

MARKER FOR GROVER ADAPTIVE SEARCH BASED CONSTRAINED POLYNOMIAL OPTIMIZATION: ANALYSIS OF COMPLEXITY

Goal

Given an integer-valued function $f : \mathbb{F}_2^n \rightarrow \mathbb{Z}$ in n boolean variables (the objective function), an integer-valued function $C : \mathbb{F}_2^n \rightarrow \mathbb{Z}$ in n -boolean variables (the constraint function) and a threshold $t \in \mathbb{Z}$, write a qiskit function that outputs the marker oracle $U_{f,t,C}$ such that

$$U_{f,t,C}|x\rangle|y\rangle_1 = \begin{cases} |x\rangle|y \oplus 1\rangle & \text{if } f(x) > t \text{ and } C(x) \geq 0, \\ |x\rangle|y\rangle & \text{otherwise.} \end{cases}$$

The implementation may use any number of ancillas, MCX gates and 1-qubit gates.

The marker oracle is implemented using the ideas in [\[GWG21\]](#).

Encoding an integer valued function in several boolean variables

Let $f : \mathbb{F}_2^n \rightarrow \mathbb{Z}$ be the function to be encoded. We first note that any such function has to be a polynomial of degree at most n since $x_i^2 = x_i$ for a boolean variable. Moreover, we have $\binom{n}{k}$ monomials of degree k , $0 \leq k \leq n$. Hence, any such f is a \mathbb{Z} -linear combination of a total of 2^n monomials (including the degree 0 constant monomial).

On the input side we use binary numbers to represent arbitrary monomials. A total of n -qubits are needed for the 2^n possible inputs. A monomial

$$m_{i_0 \dots i_{n-1}} = x_0^{i_0} x_1^{i_1} \dots x_{n-1}^{i_{n-1}},$$

where $i_k \in \{0, 1\}$ is represented by the number corresponding to the binary string $i_{n-1} \dots i_1 i_0$. Then we have

$$f = \sum_{i_0, \dots, i_{n-1} \in \mathbb{F}_2} f(i_{n-1} \dots i_1 i_0) m_{i_0 i_1 \dots i_{n-1}} \equiv \sum_{j=0}^{2^n-1} f(j) m_j. \quad (1)$$

Thus, we can encode f as a list $[f(0), f(1), \dots, f(2^n - 1)]$ of length 2^n , and $f(j)$ is the coefficient of the monomial corresponding to the binary string representing j . Note the the left most qubit here is the most significant.

Implementation of monomials using quantum gates

As in [\[GWG21\]](#), a monomial of degree k can be implemented by a k -controlled $U_G(\theta)$ gate. The $U_G(\theta)$ gate when composed after the Hadamard gate generates a geometric sequence

$$U_G(\theta) H^{\otimes m} |0\rangle_m = \frac{1}{\sqrt{2^m}} \sum_{a=0}^{2^m-1} e^{ia\theta} |a\rangle_m.$$

Using the inverse quantum Fourier transform we have

$$\text{QFT}^\dagger U_G\left(\frac{2\pi f(j)}{2^m}\right) H^{\otimes m} |0\rangle_m = \frac{1}{\sqrt{2^m}} \sum_{a=0}^{2^m-1} e^{ia2\pi f(j)/2^m} |a\rangle_m = |f(j)\rangle_m .$$

This gives representation of an integer on m -qubits. The representation is understood to be in 2's complement. Thus we have $-2^{m-1} \leq f(j) < 2^{m-1}$, with the understanding that

$$2^{m-1} + z = 2^{m-1} + z - 2^m = -(2^{m-1} - z) , \quad 0 \leq z \leq 2^{m-1} - 1 .$$

The controlled version of $U_G(2\pi f(j)/2^m)$ is then used to implement a monomial $f(j) m_j$ in (1) by controlling the m -qubit gate $U_G(2\pi f(j)/2^m)$ from the input register qubits which correspond to a 1. For instance, the constant monomial will be not controlled at all, the monomial x_i will be controlled by 1-qubit, namely qubit i , the monomial $x_i x_j$, ($i \neq j$), will be controlled by qubits i, j , and so on. In general a k -degree monomial will be controlled by k -qubits in the input register, precisely those which would be 1 in the binary string representing the corresponding monomial.

