algorithms

MDPI

Review

Review Quantum Circuit Synthesis for Grover's Algorithm Oracle

Miguel A. Naranjo * and Luis A. Fletscher *

GITA Lab, Faculty of Engineering, Universidad de Antioquia, Medellín 050010, Colombia
* Correspondence: angel.naranjo@udea.edu.co (M.A.N.); luis.fletscher@udea.edu.co (L.A.F.)

Abstract: The search for information in a system has been a continuous problem for a computer. This has resulted in the construction of a set of classical algorithms that can search for a set of data. This is why search systems can be divided into the type of information being searched, the number of solutions to find, and even the terms used for searching. With the emergence of quantum computing, new algorithms have been generated for this type of process. An example is the Grover algorithm, which performs theoretically better than traditional algorithms. This is why there has been research on optimizing it, applying it to new fields, and making it more accessible to industry users. Even if the algorithm is a promising alternative, one of the disadvantages of Grover's algorithm is the use of an oracle function that must be generated for every set of search data. This review describes three sets of methodologies for generating quantum circuits that can be applied to constructing this oracle quantum circuit.

Keywords: Grover's algorithm; quantum computing; oracle synthesis

1. Introduction

The vast amount of information available on the network requires efficient search methods [1]. While numerous classical search algorithms have been developed, the increasing volume of data demands innovative approaches to data traversal [2]. Quantum computing emerges as a promising alternative, offering potential improvements in computational time and complexity over traditional methods [3]. Current quantum computers leverage quantum state properties, such as superposition and entanglement, to achieve these enhancements [4]. These properties have led to the development of a new set of algorithms that redefine the search problem through a quantum lens [5].

Implementing these algorithms has demonstrated a quadratic improvement in query complexity for specific cases compared to existing classical algorithms [6]. A notable example is Grover's search algorithm, introduced in 1997 [7]. This algorithm utilizes an oracle, a black box that functions as a unitary operator capable of recognizing the solution to the search problem [8]. For Grover's algorithm to function correctly, the oracle must be explicitly mapped from classical information [9]. Additionally, the performance of quantum computers executing this oracle is contingent on the number of qubits available and the number of gates used when making the implementation [9].

The underlying complexity of mapping oracles to functional quantum circuits remains a significant challenge in applying these algorithms to industrial problems [10]. That is why this article reviews the current methodologies available for generating quantum oracles for Grover's search. It begins with the classical synthesis of reversible quantum circuits [11] and then explores trends in circuit generation, such as metaheuristic methods [12] and the generation of quantum circuits using neural networks [13]. This article provides insights into the methodologies for generating quantum circuits, their current limitations, and the efforts to bridge the gap between practical applications of quantum algorithms and traditional software [10].

check for updates

Citation: Naranjo, M.A.; Fletscher, L.A. Review Quantum Circuit Synthesis for Grover's Algorithm Oracle. *Algorithms* **2024**, *17*, 382. https://doi.org/10.3390/a17090382

Academic Editors: Hua-Lei Yin and Nan-Run Zhou

Received: 29 July 2024
Revised: 24 August 2024
Accepted: 25 August 2024
Published: 28 August 2024

2. Background

2.1. Quantum Computing

Quantum computing is a field that relies on the principles of quantum mechanics to solve problems more efficiently than classical computers [4]. Unlike classical computers, where information is represented by the binary states 1 and 0, quantum mechanics allows the extension of classical systems by using the superposition of these two states. This new representation receives the name of qubit, the fundamental unit of information in quantum computing [14].

The qubit representation can be described using the Dirac notation; two vectors represent the binary states $|0\rangle$ and $|1\rangle$, as seen in Equation (1).

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1}$$

The linear combination of these two vectors can produce a new vector ψ, defined as a new quantum state in an arbitrarily unknown state using the quantum superposition principle [15]. The complex coefficients accompanying this expression are α and β, as shown in Equation (2).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

The superposition principle states that the linear combination of two or more state vectors is another state vector in the same Hilbert space and describes another system state [16]. This is why every quantum circuit corresponds to a particular unitary operator U in the Hilbert space that transforms an initial quantum state into another quantum state, meeting the criteria shown in Equation (3) for maintaining the superposition principle [15].

$$U^{\dagger}U = UU^{\dagger} = 1 \tag{3}$$

In addition to the superposition principle, the Born rule states that the modulus square of the amplitude of a state is the probability of that state's resulting after measurement [16]. This implies that the coefficient present in Equation (2) can be operated to obtain the likelihood of the states $|0\rangle$ and $|1\rangle$, and because all the possible states are present in Equation (2), they must meet a normality condition where the sum of the probabilities of the states is equal to 1, as shown in Equation (4).

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4}$$

Then, based on the definition of the quantum circuit as U matrix, we can decompose this matrix into smaller matrices representing part of the circuit, called quantum gates. These gates can interact with a single qubit, as in the case of the X gate that is shown in Equation (5).

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{5}$$

This gate is a unary gate that is responsible for flipping the states of a single qubit, as it is shown in Equation (6).

$$X|0\rangle = |1\rangle; X|1\rangle = |0\rangle \tag{6}$$

There also exist gates that can interact with multiple qubits at the same time. One of them is the CX ($CNOT$) Gate, shown in Equation (7), where an X gate is applied depending on the value of a control qubit.

