBOs and in Section 2.2 we give our modified design.

Acknowledgment: We are grateful to Amazon Web Services and QLab for providing credits to access IonQ's QPUs. We also thank to Constantin Gonciulea for providing feedback on an earlier version of the work and QuForce.org for bringing the authors together for the project. Section 3 introduces the Grover Fixed-point search for QUBOs

2. (POLYNOMIAL UNCONSTRAINED) BINARY OPTIMIZATION AND QUANTUM DICTIONARIES

For the rest of the paper, $\mathbb{Z}_2^n$ denotes the space of length-n bitstrings. If $x \in [0, 2^n - 1] \cap \mathbb{Z}$ and its binary representation if $x = \overline{x_0 x_1 \dots x_{n-1}} = \sum_{i=0}^{n-1} x_i 2^{n-1-i}$, then $|x\rangle_n := |x_0\rangle |x_1\rangle \dots |x_{n-1}\rangle$, and for an arbitrary $x \in \mathbb{Z}$, $|x\rangle_n = |\bar{x}\rangle_n$, where $x \equiv_{\mod n} \bar{x} \in [0, 2^n - 1] \cap \mathbb{Z}$. We also drop the subscript n, when it is unambiguous. Given a function $f : \mathbb{Z}_2^n \to \mathbb{R}$, the associated (Unconstrained) Binary Optimization problem is the task of finding an element $x \in \mathbb{Z}_2^n$ such that $f(x)$ is maximal. Note that every binary function is polynomial, which can be seen by simple dimension count.

Many interesting Binary Optimization problems, such as finding maximal graph cuts or the Max 2-SAT problems are quadratic, and most of the contemporary research centers around Quadratic Unconstrained Binary Optimization (QUBO) problems.

The first main contribution of the paper is an oracle design for QUBO problems. More concretely, we construct encoding operators of *quantum dictionaries*, as introduced in [2]. Such oracles have applications, for example, in Grover type algorithms and threshold QAOA [4]. While such designs have already existed, cf. [3], ours has better circuit depth, gate count, and CNOT count. Thus, they are simultaneously faster and more noise-resistant.

Briefly, the quantum dictionary, corresponding to a function (thought of as a classical dictionary), $F : \mathrm{dom}(F) \to \mathbb{Z}_2^d$, where $\mathrm{dom}(F) \subseteq \mathbb{Z}_2^n$, is the following quantum state on $n + d$ qubits:

$$|\mathrm{QDICT}(F)\rangle := \frac{1}{\sqrt{|\mathrm{dom}(F)|}} \sum_{x \in \mathrm{dom}(F)} |x\rangle_n |F(x)\rangle_d .$$

Date: October 22, 2023.
2020 *Mathematics Subject Classification.* 11E16, 68Q12, 81P65, 81P68.
Key words and phrases. Grover Fixed-point Search, Quadratic Binary Optimization.

An integer-valued function $f : \mathbb{Z}_2^n \to \mathbb{Z}$ canonically determines a quantum dictionary via first defining $F(x)$ to be the digits of $f(x)$, then setting, by a slight abuse of notation, $|\mathrm{QDICT}(f)\rangle = |\mathrm{QDICT}(F)\rangle$. Let us handle signs via the "Two's complement" convention, in particular, a binary number $y_0 y_1 \dots y_{d-1}$ is negative exactly when $y_0 = 1$. In fact, every quantum dictionary can be realized in this a way.

We construct the above-mentioned encoding oracles in two steps. First, we outline a modified version of the encoding operator given in [3] that is convenient to encode quadratic polynomials. Then we show that quadratic polynomial can be expressed in a basis of functions that can be more efficiently encoded.

2.1. **Quadratic encoder.** Let $I \subseteq \{1, 2, \dots, n-1\}$ and $p_I(x) := x_{i_1} x_{i_2} \cdots x_{i_j}$ be an arbitrary monomial and consider a quantum circuit with $n + d$ qubits. Following [3], we construct an oracle that sends $|x\rangle_n |0\rangle_d$ to $|x\rangle_n |p_I(x)\rangle_d$, for any $x \in \mathbb{Z}_2^n$.

