Portland State University

PDXScholar

Electrical and Computer Engineering Faculty Publications and Presentations

Electrical and Computer Engineering

10-2024

BHT-QAOA: The Generalization of Quantum Approximate Optimization Algorithm to Solve Arbitrary Boolean Problems as Hamiltonians

Ali Al-Bayaty
Portland State University, albayaty@pdx.edu

Marek Perkowski
Portland State University, h8mp@pdx.edu

Follow this and additional works at: https://pdxscholar.library.pdx.edu/ece_fac

 Part of the Electrical and Computer Engineering Commons

Let us know how access to this document benefits you.

Citation Details

Al-Bayaty, A., & Perkowski, M. (2024). BHT-QAOA: The Generalization of Quantum Approximate Optimization Algorithm to Solve Arbitrary Boolean Problems as Hamiltonians. Entropy, 26(10), 843.

This Article is brought to you for free and open access. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Publications and Presentations by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

entropy

IMPACT FACTOR 2.1

Indexed in: PubMed

CITESCORE 4.9

Article

BHT-QAOA: The Generalization of Quantum Approximate Optimization Algorithm to Solve Arbitrary Boolean Problems as Hamiltonians

Ali Al-Bayaty and Marek Perkowski

Special Issue

The Future of Quantum Machine Learning and Quantum AI

Edited by

Prof. Dr. Andreas (Andrzej) Wichert

MDPI

https://doi.org/10.3390/e26100843

entropy

MDPI

Article

BHT-QAOA: The Generalization of Quantum Approximate Optimization Algorithm to Solve Arbitrary Boolean Problems as Hamiltonians

Ali Al-Bayaty * and Marek Perkowski

Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97201, USA; h8mp@pdx.edu

* Correspondence: albayaty@pdx.edu

Abstract: A new methodology is introduced to solve classical Boolean problems as Hamiltonians, using the quantum approximate optimization algorithm (QAOA). This methodology is termed the "Boolean-Hamiltonians Transform for QAOA" (BHT-QAOA). Because a great deal of research and studies are mainly focused on solving combinatorial optimization problems using QAOA, the BHT-QAOA adds an additional capability to QAOA to find all optimized approximated solutions for Boolean problems, by transforming such problems from Boolean oracles (in different structures) into Phase oracles, and then into the Hamiltonians of QAOA. From such a transformation, we noticed that the total utilized numbers of qubits and quantum gates are dramatically minimized for the generated Hamiltonians of QAOA. In this article, arbitrary Boolean problems are examined by successfully solving them with our BHT-QAOA, using different structures based on various logic synthesis methods, an IBM quantum computer, and a classical optimization minimizer. Accordingly, the BHT-QAOA will provide broad opportunities to solve many classical Boolean-based problems as Hamiltonians, for the practical engineering applications of several algorithms, digital synthesizers, robotics, and machine learning, just to name a few, in the hybrid classical-quantum domain.

Keywords: Boolean oracles; phase oracles; logical structures; logic synthesis; Hamiltonians; QAOA

check for updates

Citation: Al-Bayaty, A.; Perkowski, M. BHT-QAOA: The Generalization of Quantum Approximate Optimization Algorithm to Solve Arbitrary Boolean Problems as Hamiltonians. *Entropy* **2024**, *26*, 843. https://doi.org/10.3390/e26100843

Academic Editor: Andreas (Andrzej) Wichert

Received: 10 September 2024
Revised: 4 October 2024
Accepted: 4 October 2024
Published: 6 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The quantum approximate optimization algorithm (QAOA) was introduced by Farhi et al. [1,2] to mainly solve combinatorial optimization problems, such as the MaxCut [3,4], in the quantum domain. The MaxCut problem is a subset of the classical graph theoretical problems, which is represented by a number of nodes (n) connected through a set of edges {j, k}, where j and k are the indices of two connected nodes. The solutions for a MaxCut problem can be determined by finding the maximum number of cuts (in edges) for the connected nodes. In the quantum domain, the QAOA represents the MaxCut problem in the form of an ansatz Hamiltonian oracle, which is termed the "Hamiltonian clauses (H_C)", and ansatz Hamiltonian operator, which is termed the "Hamiltonian mixer (H_M)".

The wording "ansatz" means that H_C and H_M consist of parameterized rotational quantum gates of Pauli-Z (RZ) and Pauli-X (RX), respectively [1,2], such that H_C consists of a number of RZ($v \cdot \gamma$), RZZ($v \cdot \gamma$), RZZZ($v \cdot \gamma$), and so on, while H_M consists of n numbers of RX($\omega \cdot \beta$), where v and ω are the coefficients (as the time evolutions for both H_C and H_M), γ and β are the parameterized angular rotations, and n is the number of input qubits. Note that γ and β are rotated between the angles of [0, 2π] and [0, π], respectively [1,2].

In general, the following steps illustrate the complete construction of QAOA, to find all approximated solutions for a classical MaxCut problem:

1. All n nodes of the MaxCut problem are represented into their equivalent n input qubits, which are initially set to the $|0\rangle$ state.

2. Hadamard (H) gates are applied to all n input qubits, to create the complete quantum search space of $\{0,1\}^{\otimes n}$ for QAOA to find all solutions.
3. Hamiltonian H_C represents the quantum circuit of a MaxCut problem as the unitary operator $\left(e^{-i\gamma H_C}\right)$, which is a set of non-connected nodes as RZj($v\cdot\gamma$) and connected nodes as RZjZk($v\cdot\gamma$), where j and k are the indices of input qubits.
4. Hamiltonian H_M represents the quantum circuit for the sum of all n input qubits as the unitary operator $\left(e^{-i\beta H_M}\right)$, which is a set of n RX($\omega\cdot\beta$). Note that H_M acts as the quantum diffusion operator of QAOA analogous to the diffusion operator in Grover's algorithm [5–8], and H_M may include other variants and types of gates, not just RX gates, depending on the model of QAOA used (see the Related Work section).
5. To improve the quality of all approximated solutions, H_C and H_M are iterated for a number of repetitions (p), where $p \geq 1$, such that every $e^{-i\gamma_p H_C}$ consists of RZ($v\cdot\gamma_p$), RZZ($v\cdot\gamma_p$), etc., and every $e^{-i\beta_p H_M}$ consists of RX($\omega\cdot\beta_p$).
6. The numerical values of coefficients (v and ω) are calculated during the construction of H_C and H_M in the classical domain.
7. The numerical values of angles (γ and β) are initially randomized as $[\gamma_1 \ldots \gamma_p, \beta_1 \ldots \beta_p]$ for H_C and H_M, respectively, in the classical domain. Note that some studies initialized such angles to defined values using machine learning and tensor techniques [9–13].
8. The quantum circuit of QAOA (H $\{H_C\ H_M\}^p$) is executed with a quantum processing unit (QPU) and then measured (in the classical domain) for approximated solutions depending on the chosen values of γ and β.
9. The measured solutions (as the energy cost of QAOA [1,2]), the chosen values of γ and β (as the optimization parameters of QAOA), and the Hamiltonians (H_C and H_M as an objective function) are fed to a classical optimization minimizer [14–16]. This minimizer re-calculates the numerical values of these optimization parameters based on the energy cost from the objective function and updates the H_C and H_M of QAOA with a new set of optimized numerical values of γ and β, respectively.
10. For a number of objective function evaluations (*nfev*), Steps 8 and 9 are concurrently repeated between a QPU and a minimizer, until finding all optimized approximated solutions for a MaxCut problem or stopping based on a pre-defined "halt condition".

The aforementioned ten steps show why QAOA is considered a variational quantum algorithm solving combinatorial optimization problems in the hybrid classical-quantum domain [17–19]. The wording "variational" is equivalent to the meaning of "ansatz".

The goal of this article is to introduce a new methodology to solve classical Boolean problems as Hamiltonians (H_C and H_M) using QAOA. For that, our methodology is termed the "Boolean-Hamiltonians Transform for QAOA" (BHT-QAOA), which can be summarized as follows. Firstly, an arbitrary classical Boolean problem is constructed as a quantum Boolean oracle [8,20]. This constructed oracle can be expressed in arbitrary structures, such as the Product-Of-Sums (POS) [21,22], Sum-Of-Products (SOP) [21,23], Exclusive-or Sum-Of-Products (ESOP) [24,25], XOR-Satisfiability (CNF-XOR SAT and DNF-XOR SAT) [26,27], and Algebraic Normal Form (ANF) (or Reed–Muller expansion) [28,29], just to name a few. Secondly, this constructed oracle (in any structure) is converted into its equivalent quantum Boolean oracle in ESOP structure, unless it was initially constructed in ESOP structure. Thirdly, the quantum Boolean oracle in ESOP structure is transformed into its equivalent quantum Phase oracle [8,20]. Fourthly, the Hamiltonians (H_C and H_M) of QAOA are generated from this transformed quantum Phase oracle, based on our modified composition rules originally presented by Hadfield [30]. Finally, all the above-mentioned ten steps of QAOA are performed in sequence using the generated H_C and H_M.