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{7}$$

A quantum circuit can compromise multiple gates for its construction, as shown in Figure 1. Each horizontal line represents a qubit, and the symbols on the lines represent the basic operations applied to that qubit. The circuit is organized chronologically from left to right.

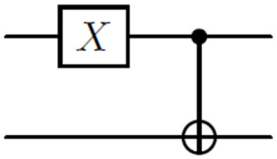

Figure 1. Layout of a fundamental quantum circuit.

2.2. Grover's Algorithm

Based on quantum circuits and amplitude amplification, Grover's search algorithm was proposed in 1997 to search an unstructured dataset faster than any known classical algorithm [15]. The algorithm consists of three steps:

- Superposition of the initial state.
- Oracle that marks the desired state.
- Diffusion operation that amplifies the desired state.

The superposition step represents all the search elements as a linear combination with equal probabilities [17]. After having a superposition step, the oracle marks the desired state of the search problem, which the diffusion operator later amplifies. A measurement procedure is conducted after a series of iterations, making the probability of collapsing the superposition to our desired result higher [16]. In this case, at most, we must iterate $\frac{\pi}{4}\sqrt{N}$ times [18], where N is the number of search terms available, to obtain a better approximation of the desired search state. The general structure of the Grover algorithm is shown in Figure 2.

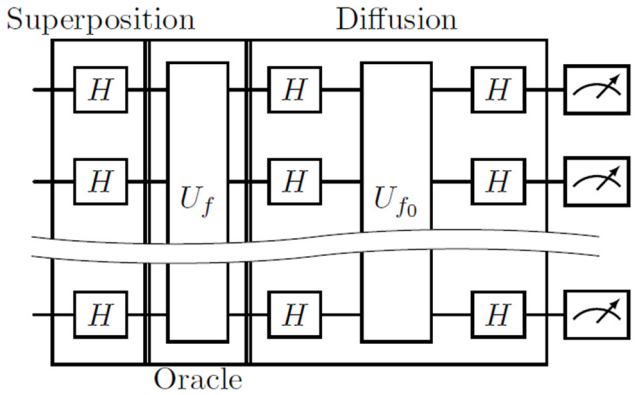

Figure 2. Basic schema of the Grover's algorithm.

The main section of the algorithm is the oracle. It is a subcircuit used in quantum computing to introduce an instance of a mathematical function into a bijective function that is then represented as a quantum circuit [19]. They represent the "black box" function $f(x)$ [10]. This applies to Grover's algorithm and multiple quantum algorithms that can have oracles to represent one of their sections.

In the case of Grover's algorithm, the oracle represents a function $f(x) = 1$ where x is the term searched those outputs 1 when used in the function. A quantum representation of the function is shown in Equation (8). Where operation U_f is the Grover's oracle, $|s\rangle$

represents the current state of the qubits, and ω is the desired state (the searched element). These representations allow inverting the phases only on the desired output state.

$$U_f|s\rangle = \begin{cases} |s\rangle & s \neq \omega \\ -|s\rangle & s = \omega \end{cases} \tag{8}$$

The main problem with this representation is that it requires the construction of an algorithm that does the work of the oracle [Bernhardt], and the user is responsible for generating the oracle circuit. This poses multiple challenges that compromise the quadratic improvement reported by the algorithm [9], such as:

- The oracle is not optimized.
- The oracle depends on the data input, generating it each time.
- The oracle circuit generation is not a direct task.

Due to these problems, multiple approaches have been developed to make the oracle's logical synthesis more approachable, optimized, and automated. Three main groups of methodologies have been identified: classical synthesis, metaheuristics, and neural networks.

3. Methodology

This review methodology is based on the framework proposed by Levac [20]. Following is the set of steps used to obtain and summarize the relevant data.

3.1. Identification of the Research Question

As described in Section 2.2, the problem was identified before starting the research process. The research question was generated about the possible available methodologies for synthesizing oracles, specifically Grover's oracles.

3.2. Identify Relevant Studies

The second step was identifying studies related to the topic. For this, targeted search phrases were employed in academic databases like Arxiv, Science Direct, IEEE Xplore, Google Scholar, and Scopus. The starting targeted search phrases were Grover's oracle synthesis, design automation Grover's search, and quantum circuit oracle synthesis. The keywords used for the search were Grover's algorithm, oracle synthesis, quantum circuit generation, quantum oracle transpilation, and compile quantum circuits.

This search was limited depending on the database used, and because of the topic's novelty, the data range was limited from 2000 to 2024.

3.3. Study Selection

Due to the quantity of available data, a filtering process was completed for the available studies. The criteria used for selecting the studies were the following:

- Relevance: The article's main topic was the synthesis of quantum oracles. Articles on principles of hardware, physics, and layout synthesis were excluded. The selected articles emphasized Grover's algorithm as their primary research topic or as a way of verifying the viability of the stated methodology.
- Credibility: The article was published in a reputable journal.
- Methodology: The article described a proper proof or methodology for validating the oracle synthesis. The articles that contained experimental data were presented clearly and supported by the described process.
- Detail: The article provided concise coverage of the topic, including background on the relevance of these procedures to quantum computing.
- Replicability: The articles exposed code or a detailed step-by-step process that allows future replication and verification of the methodology by other peers.