Let us make two definitions: Let QFT_d be the Quantum Fourier Transform on m qubits, that is for any $-2^{d-1} \leqslant y < 2^{d-1}$, we have

$$\mathrm{QFT}_d |y\rangle_d = 2^{-\frac{d}{2}} \sum_{z=-2^{d-1}}^{2^{d-1}-1} e^{\frac{2\pi yz}{2^d} i} |z\rangle_d .$$

Then

$$\mathrm{QFT}_d^\dagger |z\rangle_d = 2^{-\frac{d}{2}} \sum_{y'=-2^{d-1}}^{2^{d-1}-1} e^{-\frac{2\pi y' z}{2^d} i} |y'\rangle_d .$$

Now let $\mathscr{P}_d(k)$ be the following m-qubit gate

$$\begin{array}{lcl}
|z_0\rangle & — \boxed{\mathrm{PHASE}(\pi k)} — & e^{\frac{2\pi i}{2^d} k z_0 2^{d-1}} |z_0\rangle \\
\vdots & & \\
|z_j\rangle & — \boxed{\mathrm{PHASE}\left(\frac{2\pi k}{2^{j+1}}\right)} — & e^{\frac{2\pi i}{2^d} k z_j 2^{d-j-1}} |z_j\rangle \\
\vdots & & \\
|z_{d-1}\rangle & — \boxed{\mathrm{PHASE}\left(\frac{2\pi k}{2^{d}}\right)} — & e^{\frac{2\pi i}{2^d} k z_{d-1}} |z_d\rangle
\end{array}$$

Thus $\mathscr{P}_d(k) |z\rangle_d = e^{\frac{2\pi k z}{2^d} i} |z\rangle_d$.

Now we can prove a well-known lemma. citation needed

Lemma 2.1. *For any $-2^{d-1} \leqslant y < 2^{d-1}$ and $k \in \mathbb{Z}$ we have*

$$\mathrm{QFT}_d^\dagger \circ \mathscr{P}_d(k) \circ \mathrm{QFT}_d |y\rangle_d = |y + k\rangle .$$

Proof. First we compute

$$\begin{aligned}
\mathscr{P}_d(k) \circ \mathrm{QFT}_d |y\rangle_d &= \mathscr{P}(k) \left(2^{-\frac{d}{2}} \sum_{z=-2^{d-1}}^{2^{d-1}-1} e^{\frac{2\pi yz}{2^d} i} |z\rangle_d \right) \\
&= 2^{-\frac{d}{2}} \sum_{z=-2^{d-1}}^{2^{d-1}-1} e^{\frac{2\pi yz}{2^d} i} \mathscr{P}(k) |z\rangle_d
\end{aligned}$$

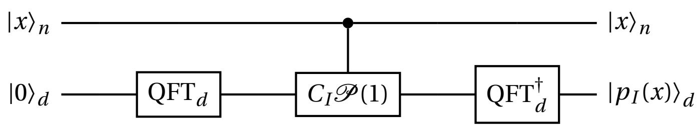
$$= 2^{-\frac{d}{2}} \sum_{z=-2^{d-1}}^{2^{d-1}-1} e^{\frac{2\pi(y+k)z}{2^d} i} |z\rangle_d$$
$$= \mathrm{QFT}_d |y+k\rangle ,$$

hence

$$\mathrm{QFT}_d^\dagger \circ \mathscr{P}(k) \circ \mathrm{QFT}_d |y\rangle_d = |y+k\rangle .$$

□

By Lemma 2.1 it is immediate that



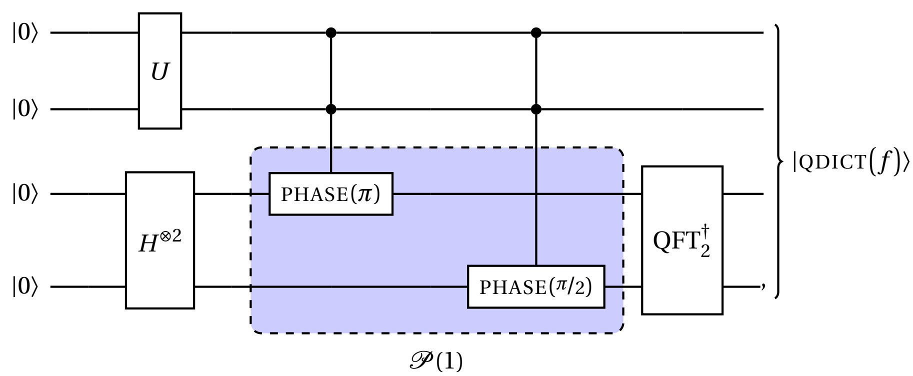
where C_I means control by the qubits $i_1, i_2, \ldots i_j$.