Please observe that a quantum Boolean oracle is easier and more straightforward in expressing an arbitrary classical Boolean problem than a quantum Phase oracle, because (i) the quantum Boolean-based gates can be directly realized using the truth tables (and De Morgan's Laws [21]) of their equivalent classical Boolean gates, and (ii) the quantum Boolean-based gates and their qubits can be easily analyzed using classical Boolean logic, such as a Boolean logic of '0' represents a quantum state of $|0\rangle$ and a Boolean logic of '1'

represents a quantum state of $|1\rangle$. For ease of description, the quantum Boolean oracle and the quantum Phase oracle will be simply denoted as the "Boolean oracle" and the "Phase oracle", respectively.

In BHT-QAOA, converting a Boolean oracle (in any structure) into a Phase oracle will (i) remove all ancilla qubits (including the output qubit), i.e., the total number of utilized qubits will be dramatically reduced to the number of input qubits only, and (ii) omit the mirror (as the uncomputing part) of an oracle, i.e., the total number of quantum gates are significantly minimized for the quantum circuit of a Phase oracle, depending on the initial construction of a Boolean oracle that expresses a classical Boolean problem.

In this article, arbitrary classical Boolean problems (as applications) are expressed as Boolean oracles in various structures, and these Boolean oracles are then solved using BHT-QAOA for p repetitions, with an IBM QPU and SciPy optimization minimizer [31]. These applications are (i) an arbitrary Boolean problem in POS structure, (ii) an arbitrary Boolean problem in SOP structure, (iii) an arbitrary Boolean problem in ESOP structure, (iv) a 2×2 Sudoku game, and (v) a 4-bit conditioned half-adder digital circuit. Eventually, our proposed BHT-QAOA successfully finds all optimized approximated solutions for these applications, as a proof of concept for utilizing BHT-QAOA to solve arbitrary and practical classical Boolean problems in the hybrid classical-quantum domain.

2. Related Work

Various research works and different implementations have been proposed to focus on enhancing the optimization workflow of the following essential topics of QAOA:

1. Solving combinatorial optimization problems, such as graphs, k-SAT, and MaxCut problems [3,4,32,33], where $k \geq 3$ literals (inputs).
2. Finding the optimized numerical values of the angles (γ and β) for the Hamiltonians (H_C and H_M), respectively, with fewer function evaluations (*nfev*) and repetitions (p) [34–38], or initially using machine learning techniques [9–13].
3. Developing variants of H_M for a better QAOA in finding all optimized approximated solutions for combinatorial optimization problems [11,39–45].

On the other hand, Hadfield [30] proposed a set of composition rules for constructing the Hamiltonians (H_f), to represent a wide variety of Boolean operators and functions. These Hamiltonians (H_f) are then combined to generate simpler clauses (building blocks) of the final Hamiltonians for the applications of quantum annealing [46,47] and QAOA.

In BHT-QAOA, based on our modified sets of Hadfield's composition rules, the Hamiltonians (H_C and H_M) of QAOA are simply generated from a Phase oracle, which is transformed from a Boolean oracle (in any structure) representing a Boolean problem.

3. Materials and Methods

The essential methodology of BHT-QAOA can be discussed and evaluated using the following illustrative example. Assume an arbitrary classical Boolean problem is given, as stated in Equation (1) below, and then expressed as a Boolean oracle in POS structure, as shown in Figure 1. The following subsections discuss how to (i) convert this Boolean oracle in POS structure into the Boolean oracle in ESOP structure, (ii) transform the Boolean oracle in ESOP structure into the Phase oracle, (iii) generate the Hamiltonians (H_C and H_M) based on the transformed Phase oracle, and (iv) construct the overall architecture of BHT-QAOA for H_C and H_M in p repetitions, where $p \geq 1$. Note that, for other classical Boolean problems, differently expressed Boolean oracles (in any structure) can simply follow the same steps of conversion, transformation, and generation stated in these subsections, and the *fqubit* (functional qubit) is the output ancilla qubit of a Boolean oracle (in any structure).

$$(a \vee b \vee \neg c) \wedge (\neg a \vee c) \wedge (\neg b \vee c) \tag{1}$$

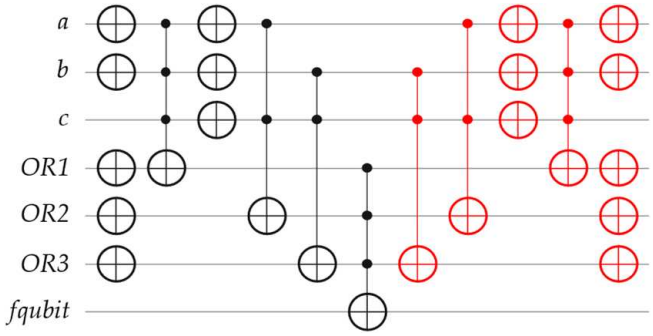

Figure 1. The quantum circuit of a Boolean oracle in POS structure for the classical Boolean problem (a ∨ b ∨ ¬ c) ∧ (¬ a ∨ c) ∧ (¬ b ∨ c), where the *OR1* ancilla qubit represents the term (a ∨ b ∨ ¬ c), the *OR2* ancilla qubit represents the term (¬ a ∨ c), the *OR3* ancilla qubit represents the term (¬ b ∨ c), the *fqubit* ancilla qubit performs all AND operations (∧), and the quantum gates in red denote the mirror (as the uncomputing part) of this Boolean oracle to reset all ancilla qubits to their initial quantum states. Note that all ancilla qubits are initially set to the $|0\rangle$ states.

3.1. Converting Boolean Oracles from Any Structure to ESOP Structure

There are many synthesis methods to convert a Boolean oracle (in any structure) to its equivalent Boolean oracle in ESOP structure. These synthesis methods include ESOP synthesis [24], Karnaugh map synthesis [21], and binary decision diagram (BDD) synthesis [48,49], just to name a few. For instance, the Karnaugh map synthesis for a Boolean oracle (from any structure to ESOP structure) can be summarized in the following steps.

1. Sketch an empty Karnaugh map with literals (a, b, c, ...) and their binary Gray codes.
2. Evaluate a Boolean oracle (in any structure) for solutions (as the true minterms of '1') and non-solutions (as the false minterms of '0').
3. Group all solutions together from step 2, using 1-cell groups, 2-cell groups, etc.
4. Formulate each group from step 3, to generate products (∧) of literals.
5. XOR (⊕) all formulated groups together from step 4, to generate an ESOP structure.

From Equation (1) above, since the Boolean oracle in POS structure is simple, the Karnaugh map synthesis is utilized to convert it to the Boolean oracle in ESOP structure, as stated in Equation (2) below and illustrated in Figure 2a,b, and the final quantum circuit of this Boolean oracle in ESOP structure is shown in Figure 2c. Note that in Figure 2c, (i) there is no need to optimize this Boolean oracle by removing the identical neighboring X gates, since all these gates are required for generating Hamiltonians (H_C and H_M) and calculating their coefficients (v and ω), respectively, and (ii) all mirrored gates and ancillae (except for *fqubit*) are removed when a Boolean oracle is in ESOP structure.

$$(\neg\, a \wedge \neg\, b \wedge \neg\, c) \oplus (a \wedge \neg\, b \wedge c) \oplus (b \wedge c) \tag{2}$$

Please observe that the aforementioned steps of Karnaugh map synthesis may not generate the minimized ESOP structure, as shown in Figure 2b, since these steps usually create the DSOP (Disjoint Sum-Of-Products) structure, as shown in Figure 2c, which is an expensive structure as compared to the minimized ESOP structure depending on the numbers of n-bit Toffoli gates, where $n \geq 3$ qubits. However, we just want to illustrate how to convert a Boolean oracle (in any structure) to its equivalent Boolean oracle in ESOP (or DSOP) structure.

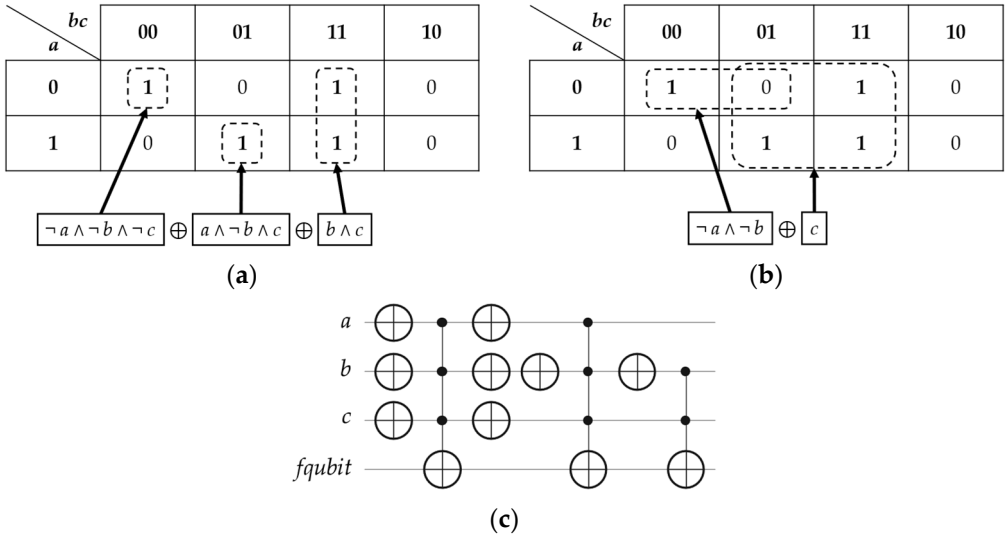

Figure 2. The synthesis method for the classical Boolean problem of Equation (1) above: (**a**) the Karnaugh map synthesis from the Boolean oracle (in Figure 1) to the Boolean oracle in (**c**), based on grouping all true minterms '1' (as solutions) only, (**b**) the Karnaugh map synthesis from the Boolean oracle (in Figure 1) to its equivalent Boolean oracle in ESOP structure $((\neg\, a \wedge \neg\, b) \oplus c)$, based on grouping all '1' minterms (as solutions) with one '0' minterm (as an XORed solution), and (**c**) the Boolean oracle in DSOP structure of Equation (2) above, where the *fqubit* performs all XORing operations ($\oplus$).