Based on these criteria, the studies were selected by initial fast filtering of the articles based on the summary proposed and later a detailed review of the topic covered in the

remaining ones. The involvement of different approaches to the solution of generating oracles for quantum computing was considered, leaving us with ten articles.

3.4. Summarizing and Charting the Data

A summarizing process was conducted with the filtered set of studies, which is exposed in Section 4. Based on the search and selection of the studies used for oracle synthesis, they were classified into three main groups. The three groups found were as follows:

- Classical with an abbreviation of [c]. It comprises articles that deal with mapping functions or compilers for translating a Boolean function to a quantum oracle.
- Metaheuristics with an abbreviation of [mh]. It comprises methodologies that use heuristic algorithms to generate and optimize quantum oracles.
- Neural networks with an abbreviation of [nn]. It comprises methodologies that use a computational model inspired by the human brain and the connection of neurons for generating quantum oracles. The neural network can be a classical or quantum implementation representing the oracle.

Finally, depending on the available information for each study. The systematic review was visually represented in Table 1 and in the knowledge graph in Figure 3. Table 1 includes the methods for generating the oracle, the programming language, and the different metrics used to evaluate the performance concerning other methodologies. The knowledge graph visually shows the summary of the found information, and the color represents the methodology used by each study. Green is used for classical [c] methodologies, yellow for metaheuristics [mh], and purple for neural networks [nn].

Table 1. Summary of oracle generation methodologies.

Study	Method	Languages	Libraries	Techniques	Cost Functions
Bogdanov [21]	[c]	-	-	Zhegalkin Polynomial	Number of qubits
Schmitt [22]	[c]	Python; C++	Tweedledum; Caterpillar; Qiskit	ESOP; XAG	$n_{1q} + 10 \cdot n_{2q}$
Meuli [23]	[c]	C++	Caterpillar	ROS	Number of qubits; Number of gates;
Seidel [11]	[c]	Python	Qiskit; Tweedledum	Gray; PTS	Number of qubits; CNOT Gates; U Gates
Sanchez-Rivero [17]	[c]	Python	Qiskit	Arithmetic Oracle	Circuit depth; Number of qubits
Li [24]	[c]	COQ; Haskell	Quipper; DDSIM; SQIR	Custom compiler	-
Velasquez [25]	[c]	-	-	Symbolic maps; QUASH	Number of 2-qubit gates
Atkinson [12]	[mh]	-	-	Ant Colony	$\mathcal{MSF}$
Massey [26]	[mh]	C++	Q-PACE; GALib	Genetic Algorithm	$f = h + c + e$
Ding [27]	[mh]	-	-	Genetic Algorithm	$AS;\ ST$
Abdelmoiz [28]	[mh]	Python	Qiskit	Cellular GA	-
Swaddle [29]	[nn]	Python	TensorFlow	Matrix to circuit	Euclidean distance
Murakami [13]	[nn]	-	-	Input-output Network	Two qubit gates
Cruz-Benito [30]	[nn]	Python	Pytorch; Open QASM	Seq2seq Network	Accuracy

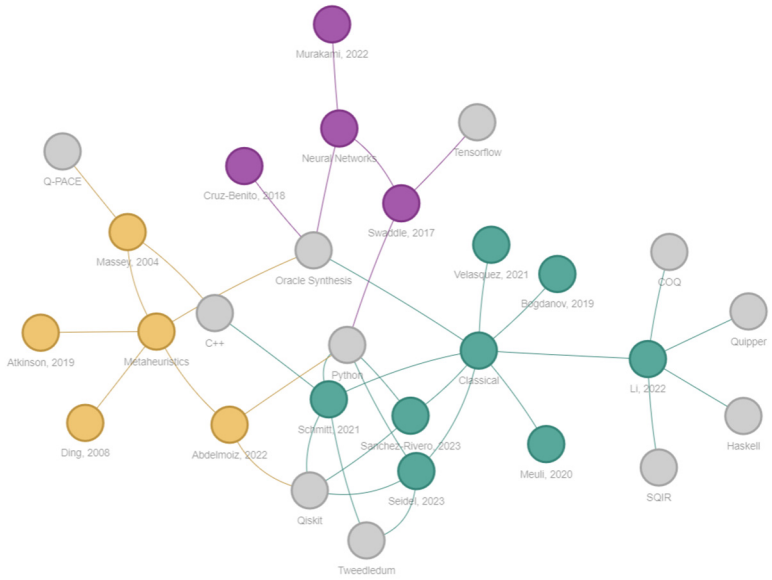

Figure 3. Knowledge graph of articles related to quantum oracle synthesis [11–13,17,21–30]. Yellow nodes represent metaheuristic methodologies, purple nodes represent neural network methodologies, and green nodes represent classical approaches to oracle synthesis. Gray nodes represent languages and libraries used by the different sets of methods.

4. State of the Art

As described in Section 2.2, oracle generation represents a challenge, so multiple methods have emerged that are directly or indirectly applied to Grover's algorithm. The current section classifies these studies into three main sets depending on the methodology used.

4.1. Classical Synthesis Methodologies

Classical synthesis methodologies start with a Boolean function as its main component for representing an oracle. This function is later transformed into a quantum circuit. The types of transformations depend on the methodology. The studies that include this methodology can be divided into two subgroups: one that uses a direct approach for converting the Boolean function in a circuit and one that uses a framework, like a compiler, for setting the transformation and can include more complex operations.