Figure 1 below shows the implementation of $f(x_0, x_1) = 1 + 2x_0 + 3x_1 + 4x_0x_1$ using one $U_G(\pi/8)$ gate, two 1-qubit controlled gates $CU_G(\pi/4)$ and $CU_G(3\pi/8)$, and one 2-qubit controlled gate $C^2U_G(\pi/2)$. The ancillas are needed only for implementing the MCP gates from mcx and 1-qubit phase and Hadamard gates, used in implementing multi-controlled $U_G(\theta)$ gates

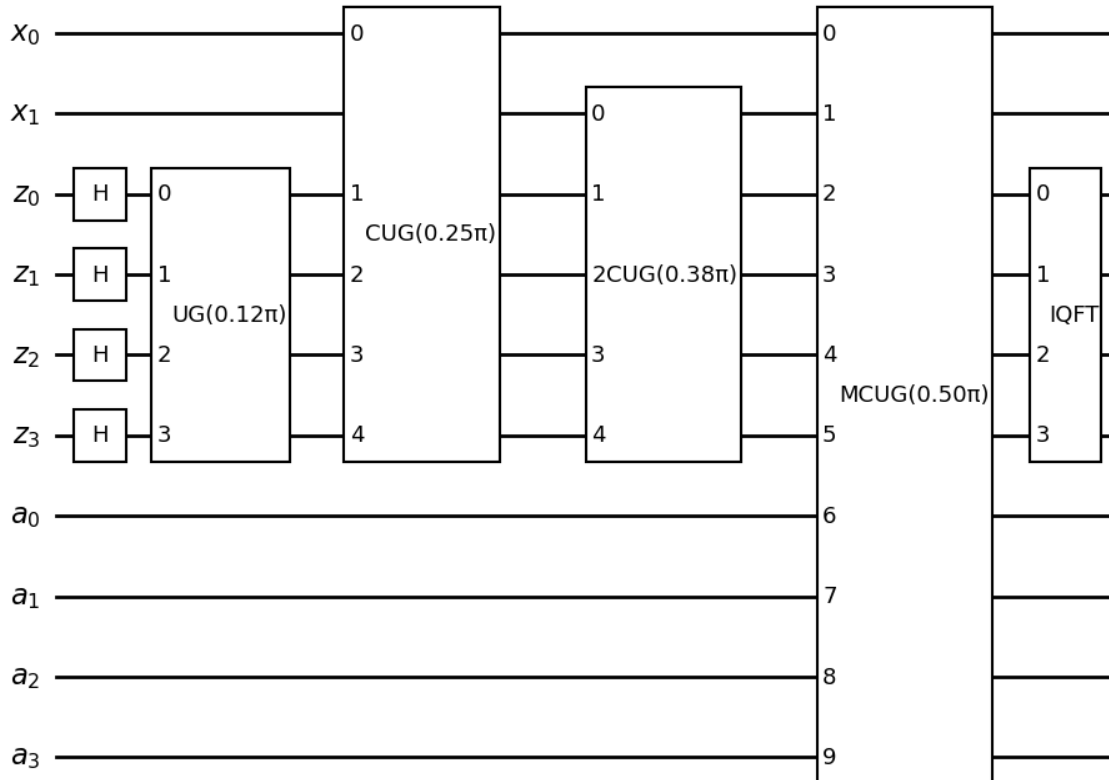


FIGURE 1. Encoding a polynomial

Implementing the oracle

Figure 2 below shows the implementation of marker oracle circuit $U_{f,t,C}$ using the circuit above used to encode a multivariable boolean function. The qubit in y_0 is flipped iff $f(x) > t$. This is done by changing $f(x)$ to $f(x) - t$ and then observing the the most significant qubit of the output, which encodes the sign. Similarly, the qubit in y_1 is flipped iff $C(x) \geq 0$. While $C(x) \geq 0$ can be implemented using controls from only the sign qubit, to implement the strict inequality $f(x) > t$, we also need to use controls from other output qubits. Finally, the qubit y_2 is flipped iff $f(x) > t$ and $C(x) \geq 0$.