3.2. Transforming Boolean Oracles in ESOP Structure to Phase Oracles

In this article, we utilize the technique originally discussed by Figgatt et al. [20] for transforming 4-bit Toffoli gates to 3-bit controlled-Z (CCZ) gates, for Grover's algorithm of single-solution [5–8]. We efficiently generalize their technique to involve the Feynman (CX) and n-bit Toffoli gates, where $n \geq 3$ qubits, for transforming Boolean oracles in ESOP structure to their equivalent Phase oracles, as presented in the following three rules and shown in Figure 3. For that, we termed these rules the "generalized transformation rules".

Rule 1: A Feynman (CX) gate is transformed into a Pauli-Z (Z) gate when Equation (3) stated below is a solution-satisfiable as demonstrated in Figure 3a, where j is the index of an input qubit (q). The left side of Equation (3) is the Boolean-based output of a CX gate, and its right side is the phase-inverted output of a Z gate.

$$\mathrm{q_j} \oplus fqubit \; = -(-1)^{\mathrm{q_j}} \tag{3}$$

Rule 2: A Toffoli gate is transformed into a controlled-Z (CZ) gate when Equation (4) stated below is a solution-satisfiable as shown in Figure 3b, where j and k are the indices of input qubits (q). The left side of Equation (4) is the Boolean-based output of a Toffoli gate, and its right side is the phase-inverted output of a CZ gate.

$$\left(\mathrm{q_j} \wedge \mathrm{q_k}\right) \oplus fqubit = -(-1)^{\mathrm{q_j} \,\cdot\, \mathrm{q_k}} \tag{4}$$

Rule 3: An n-bit Toffoli gate is transformed into an $(n-1)$-bit multi-controlled Z (MCZ) gate when Equation (5) stated below is a solution-satisfiable as shown in Figure 3c, where j is the index of an input qubit (q) and $n \geq 3$ qubits (q + *fqubit*). The left side of Equation (5) is the Boolean-based output of an n-bit Toffoli gate, and its right side is the phase-inverted output of an $(n-1)$-bit MCZ gate.

$$\left(\bigwedge_{\mathrm{j}=1}^{n-1} \mathrm{q_j}\right) \oplus fqubit = -(-1)^{\prod_{\mathrm{j}=1}^{n-1} \mathrm{q_j}} \tag{5}$$

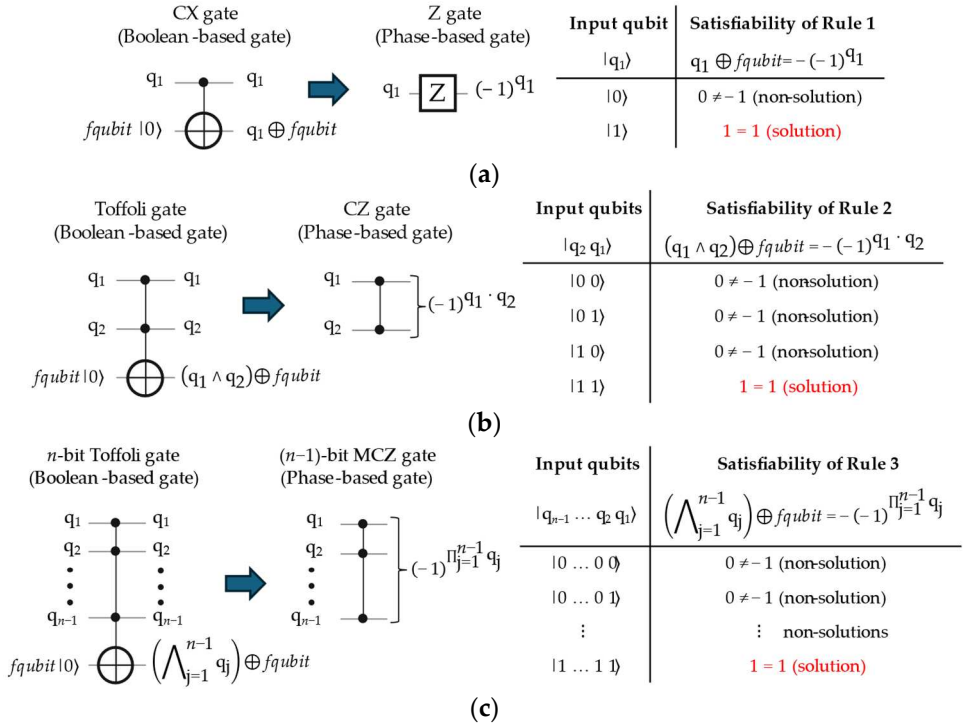

Figure 3. Schematics of our generalized transformation rules from the quantum Boolean-based gates (Feynman, Toffoli, and n-bit Toffoli gates) to the quantum Phase-based gates (Z, CZ, and $(n-1)$-bit MCZ gates) with their truth tables, where $n \geq 3$ qubits (q inputs + *fqubit*), and texts in red indicate the solutions: (**a**) Rule 1 transforms a Feynman (CX) gate into a Z gate, (**b**) Rule 2 transforms a Toffoli gate into a CZ gate, and (**c**) Rule 3 transforms an n-bit Toffoli gate to an $(n-1)$-bit MCZ gate. Note that the total number of qubits is reduced by one after applying these rules, i.e., the *fqubit* is removed.

After applying our generalized transformation rules on the Boolean oracle in ESOP (or DSOP) structure, as stated in Equation (2) above and shown in Figure 2c, the resultant quantum circuit of the Phase oracle is then simply constructed, as illustrated in Figure 4.

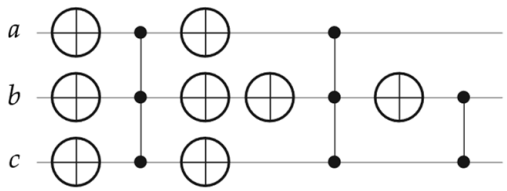

Figure 4. The Phase oracle for the classical Boolean problem $(a \vee b \vee \neg c) \wedge (\neg a \vee c) \wedge (\neg b \vee c)$, after applying our generalized transformation rules on its Boolean oracle in ESOP structure, such that (i) two 4-bit Toffoli gates are transformed into two 3-bit MCZ gates and one Toffoli gate into one CZ gate, (ii) all ancilla qubits (including *fqubit*) are removed, and (iii) there is no mirror for this oracle.

Table 1 summarizes the advantages of converting the Boolean oracle in POS structure to the Boolean oracle in ESOP structure, and transforming the Boolean oracle in ESOP structure to the Phase oracle, in the context of (i) the utmost removal of all ancilla qubits (including the *fqubit*), i.e., the width of the final quantum circuit is reduced, and (ii) the dramatic minimization of multi-controlled quantum gates (after transforming the POS structure into the ESOP structure and then applying the three aforementioned generalized transformation rules), i.e., the depth of the quantum circuit is shrunk.

Table 1. Comparison of the number of qubits and quantum gates for the Boolean and Phase oracles for the Boolean problem of Equation (1) stated above, after applying the generalized transformation rules.

Oracular Problems	Number of Qubits			Number of Multi-Controlled Gates			Quantum Circuit Required a Mirror?
	Inputs	Ancillae (with *fqubit*)	Total	Feynman (CX)	3-bit Toffoli	4-bit Toffoli	
Boolean oracle in POS structure	3	4	7	0	4	3	Yes
Boolean oracle in ESOP structure	3	1	4	0	1	2	No
Phase oracle	3	0	3	1 (as a CZ)	2 (as a CCZ)	0	No

3.3. Generating Hamiltonians (H_C and H_M) from Phase Oracles

Hadfield discussed, in [30], the composition rules for generating H_C from a set of Hamiltonians (H_f), which represent a variety of Boolean functions as simpler clauses, as stated in Table 2. In this article, to generate H_C and H_M from Phase oracles, we generalize some of Hadfield's Boolean-based composition rules (H_f) to Phase-based composition rules, which we termed the "generalized composition rules (H_g)". Based on our proposed three generalized transformation rules stated above, four generalized composition rules (H_g) are derived from H_f, as expressed in Table 3, where Rules 1, 2, and 3 of H_g are simply inverting the signs ($\pm$) of their corresponding H_f, for both identity (I) and RZ gates.

Table 2. Some of Hadfield's Boolean-based composition rules (H_f) [30], where j and k are the indices of input qubits (q), Z_j is the RZ gate applied on q_j, and *fqubit* initially sets to the $|0\rangle$ state.