For example, Bogdanov's initial approach [21] starts with a function representing the expected output of the circuit as shown in Table 2, where x represents an input bit and f_i a Boolean function acting on x.

Table 2. Truth table for one bit.

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Bogdanov proposes the construction of the desired output based on a vector space, the bases of which depend on the number of qubits used. The vector bases for one qubit are shown in Equation (9), where $\oplus$ represents a vector sum done in binary base, and $\mathbb{I}$ represents the vector $\begin{bmatrix}1\\1\end{bmatrix}$.

$$E_1 = x = \begin{bmatrix}0\\1\end{bmatrix}, \; e_0 = x \bigoplus \mathbb{I} = \begin{bmatrix}1\\0\end{bmatrix} \tag{9}$$

Based on these vector bases, a new polynomial vector can be written for e_0 in terms of the input x, as is shown in Equation (10). Where x^0 represents a column vector $\begin{bmatrix}1\\1\end{bmatrix}$, x^1 represents $\begin{bmatrix}0\\1\end{bmatrix}$ and the e_0 value is equivalent to $f_2(x)$ in Table 2. This is known as a Zhegalkin polynomial and is the main component used by Bogdanov for obtaining the quantum circuit [21].

$$E_0 = f_2(x) = 1 \cdot x^0 \bigoplus 1 \cdot x^1 \tag{10}$$

If the coefficients of variable x are represented as a column vector $\begin{bmatrix}1\\1\end{bmatrix}$ of name p_0, each of the values of vector p_0 represents a quantum gate that can be mapped on a quantum circuit. This is shown in Figure 4, where the first qubit represents x and the second qubit represents the output of the function $f_2(x)$.

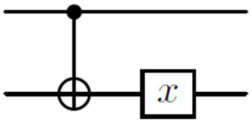

Figure 4. Quantum circuit representation of f_2.

This procedure can be amplified to multiple inputs and outputs, which is Bogdanov's main proposition. The main advantage of this approach is that it considers the reversibility of the quantum circuit. This fundamental property is inherited from the unitary matrices that represent the operations. On the other hand, this approach limits the number of qubits used because it involves adding a qubit for each function output. The input qubits are only used as operation resources.

Similar problems happen with Schmitt's proposition [22]. He uses the problem stated in the 2019 IBM hackathon, called the Zed City problem. The problem consists of a graph representing the distribution of districts in a specific place; the objective is to assign a color to each place, avoiding having adjacent vertices with the same color. Schmitt proposed representing the problem as a Boolean function that describes the search scope of viable solutions to a specific layout. Then, this function would be processed with Grover's algorithm to obtain a viable layout [22].

The process was performed by implementing a Python function that represents the Boolean scope of search. This function was the input to the Tweedledum [31] library; this C++ library has several tasks for synthesizing and optimizing quantum circuits. The specific function used was based on the ESOP-based technique. This technique works similarly to the Zeghalkin polynomials, decomposing the function into an expression that can be later transformed into a particular type of gate, the Toffoli gates. The main advantage of this technique is that it does not require a previous status of reversibility for the Boolean function. Still, it is only performant to small circuits, and it increases the required number of qubits quickly.

To solve this problem, a method of hierarchical synthesis was implemented. It divides the Boolean function into intermediate computations that can be later run with specific synthesis techniques, such as ESOP. For this purpose, the library Caterpillar [32] was used. This library has multiple techniques for implementing hierarchical synthesis. The one implemented was XAG-based, which uses the AND and XOR gates for dividing the circuit.

This type of implementation has the number of qubits defined by $n + 1 + a$, where n represents the number of qubits required by the search function, and a represents the number of temporary ancilla qubits used to store the intermediate results. Finally, the Boolean function that defines the system can be represented by Equation (11). The main difference between this methodology and Bogdanov's is that apart from the input and output qubits required for a quantum circuit, ancilla qubits a appear and serve as an

anchor point to additional calculations and divisions of the quantum circuit that allow more complex procedures.

$$|x\rangle|y\rangle|0\rangle^a \rightarrow |x\rangle|y \oplus f(x)\rangle|0\rangle^a \tag{11}$$

Finally, Schmitt implemented the circuit in Qiskit, a Python library created by IBM to simulate and run the quantum circuits. He took the winning implementation for the challenge and compared it to the obtained results from their procedure. The cost function used for evaluation was $cost = n_{1q} + 10 \cdot n_{2q}$, where n_{1q} represents the number of one-qubit gates and n_{2q} represents the number of two-qubit gates. The circuit optimization was performed manually, and they concluded that their approach had better results than the ones given on the challenge after optimization. The main limitations of this approach were the number of qubits used due to the necessity of a division of circuits that can manage reversible Boolean functions and the manual process of optimization for obtaining comparable results [22].

A similar approach is the one conducted by Meuli [23]. It consists of using the XAG graphs as a representation of the quantum circuit that can then be divided into smaller circuits, with the main difference of implementing an optimizer for the division of the graphs based on the number of qubits required and the number of gates. This methodology allows for a reduction in the number of ancilla qubits used to store the intermediate results in qubits of the operation without affecting the reversibility of the unitary matrices. This algorithm was evaluated using Grover's algorithm search of functionally different graphs, finding a reduction of 30% in the number of gates used and a 2% reduction in the qubits needed.