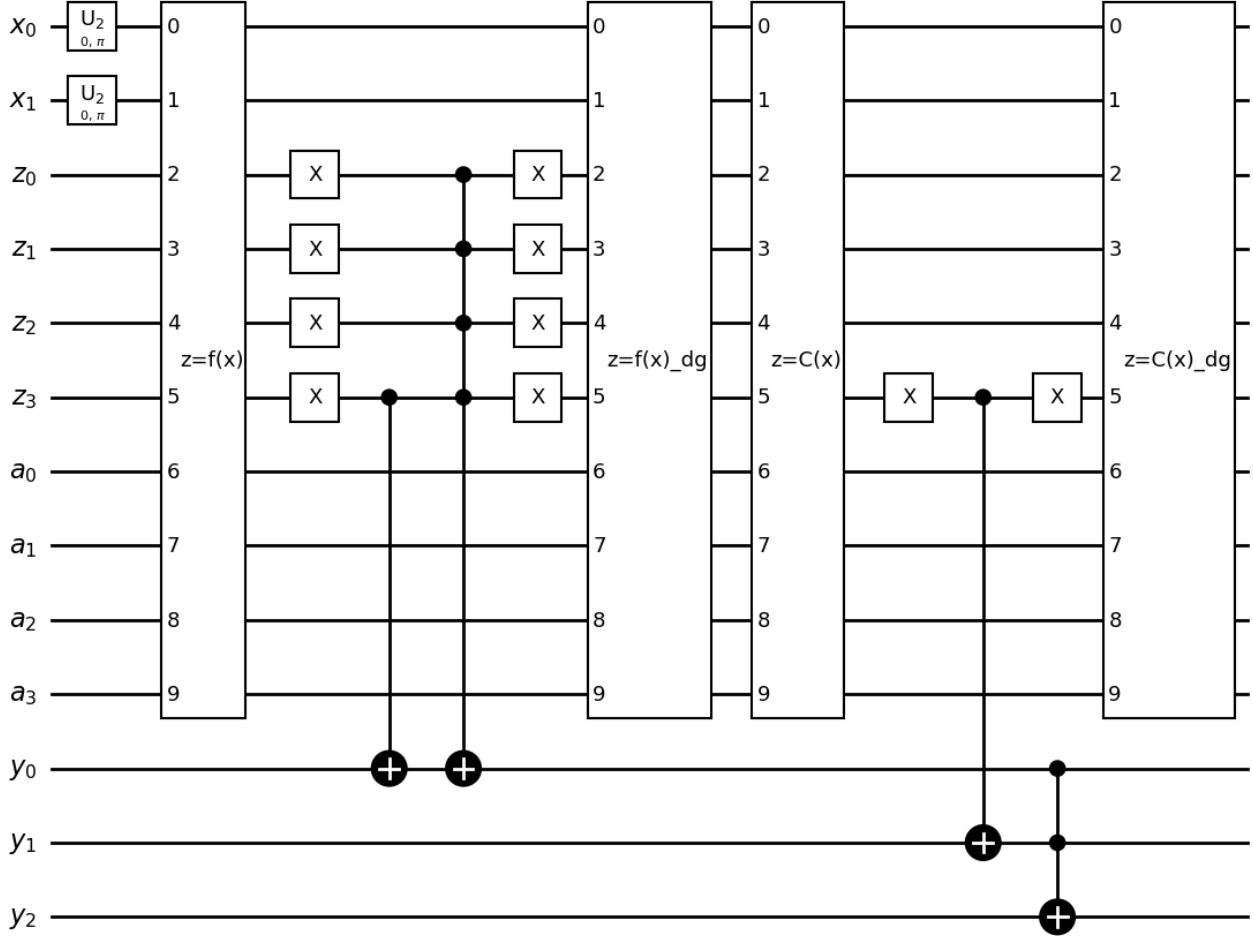


FIGURE 2. Encoding a polynomial

Analysis of space and time complexity

Suppose the output is stored in an m -qubit register. One can implement $U_G(\theta)$ using m 1-qubit phase gates. The 1-qubit controlled version of $U_G(\theta)$ requires m CP-gates, each of which requires 2 CX and 3 phase gates. Both of these do not require any ancillas. For $k \geq 2$, a k -controlled version of $U_G(\theta)$ requires m MCP-gates, each of which requires 3 MCX gates and 3 phase gates, along with 1 ancilla. A function encoding unit requires m Hadamard gates, one $U_G(\theta)$ gate, n $CU_G(\theta)$ gates and $(2^n - n - 1)$

$MCU_G(\theta)$ gates, along with an inverse QFT gate. The inverse QFT on m qubits is implemented using m Hadamard gates, $m(m-1)/2$ CP-gates and $m/2$ swap gates. The swap gates are implemented using 3 CX-gates. The number of gates and ancillas used in each module of the function encoding unit can be described in the following table:

Module	1-qubit Gates	mcx gates	Ancillas
$U_G(\theta)$	m p-gates	0	0
$CU_G(\theta)$	$3m$ p-gates	$2m$	0
$MCU_G(\theta)$	$3m$ p-gates	$3m$	m
IQFT(m)	m h-gates $3m(m-1)/2$ p-gates	$m(2m+1)/2$	0

TABLE 1. The number of gates and ancillas in function encoding unit.

Therefore, in total, one function encoding unit uses $O(m)$ Hadamard gates, $O(m2^n)$ p -gates, $O(m2^n)$ mcx gates, and $O(m2^n)$ ancillas, apart from $(n+m+3)$ quantum registers. There are four such function encoding units in the oracle circuit, apart from $O(n)$ Hadamard gates, $O(m)$ CNOT-gates and 4 mcx gates.

In conclusion, implementing the marker circuit requires $O(m)$ CNOT-gates, $O(n+m)$ Hadamard gates, $O(m2^n)$ phase gates, $O(m2^n)$ mcx gates, $O(m2^n)$ ancillas and $O(n+m)$ quantum registers.

Each of the $U_G(\theta)$ gates, controlled or otherwise, can be executed in one step, since the phase gates are in separate registers, and the control registers do not change. So they have a gate depth of 1. The IQFT(m) has a gate depth of $O(m)$. So, in total one function encoding unit has a gate depth of $O(2^n + m)$. There are four function encoding units in the marker circuit along with some finitely many more steps. Therefore, the gate depth of the marker circuit is also $O(2^n + m)$.

REFERENCES

- [GWG21] A. Gilliam, S. Woerner, and C. Gonciulea, *Grover Adaptive Search for Constrained Polynomial Binary Optimization*, Quantum, **5**, (2021), 428.