Gate	Type	$f(x)$	H_f
Feynman (CX)	Boolean	$q_j \oplus fqubit = q_j$	$\frac{1}{2}\,I - \frac{1}{2}Z_j$
Toffoli	Boolean	$q_j \wedge q_k$	$\frac{1}{4}\,I - \frac{1}{4}(Z_j + Z_k - Z_jZ_k)$
n-bit Toffoli	Boolean	$\bigwedge_{j=1}^{n-1} q_j$	$\frac{1}{2^{n-1}}\prod_{j=1}^{n-1}(I - Z_j)$

Table 3. Our proposed generalized composition rules (H_g) for Phase oracles, where j and k are the indices of input qubits (q), Z_j is the RZ gate applied on q_j, $Q = \{q_j, q_k \ldots q_j q_k \ldots\}$, and $Z_Q = \{Z_j, Z_k \ldots Z_j Z_k \ldots\}$.

Rules	Gate	Type	$g(x)$	H_g
Rule 1	Pauli-Z (Z)	Phase	$(-1)^{q_j}$	$-\frac{1}{2}\,I + \frac{1}{2}Z_j$
Rule 2	CZ	Phase	$(-1)^{q_j \cdot q_k}$	$-\frac{1}{4}\,I + \frac{1}{4}(Z_j + Z_k - Z_jZ_k)$
Rule 3	MCZ	Phase	$(-1)^{\prod_{j=1}^{n} q_j}$	$\frac{1}{2^n}\prod_{j=1}^{n}(-1)^{j+1}(I - Z_j)$
Rule 4	Pauli-X (X)	Phase	$(-1)^{\forall j \in Q}$	Invert signs ($\pm$) of all jth qubits in Z_Q

Rule 1: Performs Rule 1 of the generalized transformation rules, as stated in Equation (3) above and previously shown in Figure 3a.

Rule 2: Performs Rule 2 of the generalized transformation rules, as stated in Equation (4) above and previously shown in Figure 3b.

Rule 3: Performs Rule 3 of the generalized transformation rules, as stated in Equation (5) above and previously shown in Figure 3c.

Rule 4: Inverts all signs ($\pm$) of generated RZ gates in H_g (Rules 1, 2, and 3), since the X gates are proposed here to only invert the phases of qubits in an H_g.

Consequently, from Table 3, the four generalized compositions rules (H_g) will be then directly applied to the Phase oracle (shown in Figure 4) to generate H_C and calculate its v coefficient, as demonstrated in Figure 5 and expressed in the following steps (starting from the left side of this figure).

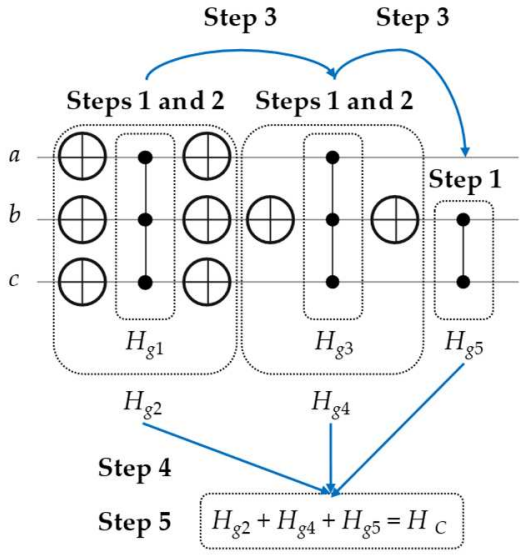

Figure 5. Steps of generating the Hamiltonian (H_C) using our generalized composition rules (H_g) from the Phase oracle shown in Figure 4, such that $H_C = (H_{g1} \rightarrow H_{g2}) + (H_{g3} \rightarrow H_{g4}) + H_{g5}$.

1. Construct one H_g for one Z, CZ, or MCZ, using Rule 1, Rule 2, or Rule 3, respectively.
2. If there are X gates (with their mirrored gates) surrounding Z, CZ, or MCZ in Step 1, then apply Rule 4 on H_g from Step 1 to construct a new H_g. If not, proceed to Step 3.
3. Repeat Steps 1 and 2 for another H_g until there are no remaining Z, CZ, and MCZ.
4. Group all constructed H_g into one Hamiltonian, which is H_C.
5. Calculate (add or subtract) all the identical terms of H_C to find the v coefficient.

From Table 3 and Figure 5, the Hamiltonians (H_g) are step-by-step calculated, as stated in Equation (6) below. Next, the H_C is simply generated from $H_{g2} + H_{g4} + H_{g5}$, as expressed in Equation (7) below.

$$\begin{aligned}
H_{g_1} &= \tfrac{1}{8}((\mathrm{I}-\mathrm{Z}_a)(-\mathrm{I}+\mathrm{Z}_b)(\mathrm{I}-\mathrm{Z}_c)) = \tfrac{1}{8}(-\mathrm{I}+\mathrm{Z}_a+\mathrm{Z}_b+\mathrm{Z}_c-\mathrm{Z}_a\mathrm{Z}_b-\mathrm{Z}_a\mathrm{Z}_c-\mathrm{Z}_b\mathrm{Z}_c+\mathrm{Z}_a\mathrm{Z}_b\mathrm{Z}_c)\\
H_{g_2} &= \tfrac{1}{8}(-\mathrm{I}-\mathrm{Z}_a-\mathrm{Z}_b-\mathrm{Z}_c-\mathrm{Z}_a\mathrm{Z}_b-\mathrm{Z}_a\mathrm{Z}_c-\mathrm{Z}_b\mathrm{Z}_c-\mathrm{Z}_a\mathrm{Z}_b\mathrm{Z}_c)\\
H_{g_3} &= \tfrac{1}{8}((\mathrm{I}-\mathrm{Z}_a)(-\mathrm{I}+\mathrm{Z}_b)(\mathrm{I}-\mathrm{Z}_c)) = \tfrac{1}{8}(-\mathrm{I}+\mathrm{Z}_a+\mathrm{Z}_b+\mathrm{Z}_c-\mathrm{Z}_a\mathrm{Z}_b-\mathrm{Z}_a\mathrm{Z}_c-\mathrm{Z}_b\mathrm{Z}_c+\mathrm{Z}_a\mathrm{Z}_b\mathrm{Z}_c)\\
H_{g_4} &= \tfrac{1}{8}(-\mathrm{I}+\mathrm{Z}_a-\mathrm{Z}_b+\mathrm{Z}_c+\mathrm{Z}_a\mathrm{Z}_b-\mathrm{Z}_a\mathrm{Z}_c+\mathrm{Z}_b\mathrm{Z}_c-\mathrm{Z}_a\mathrm{Z}_b\mathrm{Z}_c)\\
H_{g_5} &= \tfrac{1}{4}((\mathrm{I}-\mathrm{Z}_b)(-\mathrm{I}+\mathrm{Z}_c)) = \tfrac{1}{4}(-\mathrm{I}+\mathrm{Z}_b+\mathrm{Z}_c-\mathrm{Z}_b\mathrm{Z}_c)
\end{aligned} \tag{6}$$

$$H_C = H_{g_2} + H_{g_4} + H_{g_5} = -\frac{1}{2}\mathrm{I}+\frac{1}{4}\mathrm{Z}_c-\frac{1}{4}(\mathrm{Z}_a\mathrm{Z}_c+\mathrm{Z}_b\mathrm{Z}_c)-\frac{1}{4}\,\mathrm{Z}_a\mathrm{Z}_b\mathrm{Z}_c \tag{7}$$

Hence, from Equation (7) above, $v = [v_1, v_2, v_3, v_4] = [-\frac{1}{2}, \frac{1}{4}, -\frac{1}{4}, -\frac{1}{4}]$, where v_1 is for all non-connected input qubits (as the identity 'I'), v_2 is for c input qubit only, i.e., $\mathrm{RZ}_c(0.25{\cdot}\gamma)$, v_3 is for only two connected input qubits {a and c; b and c}, i.e., $\mathrm{RZ}_a\mathrm{Z}_c(-0.25{\cdot}\gamma)$ and $\mathrm{RZ}_b\mathrm{Z}_c(-0.25{\cdot}\gamma)$, and v_4 is for all connected input qubits {a, b, c}, i.e., $\mathrm{RZ}_a\mathrm{Z}_b\mathrm{Z}_c(-0.25{\cdot}\gamma)$. For a practical quantum implementation of QAOA as well as our introduced BHT-QAOA, H_C in Equation (7) is rewritten as the following format for the sequence of three qubits {c, b, a}, as stated in Equation (8) below, where 'ZII' is equivalent to RZ_c, 'ZIZ' is equivalent to $\mathrm{RZ}_a\mathrm{Z}_c$, 'ZZZ' is equivalent to $\mathrm{RZ}_a\mathrm{Z}_b\mathrm{Z}_c$, and so on.

$$H_C = (['\mathrm{III}', '\mathrm{ZII}', '\mathrm{ZIZ}', '\mathrm{ZZI}', '\mathrm{ZZZ}'], \mathit{coeffs} = [-0.5, 0.25, -0.25, -0.25, -0.25]) \tag{8}$$

Because β (as a set of rotational angles of H_M) rotates between [0, π] [1,2], we set its coefficient (ω) to cover the entire range between [0, 2π] for possible phase values of RX gates in H_M, to find all optimized approximated solutions for an arbitrary classical Boolean problem. In other words, ω (as the coefficient of β) is initially set to '2.0' for all n numbers

of RX($\omega \cdot \beta$) gates in H_M, where n is the total number of input qubits. Similar to the practical implementation of H_C, H_M is rewritten as the following format for the sequence of three qubits {c, b, a}, as stated in Equation (9) below, where 'XII' is equivalent to RX_c, 'IXI' is equivalent to RX_b, and 'IIX' is equivalent to RX_a.

$$H_M = (['XII', 'IXI', 'IIX'], coeffs = [2.0, 2.0, 2.0]) \quad (9)$$

From Equations (8) and (9) above, the quantum circuit of BHT-QAOA for the generated Hamiltonians (H_C and H_M) of the Boolean problem $(a \vee b \vee \neg c) \wedge (\neg a \vee c) \wedge (\neg b \vee c)$ is eventually constructed for one repetition ($p = 1$), as illustrated in Figure 6.