Another approach that tries to tackle the number of qubits problems is the one conducted by Seidel [11]. He proposes two alternatives for optimizing the oracle synthesis for Grover's algorithm. The first proposition is to search for similarity instead of equality, which means searching for a distance between the required field and the desired output. This type of procedure can be conducted with a quantum function T that represents the distance between the desired output and the evaluated qubits. This type of search also reduces the number of CX gates that the search procedure requires but has the cost of requiring a more precise number of iterations for the oracle and the diffusion operator [11].

The second alternative he proposes is a Phase-Tolerant Synthesis. This type of synthesis considers that the phase inversion of a state is not required for it to be applied to Grover's algorithm since it does not change the result obtained. For example, the state shown in Equation (12) represents the desired state of the synthesis method, where x represents the input, $T(x)$ the distance to the desired output, $\mathcal{X}_x$ garbage phases, and n the number of inputs. This expression has additional phase terms $\mathcal{X}_x$ that can be eliminated from the synthesis process.

$$\sum_{x=0}^{2^n-1} \exp(i\chi_x)|x\rangle|T(x)\rangle \tag{12}$$

This procedure significantly reduces the gates required for constructing the quantum circuit. Seidel assessed this implementation with a database search. He encrypted the data in a quantum circuit. Then, he used the Gray synthesis method from the Tweedledum library and the Phase Tolerant Synthesis to construct an oracle that can search for a register [11]. After the implementation, he iterated thirty times for different database sizes and measured the average $CNOT$ gates, U rotation gates that were happening and the number of qubits used. The experiment concluded that the metrics scaled linearly with the database being searched independent of the method used. The limitations of this approach are that the number of qubits is very dependent on the database being used in the search process, the hash algorithm limits the similarity search, and the involvement of a logic synthesis increases the computational complexity compared to the classical search.

Li [24] took another approach to synthesizing Oracles. This approach is similar to the one proposed by Sanchez-Rivero [17], where instead of generating oracles based on a

Boolean function, an initial set of quantum circuits is defined for performing specific arithmetic operators. In the case of Sanchez-Rivero [17], this is conducted with a less-than operation and a search of numbers that are less than a specific value with Grover's algorithm.

Li uses this concept to define a compilation framework [24]. He proposes a language with a series of steps that transform a function into an oracle that can be used on a quantum circuit. The process starts with the OQIMP, a predefined language that allows the expression of oracles in terms of a series of operations already available. The representation of an Oracle in OQIMP language is then translated to OQASM, a quantum assembly language, throughout a series of predefined circuits and gates that do not have an entanglement due to a language restriction. The quantum circuit is then translated to a Quantum Computer or simulator for testing.

Besides the primary process, Li has designed a test suite to verify the compatibility of the circuits [24]. This verification consists of a formal automated tester set programmed in COQ; additionally, with a randomized test suite PBT, this last one allows testing the process based on random inputs and expected outputs. This methodology has a predefined reduction in the number of qubits and the depth of the circuit.

Li implemented an oracle to evaluate the implementation and find matching registers in a hash function (ChaCha20). This oracle was later used in Grover's algorithm to assess its validity, but its performance was not evaluated. The main limitation of this approach is that it is restricted to the language being used, properties like the entanglement are not an option for the qubits available, and the optimization process is performed after the circuit's construction.

A similar approach was implemented by Velasquez [25], establishing a limited set of symbolic mapping between the input of the gates and the outputs. With this set of symbolic mappings, a symbolic search procedure denoted as QUASH could be implemented to search the set gates that apply to the constraints of the desired input and output. The procedure was evaluated using the number of 2-qubit gates used in constructing the circuit. This approach has the limitation of restricting the type of gates to the set initially defined and the required runtime of the synthesis procedure, which does not enable the implementation of Grover's algorithm with an extensive search scope [25].

Having an overview of the classical methodologies, it can be deduced that they have the following limitations: they require the construction of a Boolean function, which sometimes is not easy depending on the problem; they can introduce additional qubits for managing the computation; and some of them don't take advantage of the entanglement for doing synthesis processes.

4.2. Metaheuristics Methodologies

Metaheuristic methods are another methodology that uses metaheuristic optimization algorithms to generate quantum circuits. These algorithms seek to optimize the output using metrics established for each problem domain [12].

One of these methodologies is the one proposed by Atkinson [12]. The methodology is called Quantum Ant Programming (QAP) and is based on the Ant Colony Optimization metaheuristic for generating quantum circuits. Its principle is based on a graph; the ants chose a path for this graph, generating a subgraph that represents the desired circuit. The ants create a trail of pheromones to reinforce the gates they should use to obtain the desired results, and the evaporation metric is added to avoid bias in the process. Based on the trail the ants went, a circuit can be created to obtain the results based on the inputs.