Finally, to create the full quantum dictionary, $|\textsc{qdict}(f)\rangle$, we need to pre-compose an oracle, call U, for which we have

$$U|0\rangle_n = \frac{1}{\sqrt{|\mathrm{dom}(f)|}} \sum_{x \in \mathrm{dom}(f)} |x\rangle_n .$$

If $\mathrm{dom}(f) = \mathbb{Z}_2^n$, then $U = H^{\otimes n}$.

Example 2.2. *Let $n = d = 2$ and $f(x) = x_0 x_1$. Now the encoder oracle takes the form*



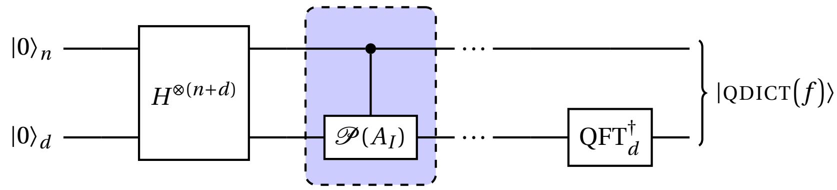
where we used that $\mathrm{QFT}_d |0\rangle_d = H^{\otimes d} |0\rangle_d$.

For the rest of the paper we assume that $\mathrm{dom}(f) = \mathbb{Z}_2^n$ and use $U = H^{\otimes n}$. Furthermore, since $\mathrm{QFT}_d |0\rangle_d = H^{\otimes m} |0\rangle_d$, we can replace QFT_d with $H^{\otimes d}$ in the oracle, as the latter has depth 1 and uses only single-qubit gates.

Let $\mathbb{P}_n$ be the power set of $\{0, 1, \ldots, n-1\}$. Now, for an arbitrary polynomial,

$$f(x) = \sum_{I \in \mathbb{P}_n} A_I x^I ,$$

and $d \in \mathbb{Z}_+$ large enough so that all values of f can be digitized on m bits, we have that



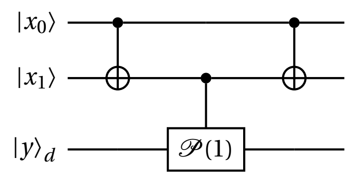
one for each $I = \{i_1, i_2, \dots, i_{|I|}\} \in \mathbb{P}_n$, controlled by qubits $x_{i_1}, x_{i_2}, \dots, x_{i_{|I|}}$

2.2. **New basis for quadratic polynomials and the new oracle design.** We motivate the idea of the new basis by outlining it in the $n = 2$ case.

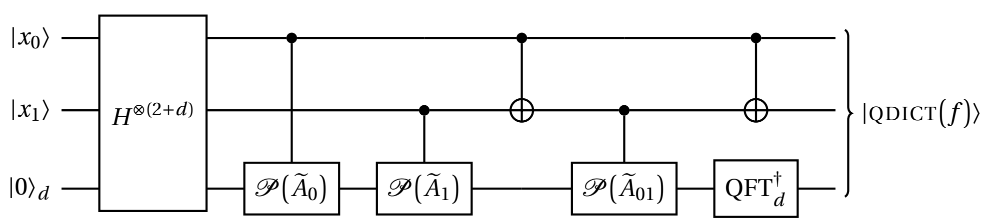
Note first that, since $x_i^2 = x_i$ for binary variables, we have that $x_0 x_1 = \frac{1}{2}\left(x_0 + x_1 - (x_0 - x_1)^2\right)$. Now $(x_0 - x_1)^2$ is also a binary variable, in fact, $(x_0 - x_1)^2 = x_0$ XOR x_1. Now let f be a generic polynomial, $f(x_0, x_1) = A_\emptyset + A_0 x_0 + A_1 x_1 + A_{01} x_0 x_1$. Since we are interested in finding the maximum of f, we can assume, without any loss of generality, that $f(0,0) = A_\emptyset = 0$. By generic, we mean that $0 \notin \{A_0, A_1, A_{01}\}$. Using d digits, we need $2d$ CNOT gates for each linear terms and $6d$ CNOT gates for the quadratic term, thus a total of $10d$ CNOT gates (not counting the CNOT gates in $\mathrm{QFT}_d^\dagger$). However, we can rewrite f as