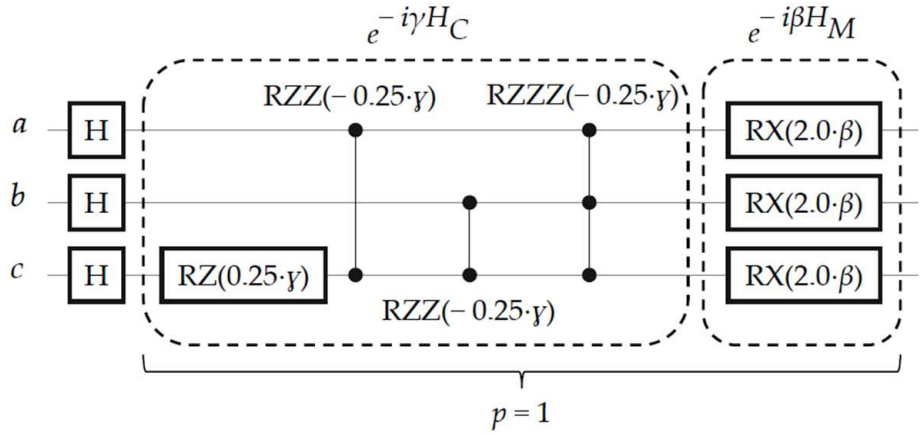

Figure 6. The quantum circuit for the classical Boolean problem of $(a \vee b \vee \neg c) \wedge (\neg a \vee c) \wedge (\neg b \vee c)$ after applying our generalized transformation rules (H_g) to generate two Hamiltonians (H_C and H_M) in one repetition (p), where H is the Hadamard gate, H_C is ('III', 'ZII', 'ZIZ', 'ZZI', 'ZZZ') with its coefficient $v = [-\frac{1}{2}, \frac{1}{4}, -\frac{1}{4}, -\frac{1}{4}]$, H_M is ('XII', 'IXI', 'IIX') with its coefficient $\omega = 2.0$, and γ and β are the optimization angular parameters for H_C and H_M, respectively.

3.4. Architecture of BHT-QAOA

After transforming an arbitrary classical Boolean problem to the Hamiltonians (H_C and H_M) and calculating their coefficients (v and ω), respectively, the numerical values of γ and β are initially randomized and then plugged into the architecture of BHT-QAOA for first execution, as shown in Figure 7. Subsequently, the SciPy optimization minimizer [31] is utilized to optimize these numerical values (γ and β) for better-approximated solutions, in a number of function evaluations (*nfev*), by employing three cofactors as follows.

1. H_C and H_M (in a number of p), as the "objective function" needs to be minimized.
2. Measured solutions of BHT-QAOA, as the "energy cost" of the objective function.
3. Previously calculated γ and β, as their "numerical values" need to be optimized.

For the SciPy optimization minimizer, we use the constrained optimization by linear approximation (COBYLA) algorithm [14,16,31], which is a parametric iterative method for derivative-free constrained optimization that updates and minimizes the approximation values (as γ and β) for an objective function (as H_C and H_M of lower energy cost). On the other hand, our future work will focus on finding a cost-effective minimizer algorithm, to efficiently implement BHT-QAOA with a smaller value of p (in the quantum domain) and fewer *nfev* (in the classical domain).

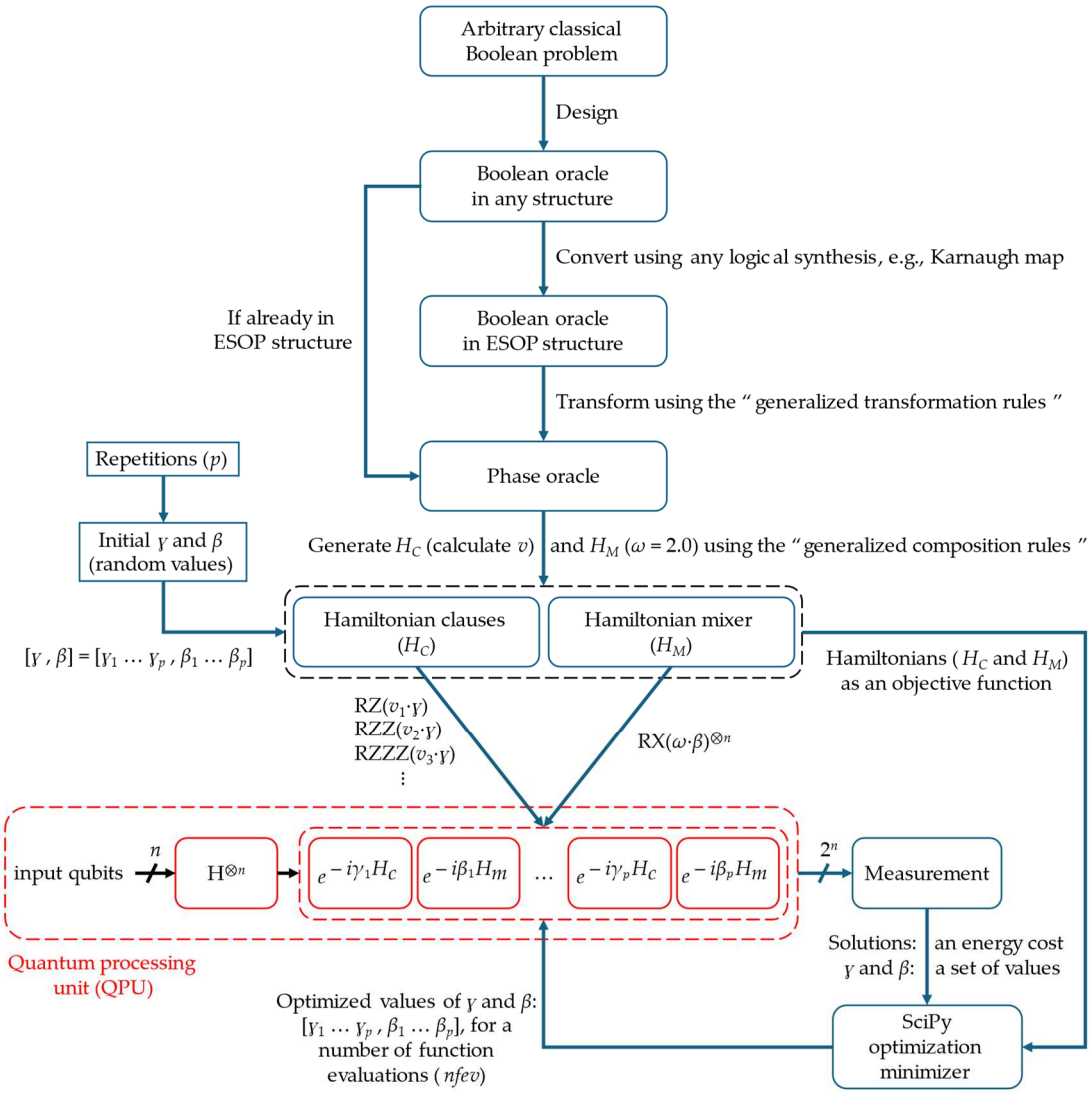

Figure 7. The architecture of our Boolean–Hamiltonian Transform for QAOA (BHT-QAOA) to solve arbitrary classical Boolean problems as Hamiltonians (H_C and H_M). The BHT-QAOA is mainly grouped into two processing domains: (i) the classical processing domain as denoted by blue, and (ii) the quantum processing domain as denoted by red.

4. Results and Discussion

Arbitrary classical Boolean problems (applications) are designed as Boolean oracles (in different structures), and these Boolean oracles are then solved using our introduced BHT-QAOA for p repetitions, where $p \geq 1$. In our experiments, the ibm_brisbane [50] QPU of 127 qubits performs the quantum processing domain of BHT-QAOA, i.e., executes the quantum circuit of an application in p repetitions, and the SciPy minimizer function performs the classical processing domain of BHT-QAOA, i.e., optimizes the numerical values of γ and β based on the minimized energy cost of their Hamiltonians (H_C and H_M), for a number of function evaluations (*nfev*).