Atkinson noticed various problems with the standard way of measuring the performance of this model, so he proposed a new metric for measuring the system's performance and used it in the optimization process. The metric is called the Fidelity of Means Squared to calculate the difference between the obtained states and the desired ones, considering

the phase of the result as shown in Equation (13), where $|y\rangle$ represents the expected output of the system and $|x\rangle$ the input state of the system.

$$\mathcal{F}(|y\rangle,\ |x\rangle) = |\langle y|x\rangle| \tag{13}$$

The mean square fidelity can be calculated as is shown in Equation (14), where A represents the realized quantum circuit expressed as a unitary matrix, X represents the set of all input states of the quantum circuit, and Y represents all expected output states of the circuit.

$$\mathcal{MSF}(A, X, Y) = \frac{\sum_{i=1}^{|X|} \mathcal{F}(|y\rangle,\ A|x\rangle)^2}{|X|} \tag{14}$$

With this metric, each algorithm was run one hundred times in a predefined test versus a random generation procedure for generating quantum circuits. This approach obtains better results than random generators but has bad fidelity scores for big circuits. It has the limitation of needing more generalization for more extensive circuits and requiring an expensive run procedure each time a new circuit is needed.

Another approach is the one proposed by Massey [26]. He uses genetic programming to generate the oracles required for Grover. The algorithm uses a tree structure to represent each circuit and tries to insert, delete, or replace the subtree structure at each iteration. Some constraints are imposed on the mutation operators to ensure syntactically valid quantum programs as individuals [26].

The fitness function used for generating these circuits is reported in Equation (15), where h represents the total number of fitness cases used minus the number of fitness cases where the program produces the correct answer.

$$f = h + c + e \tag{15}$$

The c term represents the correctness component and is shown in Equation (16), where the error term represents a threshold applied to not take into consideration more minor errors.

$$C = \frac{\sum_{i=1}^{n} \max(0, error_i - 0.48)}{\max(h, 1)} \tag{16}$$

The last term, e, represents efficiency, which means the number of gates in the final solution divided by a large constant. This allows the circuit generation to center on giving satisfactory results and then optimizing the circuit length based on those results.

Massey finally does not use the output as a whole but uses small functions that make the process more scalable to obtain the average of the fitness functions for each result and maximize the desired output. This approach is assessed with small circuits but needs to be escalated for more extensive circuits.

Another approach to generating quantum oracles as unitary matrices based on evolutionary algorithms was the one proposed by Ding [27]. It consists of encoding the quantum gates as integers and the quantum circuit output as a string of 0–1 bits, which could be expediently operated by evolutionary algorithms [27]. The effectiveness of the procedure was measured by generating some goal circuits with the algorithm and calculating the average generation of success (AS) and the number of successes in twenty tests (ST). The problem with this approach is that it requires an already defined set of gates to be operated with, a previously specified number in the quantity of qubits, and the time used in matrix multiplications for doing the training is time-consuming [27].

In a similar study, Abdelmoiz [28] proposes a variant of the genetic cellular algorithm to generate quantum circuits, where a d-dimensional mesh represents the entire population and each node represents an individual. The goal is to find some neighbor parents to an individual and generate a breeding process between them that can be later evaluated using an objective function given by the problem. Abdelmoiz uses rotation gates as the basis for the implementation.

The circuits are assessed on a fifteen-qubit quantum computer using a Qiskit framework, and they outperform existing circuits in terms of solution quality and convergence time [28].

Despite the results reported, the method of generating quantum circuits from metaheuristic algorithms does not grow linearly with the problems, and its interaction with larger-scale issues needs to be further studied [22].

4.3. Neural Network Methodologies

Classical neural networks have also been explored for generating quantum circuits by using these networks as classical encoding and decoding systems of the result obtained by a quantum circuit, taking care of the preprocessing and post-processing process in a hybrid algorithm [13]. This implies that starting from a quantum algorithm such as Grover's, a set of neural networks with a limited number of layers can be used to define or generate parts of the quantum circuit, such as the oracle, as well as being able to perform error mitigation through an implementation of a classical neural network at the output of the quantum circuit [33].

Swaddle has a proposition for generating quantum oracles based on neural networks. He starts from a unitary matrix that defines the oracle to be applied and uses two sets of neural networks to transform the matrix into trackable circuits [29]. The first step consists of taking a matrix, U, and decomposing it into simpler matrices, as shown in Equation (17). This step is conducted with a GRU network where each input represents a row in the matrix, and the output is the rows of the simplified matrix.

$$U \approx U_1 U_2 \dots U_j \dots U_N \tag{17}$$

The second network is a RELU network with two layers of 2000 neurons. These layers take as input the values of one of the simplified matrices, U_j, and obtain a list of coefficients, c, representing the exponential operations that construct the matrix in consideration, as is shown in Equation (18). These coefficients determine the lie algebra that can be easily translated to quantum gates [29].

$$U_j \approx \exp\left(c_1^j \tau_1\right) \dots \exp\left(c_m^j \tau_m\right) \tag{18}$$

In this case, Swaddle measures the viability of this approach with an implementation in Python and TensorFlow (a neural network library) using the Euclidean distance between the expected and obtained results. He took five thousand pairs of training sets, of which five hundred were used as validation. The reported error of this approach was 0.16.

The limitations of this approach are that the problem of generating the oracle is only taken into consideration from the matrix to quantum translation, and the scalability of this network is not great, which is why it was only evaluated for three qubits.

Another approach to the oracle construction of neural networks was completed by Murakami [13]. He proposed a way of decomposing a unitary operation in a set of quantum gates from a universal set. He starts with a circuit, as is shown in Figure 5, and defines the input of the neural network as the desired inputs and outputs of the circuit because the neural network is going to try to obtain the gates that receive output O_i based on the inputs A and B.