$$\begin{aligned} f(x_0, x_1) &= A_0 x_0 + A_1 x_1 + A_{01} x_0 x_1 \\ &= \overbrace{\left(A_0 + \tfrac{1}{2} A_{01}\right)}^{\widetilde{A}_0 :=} x_0 + \overbrace{\left(A_1 + \tfrac{1}{2} A_{01}\right)}^{\widetilde{A}_1 :=} x_1 + \overbrace{\left(-\tfrac{1}{2} A_{01}\right)}^{\widetilde{A}_{01} :=} (x_0 - x_1)^2, \end{aligned}$$

and note that the last term can be implemented as



which now has CNOT count only $2 + 2d$. Thus the whole oracle (for arbitrary f) can be realized as



making the new CNOT count for the whole oracle to be (at most) $2 + 6d$, again, not counting the CNOT gates in $\mathrm{QFT}_d^\dagger$. In fact, the only time the two counts equal is when $A_0 = A_1 = 0$, $A_{01} \neq 0$, and $d = 1$.

For a general, n-bit, quadratic polynomial, given by a symmetric, real, n-by-n matrix, Q via

$$f(x_0, x_1, \dots, x_{n-1}) = x^T Q x = \sum_{i=0}^{n-1} Q_{ii} x_i + 2 \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} Q_{ij} x_i x_j,$$

we have that if $q_i = \sum_{j=0}^{n-1} Q_{ij}$ is the sum of the i^{th} row, then

$$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} q_i x_i - \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} Q_{ij} \left(x_i - x_j\right)^2.$$

The addition of a quadratic term, $x_i x_j$, in the oracle design of [3] is implemented via a 2-controlled $\mathscr{P}$-gate, whereas the addition of the term, $\left(x_i - x_j\right)^2$, in our construction is implemented via a 1-controlled $\mathscr{P}$-gate and 2 additional CNOT gates, which is generally more economical both in terms of circuit depth and entangling gate counts. In particular, when decomposed to single-qubit gates and CNOT gates, the former design needs $8d$ for each 2-controlled $\mathscr{P}$-gate, while the latter requires only $2+2d$ CNOT gates. Thus the total CNOT counts for a generic quadratic polynomial are $n(2d) + \frac{n(n-1)}{2}(8d) = \left(4n^2 - 2n\right)d$ and $n(2d) + \frac{n(n-1)}{2}(2+2d) = (d+1)n^2 + (d-1)n$, respectively, which is an approximately fourfold improvement. Since up to d terms can be digitized in parallel, the time complexity of the oracle is $O\left(n^2\right)$, as long as $d = o(n)$.

Remark 2.3. *In the case a single, quadratic monomial and $d = 1$, there is no advantage; in fact, in this case, our construction is just the well-known 2-controlled phase gate from* [5, Figure 4.8]. *Using this observation and a bit more work, one can also show that our construction is never worse (in terms of gate count, CNOT count, or gate circuit depth), than that of* [3].

A further generalization of this construction to higher degree polynomials is also currently being prepared by the authors.

3. Grover Fixed-point search for QUBO

Grover Fixed-Point Search (GFPS) [6] is a variant of Grover's search algorithm that retains the original version query complexity while does not suffer from the soufflé problem (more on these below). In this section we introduce the algorithm, show how our oracle design from Section 2.2 can be used to implement GFPS for QUBO problems, and argue that this method is better suited for an adaptive optimizer algorithm than the original.

As opposed to Grover's algorithm, where the only input is the set of *good* (or *target*) configurations, $T \subset \mathbb{Z}_2^n$, the GFPS algorithm requires an additional one, that can be chosen to be either the target probability/amplitude or the query complexity. The target probability is the probability of finding the system in a good state after running the circuit. The price of this flexibility (and of the elimination of half of the soufflé problem) is that one needs to implement not only a pair of oracles, but two families of them, call $S_s(\alpha)$ and $S_t\left(\beta\right)$ (where the subscripts refer to the *start* and *target* states, and α, β are real parameters), with the following properties: let us fix gauge so that, for some $\lambda \in (0,1)$, we can write