Because of our IBM Quantum Platform account limitations, the complete architecture of BHT-QAOA (shown in Figure 7) is completely simulated in the classical domain using IBM quantum libraries (Qiskit, AerSimulator, and Aer-EstimatorV2 [50–52]), for 1024 resampling times, which are the so-called "shots" [53]. After this classically simulated BHT-QAOA, the final optimized numerical values of γ and β (from the SciPy minimizer) are plugged into their respective Hamiltonians (H_C and H_M) of the quantum circuit for an application, in which it is then executed once with ibm_brisbane QPU.

Please observe that due to the limited physical connectivity of four neighboring qubits for the recent quantum layouts of IBM QPUs, the constructed Boolean oracles (in various structures) for the following applications must have n input qubits and m ancilla qubits (including *fqubit*), where $2 \leq n \leq 4$ and $m \geq 1$. The m ancilla qubits (i) do not affect the fidelity of the final optimized approximated solutions of applications utilizing the BHT-QAOA and (ii) have no relation to the limited physical connectivity of any IBM QPU, because all these constructed Boolean oracles are transformed into Phase oracles and then into Hamiltonians (H_C and H_M). In other words, Hamiltonians do not have any ancilla qubits in their quantum circuits, and all m ancilla qubits are removed. Figure 8 demonstrates the classical representations and the quantum circuits of Boolean oracles for these applications, as follows.

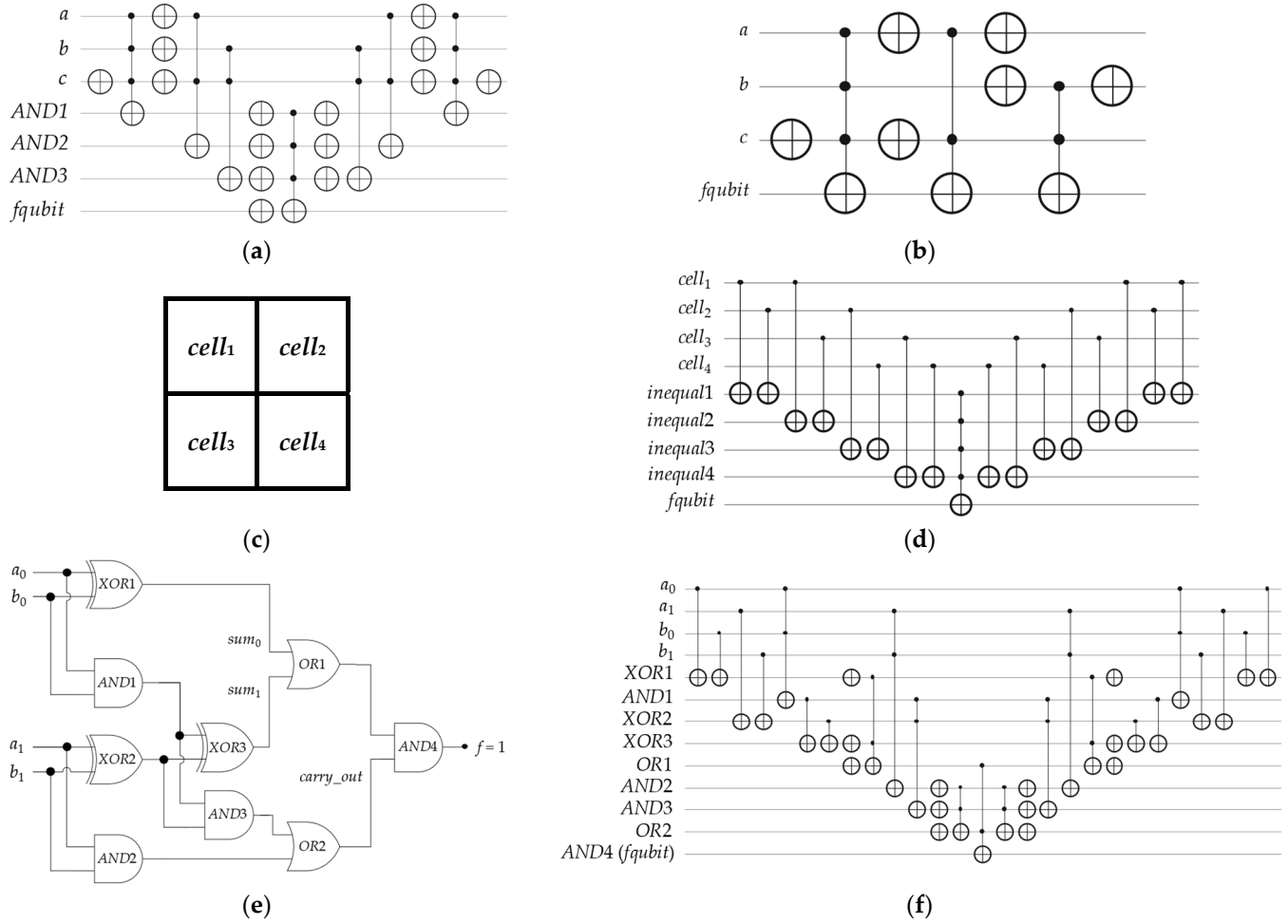

Figure 8. Schematics of the classical representations and the quantum circuits for arbitrary Boolean problems: ((**a**), *upper-left*) the Boolean oracle in SOP structure representing Equation (10) above, ((**b**), *upper-right*) the Boolean oracle in ESOP structure representing Equation (11) above, ((**c**), *middle-left*) the board layout of 2 × 2 Sudoku, ((**d**), *middle-right*) the Boolean oracle in CNF-XOR SAT structure of 2 × 2 Sudoku, ((**e**), *bottom-left*) the classical 4-bit conditioned half-adder for two 2-bit numbers (A = a_1a_0 and B = b_1b_0), and ((**f**), *bottom-right*) the Boolean oracle in a mixed structure representing Equation (13) above.

1. An arbitrary Boolean problem in POS structure, as stated in Equation (1) above and shown in Figure 1.
2. An arbitrary Boolean problem in SOP structure, as stated in Equation (10) below and shown in Figure 8a.

$$(a \wedge b \wedge \neg\, c) \vee (\neg\, a \wedge c) \vee (\neg\, b \wedge c) \tag{10}$$

3. An arbitrary Boolean problem in ESOP structure, as stated in Equation (11) below and shown in Figure 8b.

$$(a \wedge b \wedge \neg\, c) \oplus (\neg\, a \wedge c) \oplus (\neg\, b \wedge c) \tag{11}$$

4. A 2 × 2 Sudoku game, which is the constraints satisfaction problem—satisfiability (CSP-SAT) [54,55], as stated in Equation (12) below and illustrated in Figure 8c,d.

$$(cell_1 \oplus cell_2) \wedge (cell_1 \oplus cell_3) \wedge (cell_2 \oplus cell_4) \wedge (cell_3 \oplus cell_4) \tag{12}$$

5. A 4-bit conditioned half-adder digital circuit, which is ORing two 1-bit sums and then ANDing them with one 1-bit carry-out, as stated in Equation (13) below and demonstrated in Figure 8e,f.

$$[(a_0 \oplus b_0) \vee ((a_0 \wedge b_0) \oplus (a_1 \oplus b_1))] \wedge [(a_1 \wedge b_1) \vee ((a_0 \wedge b_0) \wedge (a_1 \oplus b_1))] \tag{13}$$

Table 4 states the removal of all ancilla qubits (including *fqubit*) and the reduction number of quantum gates, after applying the generalized transformation rules to form Phase oracles. These Phase oracles are then transformed into the Hamiltonians (H_C and H_M) of BHT-QAOA. Note that, in Table 4, (i) a Pauli-X (X) gate in a Boolean oracle remains as-is for a Phase oracle, (ii) a Feynman (CX) gate in a Boolean oracle is equivalent to the controlled Pauli-Z (CZ) gate in a Phase oracle, and (iii) an n-bit Toffoli gate in a Boolean oracle is equivalent to an n-bit MCZ gate in a Phase oracle, where $n \geq 3$ qubits.

Table 4. The effect of generalized transformation rules on removing all ancilla qubits (including *fqubit*) and reducing the numbers of quantum gates for the final Phase oracles for BHT-QAOA.

Qubits and Quantum Gates (Entities)	**Entities in a Boolean Oracle → Entities in a Phase Oracle**				
	Arbitrary Problem in POS	**Arbitrary Problem in SOP**	**Arbitrary Problem in ESOP**	**2 × 2 Sudoku Game**	**4-bit Conditioned Half-Adder Circuit**
Input qubits	3 → 3	3 → 3	3 → 3	4 → 4	4 → 4
Ancilla qubits	4 → 0	4 → 0	1 → 0	5 → 0	9 → 0
Pauli-X (X)	16 → 8	15 → 6	6 → 6	0 → 8	12 → 2
Feynman (CX)	0 → 1	0 → 1	0 → 2	16 → 0	12 → 0
3-bit Toffoli	4 → 2	4 → 2	2 → 1	–	11 → 1
4-bit Toffoli	3 → 0	3 → 0	1 → 0	0 → 2	0 → 1
5-bit Toffoli	–	–	–	1 → 0	–

On the one hand, the classically simulated BHT-QAOA successfully optimizes the numerical values of γ and β and finds all approximated solutions for these applications, as a proof of concept for utilizing our introduced BHT-QAOA in solving arbitrary classical Boolean problems in the simulated classical-quantum domain.