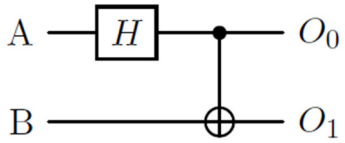

Figure 5. Quantum circuit representation of bell state for a neural network.

Then, they define a set of gates that will operate on the circuit, like the ones shown in Equation (19).

$$CV(0,3), CV(1,3), CX(0,2), CX(1,2) \tag{19}$$

A probability vector was created using the operation set as a starting point, as shown in Equation (20). The idea of the neural network was to assign a probability to the most relevant gate for generating the circuit. That way, we can reverse the operation and obtain the desired input.

$$P = (CV(0,3), CV(1,3), CX(0,2), CX(1,2)) \tag{20}$$

This approach was simulated using ten complex neural network layers and two fully connected layers. The implementation was evaluated on 1.13 million input-output pairs using batch sizes of 64 and epochs of 40. Finally, the number of two-qubit gates used in the quantum circuit measured the resulting quantum circuits.

The results from this approach give better performance in terms of the quantum gates used than other work conducted using a random approach to circuit search. It is important to consider that this approach is limited to the desired input and output pairs, but it gives better performance than the approach to U matrix generation given by Swaddle.

Another option that used natural language processing for obtaining a quantum circuit was the one proposed by Cruz-Benito [30]. The procedure consists of using an encoder-decoder neural network, in this case, the seq2seq neural network, and adapting it to receiving and predicting the construction of quantum circuits. Since the quantum circuits are currently constructed throughout programming languages such as Open QASM, a neural network can be trained in a set of oracles that have already been created to produce a network that can generate and optimize quantum circuits based on a text prompt. This approach has the advantage of optimizing and converting in a single step but depends on the set of already-defined circuits used for the training [30].

The neural network methodology is a good option that adapts to several types of input and output requirements and requires more exploration. However, we must consider the following limitations: it depends on the number of data we have available for the construction of the network; sometimes, the results are not determined; and multiple training processes must be performed to obtain a better adjustment.

5. Conclusions

This review focuses on raising awareness of the relevance of oracle generation in bridging the gap between software production and the use of quantum computing. For this reason, a series of methodologies for generating quantum circuits are presented.

Each set of methodologies has a specific type of limitation that must be considered. Classical methodologies depend on a Boolean function that is not always available. Meta-heuristic algorithms can be expensive to generate. The same happens with the neural network approach, which requires data for training and can introduce an additional error source to the circuit's output.

This contribution emphasizes the necessity of balancing the limitations presented by current methodologies and creating new interfaces that allow quantum computing to be applied more efficiently for generating Grover's algorithm oracles. This will further close the breach between quantum computing and current classical solutions. This research may have future training implications for research focused on using quantum search algorithms.