$$|s\rangle = \sqrt{\lambda}\,|t\rangle + \sqrt{1-\lambda}\,|\overline{t}\rangle\,,$$
$$|t\rangle = \sqrt{\lambda}\,|s\rangle + \sqrt{1-\lambda}\,|\overline{s}\rangle\,,$$

where $\langle t|\overline{t}\rangle = \langle s|\overline{s}\rangle = 0$. Then

$$S_s(\alpha)\left(A\,|s\rangle + B\,|\overline{s}\rangle\right) = e^{i\alpha} A\,|s\rangle + B\,|\overline{s}\rangle\,,$$
$$S_t\left(\beta\right)\left(D\,|t\rangle + C\,|\overline{t}\rangle\right) = e^{i\beta} C\,|t\rangle + D\,|\overline{t}\rangle\,.$$

Let $G(\alpha,\beta) := S_s(\alpha)S_t(\beta)$. Once in possession of such oracles and a target probability $P \in (0,1)$, the main result of [6] can be summarized as follows: for any $\mu \in (0,\lambda]$ large enough, there exists $l = l(P,\mu) \in \mathbb{Z}_+$ and $\delta = \delta_{P,\mu} \in (0,1)$, such that if, for all $j \in \{1,\dots,l\}$, we set

$$\alpha_j := 2\operatorname{arccot}\left(\tan\left(\tfrac{2\pi j}{2l+1}\right)\tanh\left(\tfrac{\operatorname{arccosh}(1/\delta)}{2l+1}\right)\right),$$
$$\beta_j := \alpha_{l-j+1},$$

then

$$P_{\text{success}} := \left|\langle t|G(\alpha_l,\beta_l)\cdots G(\alpha_1,\beta_1)|s\rangle\right|^2 \geqslant P.$$

Moreover, l,δ can be explicitly computed (see next section).

We implement $S_s(\alpha)$ and $S_t(\beta)$ in straightforward ways. If U_s is the state preparation oracle, that is, $|s\rangle = U_s|0\rangle$, and $\mathrm{MCP}_n(\alpha)$ is the $(n-1)$-controlled phase gate on n qubits, then

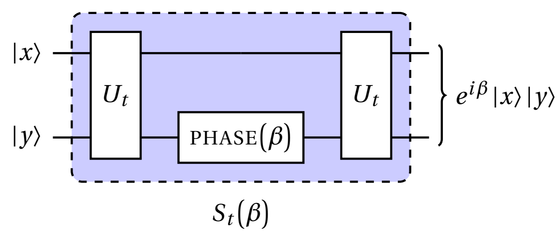
$$S_s(\alpha) = U_s \mathrm{MCP}_n(\alpha) U_s^\dagger, \tag{3.1}$$

works. Note that MCP_n can be implemented with $O(n^2)$ CNOT gates and circuit depth, and no ancillas; cf. [1].

In general, the implementation of $S_t(\beta)$ is case specific. In the case of an instance of a QUBO problem, let $T := \{x \in \mathbb{Z}_2^n | f(x) < 0\}$. Now the results of Sections 2.1 and 2.2 can be immedately used to get an oracle, U_t, on $n+1$ qubits, such that for all $x \in \mathbb{Z}_2^n$ and $y \in \mathbb{Z}_2$, we have

$$U|x\rangle|y\rangle = \begin{cases} |x\rangle|y\oplus 1\rangle, & \text{if } x \in T, \\ |x\rangle|y\rangle, & \text{if } x \notin T, \end{cases}$$

where the last qubit, $|y\rangle$, is the first ancilla of the original oracle. Now



works for $S_t(\beta)$, if the $(n+1)^{\text{st}}$ qubits is a clean ancilla kept in $|0\rangle$ before and after the usage of the oracle. Since QFT can be implemented on d qubits use $O(d^2)$ CNOT gates, the (worst case) CNOT count of U_t is $O(n^2)$, as long as $d = O(n^2)$.

Remark 3.1. *In the case when f is the cut function of a simple, unoriented, and undirected graph, with n vertices and m edges, GFPS can be implemented on $n + O(\log(m))$ qubits, with gate count being $O(m\log(m))$, and circuit depth of $O(m)$.*

3.1. **Time complexity.** In order to understand the time-complexity of GFPS for QUBO problems, we need two know three things:

(1) The query complexity, l.
(2) The complexity of the diffusion operator, $S_s(\alpha)$. Let us call this complexity C_s.
(3) The complexity of the operator, $S_t(\beta)$. Let us call this complexity C_t.