On the other hand, the quantum circuit of every application (using the simulated optimized numerical values of γ and β) is executed once with ibm_brisbane QPU for solution fidelity, as a proof of concept for utilizing BHT-QAOA to solve arbitrary classical Boolean problems in the hybrid classical-quantum domain.

Figure 9 depicts the final measured solutions for every application from the real quantum executions using ibm_brisbane QPU. In Figure 9, the first measured bit (the upper-right) of a solution is equivalent to the first input qubit, and the last measured bit (the bottom-left) of a solution is equivalent to the last input qubit for the Boolean oracle of an arbitrary application.

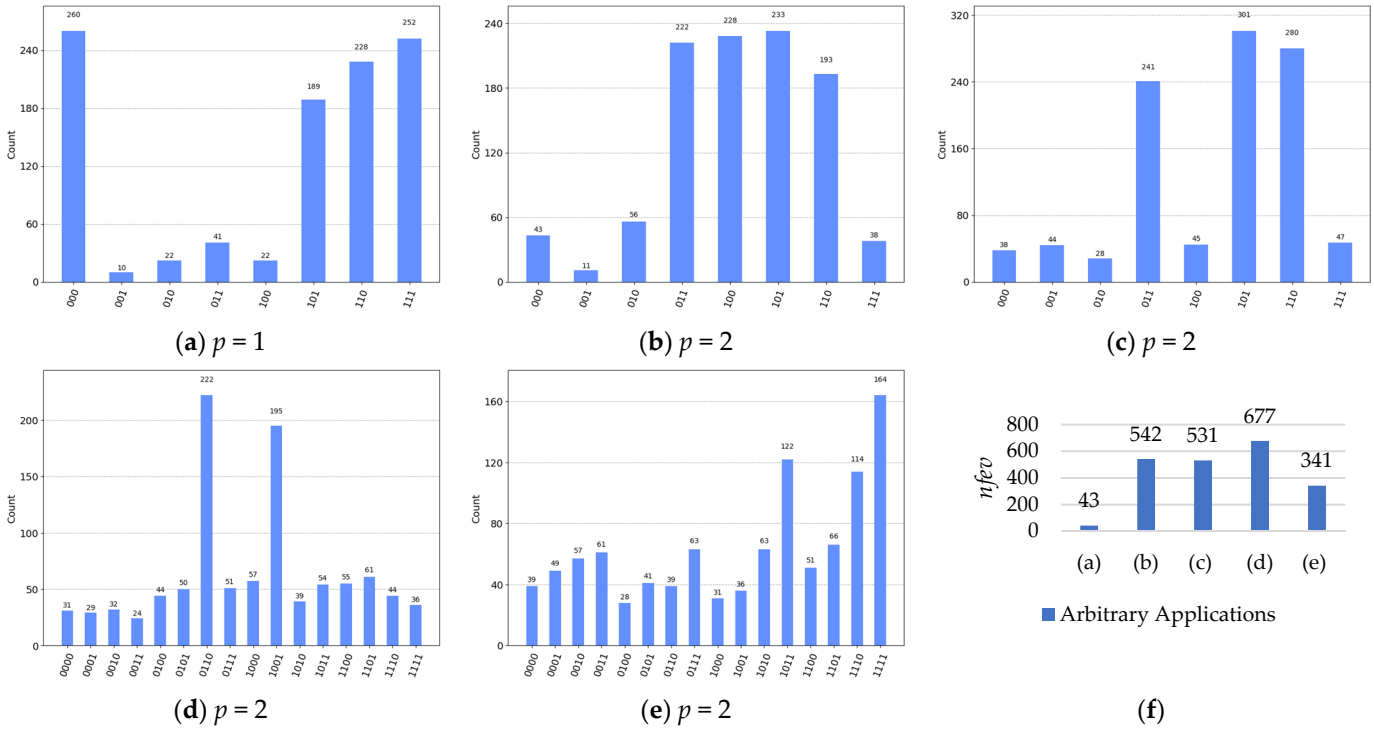

Figure 9. The final measured solutions for arbitrary applications executed with ibm_brisbane QPU (for 1024 shots): ((**a**), *upper-left*) the four solutions $\{\bar{a}\bar{b}\bar{c}, a\bar{b}\bar{c}, \bar{a}\bar{b}c, abc\}$ for the Boolean oracle in POS structure of Equation (1) above, ((**b**), *upper-middle*) the four solutions $\{ab\bar{c}, \bar{a}\bar{b}\bar{c}, a\bar{b}\bar{c}, \bar{a}\bar{b}c\}$ for the Boolean oracle in SOP structure of Equation (10) above, ((**c**), *upper-right*) the three solutions $\{ab\bar{c}, a\bar{b}\bar{c}, \bar{a}\bar{b}c\}$ for the Boolean oracle in ESOP structure of Equation (11) above, ((**d**), *bottom-left*) the two permutative solutions {solution 1: $cell_1 = cell_4 = 0$ and $cell_2 = cell_3 = 1$; solution 2: $cell_1 = cell_4 = 1$ and $cell_2 = cell_3 = 0$} for the 2×2 Sudoku game of Equation (12) above, ((**e**), *bottom-middle*) the three solutions as two 2-bit numbers $\{a_0a_1\overline{b_0}b_1, \overline{a_0}a_1b_0b_1, a_0a_1b_0b_1\}$ for the 4-bit conditioned half-adder of Equation (13) above, and ((**f**), *bottom-right*) the required *nfev* for the SciPy optimization minimizer to successfully optimize the numerical values of γ and β for the above-mentioned applications, in the subfigures (**a**–**e**).

As illustrated in Figure 9a–e, the histograms represent the final measured outputs for the arbitrary Boolean problems (as applications), and each histogram of an application denotes the results of (a) four solutions $\{\bar{a}\bar{b}\bar{c}, a\bar{b}\bar{c}, \bar{a}\bar{b}c, abc\}$ for the Boolean oracle in POS structure of Equation (1) above, (b) four solutions $\{ab\bar{c}, \bar{a}\bar{b}\bar{c}, a\bar{b}\bar{c}, \bar{a}\bar{b}c\}$ for the Boolean oracle in SOP structure of Equation (10) above, (c) three solutions $\{ab\bar{c}, a\bar{b}\bar{c}, \bar{a}\bar{b}c\}$ for the Boolean oracle in ESOP structure of Equation (11) above, (d) two permutative solutions {solution 1: $cell_1 = cell_4 = 0$ and $cell_2 = cell_3 = 1$; solution 2: $cell_1 = cell_4 = 1$ and $cell_2 = cell_3 = 0$} for the 2×2 Sudoku of Equation (12) above, and (e) three solutions $\{a_0a_1\overline{b_0}b_1, \overline{a_0}a_1b_0b_1, a_0a_1b_0b_1\}$ for the 4-bit conditioned half-adder of Equation (13) above.

Note that, in Figure 9, (i) the first measured bit (the upper-right in a histogram) is equivalent to the first input qubit of QAOA, (ii) the final measured bit (the bottom-left in a histogram) is equivalent to the final input qubit of QAOA, (iii) p states the number of repetitions for the Hamiltonians (H_C and H_M) in the quantum circuit of QAOA, (iv) *nfev* is the number of function evaluations from the SciPy optimization minimizer, (v) 'Count' is the total number of 1024 shots for all input qubits, where the higher Count indicates a solution (as a higher probability of qubits measurement) and the lower Count indicates a non-solution (as a lower probability of qubits measurement), and (vi) $\bar{x}$ means x has a false binary value of '0'; otherwise, x has a true binary value of '1', where x is a literal (variable) in an equation that represents a Boolean problem (application), in the classical domain.

Please observe that the required *nfev* for the SciPy optimization minimizer varies and fluctuates, since such a minimizer mainly depends on two factors, as follows.

1. The initially randomized numerical values of γ and β for H_C and H_M, respectively.
2. The noisy simulation models (AerSimulator and Aer-EstimatorV2), which are utilized for simulating BHT-QAOA, in the classical domain.

Accordingly, the BHT-QAOA will provide broad opportunities to find all solutions for many classical Boolean problems constructed as Hamiltonians (H_C and H_M), which are neither designed nor solved using the standard QAOA [1,2]. Therefore, various classical Boolean problems for the applications of digital logic circuits, synthesizers, robotics, and machine learning can be realized as Hamiltonians and then solved using BHT-QAOA in the hybrid classical-quantum domain.

5. Conclusions

A new methodology is introduced to solve arbitrary classical Boolean problems as Hamiltonians (H_C and H_M), using the quantum approximate optimization algorithm (QAOA) [1,2]. Our methodology is termed the "Boolean-Hamiltonians Transform for QAOA" (BHT-QAOA), which is summarized as follows: (i) an arbitrary classical Boolean problem is expressed as a Boolean oracle in arbitrary structures, e.g., POS, SOP, ESOP, and XOR SAT, just to name a few, (ii) this Boolean oracle (in any structure) is converted into its equivalent Boolean oracle in ESOP structure, unless it was firstly constructed in ESOP structure, (iii) this Boolean oracle in ESOP structure is transformed into its equivalent Phase oracle based on our modified transformations of Toffoli gates, which are originally presented by Figgatt et al. [20], (iv) the Hamiltonians (H_C and H_M) are generated from this transformed Phase oracle based on our modified set of Hamiltonian compositions, which are originally presented by Hadfield [30], and (v) all execution steps of the standard QAOA are performed on the generated H_C and H_M. A classical optimization minimizer is utilized to find better-approximated solutions for an arbitrary classical Boolean problem based on the optimized numerical values of γ and β angles for H_C and H_M, respectively.