Author Contributions: Conceptualization, M.A.N. and L.A.F.; methodology, M.A.N. and L.A.F.; investigation, M.A.N.; writing—original draft preparation, M.A.N.; writing—review and editing, M.A.N. and L.A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Thacker, U.; Pandey, M.; Rautaray, S.S. Performance of elasticsearch in cloud environment with nGram and non-nGram indexing. In Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 3–5 March 2016; pp. 3624–3628.
2. Davoudian, A.; Chen, L.; Liu, M. A survey on NoSQL stores. *ACM Comput. Surv.* **2018**, *51*, 40. [CrossRef]
3. Wong, T.G. *Introduction to Classical and Quantum Computing*; Rooted Grove: Omaha, NE, USA, 2022.
4. Sigov, A.; Ratkin, L.; Ivanov, L.A. Quantum Information Technology. *J. Ind. Inf. Integr.* **2022**, *28*, 100365. [CrossRef]
5. Montanaro, A. Quantum algorithms: An overview. *NPJ Quantum Inf.* **2016**, *2*, 15023. [CrossRef]
6. Li, D.; Qian, L.; Zhou, Y.Q.; Yang, Y.G. Quantum partial search algorithm with smaller oracles for multiple target items. *Quantum Inf. Process.* **2022**, *21*, 160. [CrossRef]
7. Grover, L.K. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **1997**, *79*, 325–328. [CrossRef]
8. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: Cambridge, UK, 2012; Volume 27.
9. Viamontes, G.F.; Markov, I.L.; Hayes, J.P. Is quantum search practical? *Quantum* **2005**, *7*, 62–70. [CrossRef]
10. Gheorghe-Pop, I.D.; Tcholtchev, N.; Ritter, T.; Hauswirth, M. Computer Scientist's and Programmer's View on Quantum Algorithms: Mapping Functions APIs and Inputs to Oracles. In *Intelligent Computing, Proceedings of the 2021 Computing Conference, Tokyo, Japan, 17–19 November 2021*; Springer: Cham, The Netherlands, 2021; Volume 1, pp. 188–201.
11. Seidel, R.; Becker, C.K.U.; Bock, S.; Tcholtchev, N.; Gheorghe-Pop, I.D.; Hauswirth, M. Automatic generation of Grover quantum oracles for arbitrary data structures. *Quantum Sci. Technol.* **2023**, *8*, 025003. [CrossRef]
12. Atkinson, T.; Karsa, A.; Drake, J.; Swan, J. *Quantum Program Synthesis: Swarm Algorithms and Benchmarks*; LNCS; Springer International Publishing: Cham, The Netherlands, 2019; Volume 11451.
13. Murakami, K.; Zhao, J. AutoQC: Automated Synthesis of Quantum Circuits Using Neural Network. *arXiv* **2022**, arXiv:2210.02766.
14. Bernhardt, C. *Quantum Computing for Everyone*; Mit Press: Cambridge, UK, 2020.
15. Kasirajan, V. *Fundamentals of Quantum Computing*; Springer International Publishing: Cham, The Netherlands, 2021.
16. Hidary, J.D. *Quantum Computing: An Applied Approach*; Springer International Publishing: Cham, The Netherlands, 2021; pp. 3–21.
17. Sanchez-Rivero, J.; Talav, D.; Garcia-Alonso, J.; Ruiz-Cort, A.; Murillo, J.M. Automatic Generation of an Efficient Less-Than Oracle for Quantum Amplitude Amplification. In Proceedings of the 2023 IEEE/ACM 4th International Workshop on Quantum Software Engineering (Q-SE), Melbourne, Australia, 17 May 2023.
18. Long, G.L. Grover algorithm with zero theoretical failure rate. *Phys. Rev. A—At. Mol. Opt. Phys.* **2001**, *64*, 4. [CrossRef]
19. Henderson, J.M.; Henderson, E.R.; Sinha, A.; Thornton, M.A.; Miller, D.M. Automated Quantum Oracle Synthesis with a Minimal Number of Qubits. In *Quantum Information Science, Sensing, and Computation XV*; SPIE: Bellingham, WA, USA, 2023; pp. 1–18.
20. Levac, D.; Colquhoun, H.; O'Brien, K.K. Scoping studies: Advancing the methodology. *Implement. Sci.* **2010**, *5*, 69. [CrossRef] [PubMed]
21. Fastovets, D.; Bogdanov, Y.; Bogdanova, N.; Lukichev, V. Representation of Boolean functions in terms of quantum computation. In Proceedings of the International Conference on Micro- and Nano-Electronics 2018, Zvenigorod, Russia, 1–5 October 2018; Volume 11022, p. 62.
22. Schmitt, B.; Mozafari, F.; Meuli, G.; Riener, H.; De Micheli, G. From Boolean functions to quantum circuits: A scalable quantum compilation flow in C++. In Proceedings of the Design, Automation and Test in Europe (DATE), Grenoble, France, 1–5 February 2021; pp. 1044–1049.
23. Meuli, G.; Soeken, M.; Roetteler, M.; de Micheli, G. ROS: Resource-constrained oracle synthesis for quantum computers. *Electron. Proc. Theor. Comput. Sci. EPTCS* **2020**, *318*, 119–130. [CrossRef]
24. Li, L.; Voichick, F.; Hietala, K.; Peng, Y.; Wu, X.; Hicks, M. Verified compilation of Quantum oracles. In *Proceedings of the ACM on Programming Languages*; Association for Computing Machinery: New York, NY, USA, 2022; Volume 6, pp. 589–615.
25. Velasquez, A.; Jha, S.K.; Ewetz, R.; Jha, S. Automated synthesis of quantum circuits using symbolic abstractions and decision procedures. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–5. [CrossRef]
26. Massey, P.; Clark, J.A.; Stepney, S. Evolving Quantum Circuits and Programs Through Genetic Programming. In Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, WA, USA, 26–30 June 2004; pp. 569–580.
27. Ding, S.; Jin, Z.; Yang, Q. Evolving quantum circuits at the gate level with a hybrid quantum-inspired evolutionary algorithm. *Soft Comput.* **2008**, *12*, 1059–1072. [CrossRef]
28. Dahi, Z.A.; Alba, E. Metaheuristics on quantum computers: Inspiration, simulation and real execution. *Future Gener. Comput. Syst.* **2022**, *130*, 164–180. [CrossRef]
29. Swaddle, M.; Noakes, L.; Smallbone, H.; Salter, L.; Wang, J. Generating three-qubit quantum circuits with neural networks. *Phys. Lett. Sect. A Gen. At. Solid State Phys.* **2017**, *381*, 3391–3395. [CrossRef]
30. Cruz-Benito, J.; Faro, I.; Martín-Fernández, F.; Therón, R.; García-Peñalvo, F.J. A Deep-Learning-Based Proposal to Aid Users in Quantum Computing Programming. In *Learning and Collaboration Technologies, Proceedings of the Learning and Teaching, LCT 2018, Las Vegas, NV, USA, 15–20 July 2018*; Lecture Notes in Computer Science; Springer International Publishing: Cham, The Netherlands, 2018; Volume 10925, pp. 421–430. [CrossRef]
31. Treinish, M. Tweedledum. 2021. Available online: https://github.com/boschmitt/tweedledum (accessed on 5 July 2023).

32. Meuli, G. Caterpillar. 2020. Available online: https://github.com/gmeuli/caterpillar (accessed on 5 July 2023).
33. Acampora, G.; Schiattarella, R. Deep neural networks for quantum circuit mapping. *Neural Comput. Appl.* **2021**, *33*, 13723–13743. [CrossRef]