The time complexity is then $l(C_s + C_t)$. From [6], we know that $l \approx \frac{\ln(2/\delta)}{\sqrt{\mu}}$ is sharp. By equation (3.1), we see that C_s is the same as the time complexity of the $\mathrm{MCP}_n(\alpha)$ gate. By [1], we get that $C_s \leqslant O\left(n^2\right)$ (and this is applies to both single qubit and CNOT gate counts). Finally, by the argument of the previous section, $C_t = O\left(n^2\right)$ as well. This yields, for a fixed δ, that the total time complexity if $O\left(\mu^{-\frac{1}{2}} n^2\right)$. Note that we still have the choice of the number $\mu \in (0, \lambda]$ that we discuss in the next section.

Remark 3.2. *In the case of maximal graph cuts, we have already seen in Remark 3.1 we have already seen that the complexity becomes $C_t = O(m) \leqslant O\left(n^2\right)$, and thus the complexity of GFPS is $O\left(\mu^{-\frac{1}{2}}\left(n^2 + m\ln(m)\right)\right)$.*

4. Adaptive Search

As mentioned in the previous section, the choice of μ is problematic; it needs to be at most $\lambda \in (0,1)$, the ratio of marked configurations to all configurations (or, more precisely, the tunneling amplitude between the initial state and the target state), but λ is not known. The ideal choice would be $\mu = \lambda$, but λ is not known. It can be, partially, remedied as follows: Using repeated runs with $\lambda_k = \lambda_0 2^{-k}$ with $k = 1, 2, \ldots$, we can get under λ while not increasing the big-O time complexity of the algorithm.

Note that so far we only implemented a search algorithm that finds negative values of a(n integer-valued) QUBO problem. For any given $y \in \mathbb{Z}$, we can repeat the algorithm with the polynomial $y - f(x)$, making the set of marked states to be $T_y := \{x \in \mathbb{Z}_2^n | f(x) > y\}$. Since our goal is not just to find configurations, x, with values, $f(x)$, above a certain threshold, but rather to find configurations with as high values as possible, we regard y as another parameter. We propose the following adaptive (quantum–classical hybrid) method that we call, following [3], *Adaptive Grover Fixed-point Search* (AGFPS): Given an instance of a QUBO problem and a time threshold $t_{\max} > 0$, set $t := 0$, $\lambda_{0,\mathrm{i}} := \frac{1}{2}$, choose $x_0 \in Z_2^n$ randomly, and set $y_0 := f(x_0)$. While $t < t_{\max}$, ($k \in \mathbb{Z}_+$), do

> Given $y_k, \lambda_{k,\mathrm{initial}}$, use the above method to search for x_{k+1} with $f(x_{k+1}) > y_k$, while in each round incrementing t by the query complexity l.
>
> If x_{k+1} is found before termination: Let $\lambda_{k,\mathrm{final}} > 0$ be the parameter of this search, and set $y_{k+1} := f(x_{k+1})$, $\lambda_{k+1,\mathrm{initial}} := \lambda_{k,\mathrm{final}}$.

Output the last configuration.

Remark 4.1. *The parameter δ, that is approximately the square root of the failure probability, is assumed to be small, but not changed throughout the iterations. Allowing δ to vary is another potential direction of improvement.*

5. Experiments on IonQ's Quantum Computers

5.1. **Oracle testing on IonQ's Aria 2:** As our main experiment, we tested the marker oracle, U_t, alone, since that is the only novel part of the GFPS circuit. The test was done using 9 qubits (5-bit QUBO with 4 ancillas) on IonQ's Aria 2 QPU with 5000 shots. The matrix of the quadratic form used for the test is

$$Q = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & -1 \\ -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & -1 & 0 & 2 \end{pmatrix},$$

with threshold $y = 5$. It is easy to see that out of the $2^5 = 32$ configurations only 3 are above or equal to the threshold, thus $\lambda = \frac{3}{32}$. We index configurations by the numerical value they represent in base 2.

index	$f(x)-y$	not marked	marked	index	$f(x)-y$	not marked	marked
0	-5	3.44%	0.74%	16	-3	1.68%	0.66%
1	-3	1.68%	0.36%	17	-1	1.06%	0.26%
2	-4	3.18%	0.52%	18	-2	1.04%	0.30%
3	-4	4.08%	0.86%	19	-2	2.02%	0.48%
4	-3	2.16%	0.68%	20	-3	3.90%	0.78%
5	-1	2.02%	0.40%	21	-1	2.30%	0.36%
6	-2	1.34%	0.36%	22	-2	3.42%	0.58%
7	-2	2.90%	0.32%	23	-2	4.54%	0.34%
8	-3	1.34%	0.36%	24	-1	1.14%	0.38%
9	-3	5.26%	0.74%	25	-1	2.82%	0.44%
10	-2	2.52%	0.34%	26	0	0.52%	1.16%
11	-4	2.84%	0.38%	27	-2	1.60%	0.52%
12	-1	1.58%	0.20%	28	-1	1.88%	0.26%
13	-1	2.94%	0.88%	29	-1	5.94%	1.74%
14	0	1.06%	1.52%	30	0	0.94%	2.46%
15	-2	2.20%	0.36%	31	-2	4.18%	0.74%

5.2. **Full GFPS testing on IonQ's Harmony and Aria** 2**:** We tested 6 randomly generated 5-vertex graphs with target probability chosen so that the query complexity is 1 on IonQ's Harmony QPU. Several trials for each graph were taken and the trial with the highest success probability was recorded for each graph. Because the algorithm is probabilistic and further bounded by some nonzero error, there can be no guarantee of a success probability of %100. These results were compared with the random probability of measuring cuts above the threshold (that is λ). The results of these experiments are displayed in Figure 1.

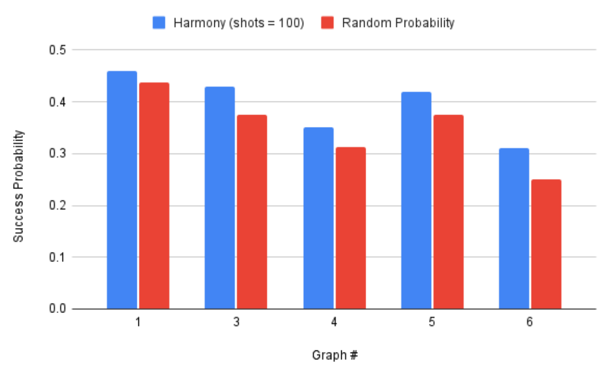


FIGURE 1. 100 shot

At 100 shots, GFPS has a higher probability of finding the maximal cut than by searching randomly. As the shot number increases to 500, this probability decreases and seems to converge with the random probability for graphs 5 and 6. However, neither of these experiments on harmony display significant increased success probability, that is to say, most advantage that the proposed oracle provides is destroyed by the noise present in NISQ devices.

6. CONCLUSION

REFERENCES

[1] Adenilton J. da Silva and Daniel K. Park, *Linear-depth quantum circuits for multiqubit controlled gates*, Phys. Rev. A **106** (2022Oct), 042602. ↑6, 7

[2] Austin Gilliam, Charlene Venci, Sreraman Muralidharan, Vitaliy Dorum, Eric May, Rajesh Narasimhan, and Constantin Gonciulea, *Foundational patterns for efficient quantum computing* (2021). ↑1

[3] Austin Gilliam, Stefan Woerner, and Constantin Gonciulea, *Grover Adaptive Search for Constrained Polynomial Binary Optimization*, Quantum **5** (2021), 428. ↑1, 2, 5, 7

[4] John Golden, Andreas Bärtschi, Daniel O'Malley, and Stephan Eidenbenz, *Threshold-Based Quantum Optimization*, 2021 ieee international conference on quantum computing and engineering (qce), 2021, pp. 137–147. ↑1

[5] Nielsen, Michael A. and Chuang, Isaac L., *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2010. ↑5

[6] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang, *Fixed-point quantum search with an optimal number of queries*, Phys. Rev. Lett. **113** (2014), 210501. ↑5, 6, 7

(Ákos Nagy) BEIT CANADA, TORONTO, ONTARIO
URL: akosnagy.com
Email address: akos@beit.tech

(Jaime Park)

(Cindy Zhang)

(Atithi Acharya)

(Alex Khan)