In BHT-QAOA, all ancilla qubits (including the output qubit) and the mirror (as the uncomputing part) of a quantum circuit will be completely removed when transforming a Boolean oracle (in any structure) into its equivalent Phase oracle. In other words, during the conversion and transformation steps of BHT-QAOA, the total number of utilized qubits will be dramatically reduced to the number of input qubits only, and the total number of quantum gates will be significantly minimized for the final quantum circuit of a Phase oracle, for an arbitrary classical Boolean problem.

In this article, arbitrary Boolean applications are constructed as Boolean oracles (in various structures), and then BHT-QAOA successfully finds all optimized approximated solutions for these applications using a classical optimization minimizer and an IBM quantum computer, since our introduced BHT-QAOA is considered as a hybrid classical-quantum algorithm. In conclusion, further classical Boolean problems can be constructed as Boolean oracles (in mixed structures) for the practical engineering applications in the topics of digital synthesizers, computer vision, robotics, and machine learning, just name a few, and BHT-QAOA will successfully solve such practical applications effectively in the hybrid classical-quantum domain.

Author Contributions: Conceptualization, A.A.-B.; methodology, A.A.-B.; formal analysis, A.A.-B. and M.P.; writing—original draft preparation, A.A.-B.; visualization, A.A.-B.; supervision, M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The original contributions presented in our study are included in this article; further inquiries can be directed to the corresponding author (A.A.-B.).

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

References

1. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm. *arXiv* **2014**. [CrossRef]
2. Farhi, E.; Goldstone, J.; Gutmann, S.; Neven, H. Quantum algorithms for fixed qubit architectures. *arXiv* **2017**. [CrossRef]
3. Goemans, M.X.; Williamson, D.P. 878-approximation algorithms for max cut and max 2sat. In Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing, Montréal, QC, Canada, 23–25 May 1994; pp. 422–431.
4. Rendl, F.; Rinaldi, G.; Wiegele, A. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Program.* **2010**, *121*, 307–335. [CrossRef]
5. Grover, L.K. A fast quantum mechanical algorithm for database search. In *Proceedings, 28th Annual ACM Symposium on the Theory of Computing*; ACM Press: New York, NY, USA, 1996; pp. 212–219.
6. Grover, L.K. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **1997**, *79*, 325. [CrossRef]
7. Grover, L.K. A framework for fast quantum mechanical algorithms. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*; ACM Digital Library: New York, NY, USA, 1998; pp. 53–62.
8. Al-Bayaty, A.; Perkowski, M. A concept of controlling Grover diffusion operator: A new approach to solve arbitrary Boolean-based problems. *Res. Sq.* **2023**. [CrossRef]
9. Moussa, C.; Wang, H.; Bäck, T.; Dunjko, V. Unsupervised strategies for identifying optimal parameters in quantum approximate optimization algorithm. *EPJ Quantum Technol.* **2022**, *9*, 11. [CrossRef]
10. Amosy, O.; Danzig, T.; Lev, O.; Porat, E.; Chechik, G.; Makmal, A. Iteration-free quantum approximate optimization algorithm using neural networks. *Quantum Mach. Intell.* **2024**, *6*, 38. [CrossRef]
11. Herrman, R.; Lotshaw, P.C.; Ostrowski, J.; Humble, T.S.; Siopsis, G. Multi-angle quantum approximate optimization algorithm. *Sci. Rep.* **2022**, *12*, 6781. [CrossRef] [PubMed]
12. Wurtz, J.; Lykov, D. Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs. *Phys. Rev. A* **2021**, *104*, 052419. [CrossRef]
13. Crooks, G.E. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv* **2018**. [CrossRef]
14. Fernández-Pendás, M.; Combarro, E.F.; Vallecorsa, S.; Ranilla, J.; Rúa, I.F. A study of the performance of classical minimizers in the quantum approximate optimization algorithm. *J. Comput. Appl. Math.* **2022**, *404*, 113388. [CrossRef]
15. Powell, M.J.D.; Gomez, S. *Advances in Optimization and Numerical Analysis*; Gomez, S., Hennart, J.P., Eds.; Springer: Dordrecht, The Netherlands, 1994; Volume 4, pp. 51–67. ISBN 978-0792326731.
16. Powell, M.J.D. A view of algorithms for optimization without derivatives. *Math. Today-Bull. Inst. Math. Its Appl.* **2007**, *43*, 170–174.
17. Cerezo, M.; Arrasmith, A.; Babbush, R.; Benjamin, S.C.; Endo, S.; Fujii, K.; McClean, J.R.; Mitarai, K.; Yuan, X.; Cincio, L.; et al. Variational quantum algorithms. *Nat. Rev. Phys.* **2021**, *3*, 625–644. [CrossRef]
18. Wecker, D.; Hastings, M.B.; Troyer, M. Progress towards practical quantum variational algorithms. *Phys. Rev. A* **2015**, *92*, 042303. [CrossRef]
19. Tilly, J.; Chen, H.; Cao, S.; Picozzi, D.; Setia, K.; Li, Y.; Grant, E.; Wossnig, L.; Rungger, I.; Booth, G.H.; et al. The variational quantum eigensolver: A review of methods and best practices. *Phys. Rep.* **2022**, *986*, 1–128. [CrossRef]
20. Figgatt, C.; Maslov, D.; Landsman, K.A.; Linke, N.M.; Debnath, S.; Monroe, C. Complete 3-qubit Grover search on a programmable quantum computer. *Nat. Commun.* **2017**, *8*, 1918. [CrossRef] [PubMed]
21. Wakerly, J.F. *Digital Design: Principles and Practices*, 4th ed.; Pearson Education: New Delhi, India, 2014; ISBN 978-0131863897.
22. Zhang, L.X.; Huang, W. A note on the invariance principle of the product of sums of random variables. *Electron. Commun. Probab.* **2007**, *12*, 59–64.
23. Zimmermann, R.; Tran, D.Q. Optimized synthesis of sum-of-products. In *Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*; IEEE: New York, NY, USA, 2003; pp. 867–872.
24. Mishchenko, A.; Perkowski, M. Fast heuristic minimization of exclusive sums-of-products. In *5th Int. Workshop on Applications of the Reed-Muller Expansion in Circuit Design*; Mississippi State University: Starkville, MS, USA, 2001; pp. 242–250.
25. Sasao, T. EXMIN2: A simplification algorithm for exclusive-or-sum-of-products expressions for multiple-valued-input two-valued-output functions. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **1993**, *12*, 621–632. [CrossRef]
26. Ibrahimi, M.; Kanoria, Y.; Kraning, M.; Montanari, A. The set of solutions of random XORSAT formulae. In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms; Siam: Bangkok, Thailand, 2012; pp. 760–779.
27. Soos, M.; Meel, K.S. BIRD: Engineering an efficient CNF-XOR SAT solver and its applications to approximate model counting. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 1592–1599. [CrossRef]
28. Stankovic, R.S.; Sasao, T. A discussion on the history of research in arithmetic and Reed-Muller expressions. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2001**, *20*, 1177–1179. [CrossRef]
29. Kurgalin, S.; Borzunov, S. *Concise Guide to Quantum Computing: Algorithms, Exercises, and Implementations*; Springer: Cham, Switzerland, 2021; Volume 7, pp. 37–43, ISBN 978-3030650513.
30. Hadfield, S. On the representation of Boolean and real functions as Hamiltonians for quantum computing. *ACM Trans. on Quantum Comput. (TQC)* **2021**, *2*, 1–21. [CrossRef]

31. Lavrijsen, W.; Tudor, A.; Müller, J.; Iancu, C.; De Jong, W. Classical optimizers for noisy intermediate-scale quantum devices. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*; IEEE: New York, NY, USA, 2020; pp. 267–277.
32. Boulebnane, S.; Montanaro, A. Solving Boolean satisfiability problems with the quantum approximate optimization algorithm. *arXiv* **2022**. [CrossRef]
33. Akshay, V.; Philathong, H.; Morales, M.E.; Biamonte, J.D. Reachability deficits in quantum approximate optimization. *Phys. Rev. Lett.* **2020**, *124*, 090504. [CrossRef] [PubMed]
34. Mandl, A.; Barzen, J.; Bechtold, M.; Leymann, F.; Wild, K. Amplitude amplification-inspired QAOA: Improving the success probability for solving 3sat. *Quantum Sci. Technol.* **2024**, *9*, 015028. [CrossRef]
35. Lin, C.Y.Y.; Zhu, Y. Performance of QAOA on typical instances of constraint satisfaction problems with bounded degree. *arXiv* **2016**. [CrossRef]
36. Bärtschi, A.; Eidenbenz, S. Grover mixers for QAOA: Shifting complexity from mixer design to state preparation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*; IEEE: New York, NY, USA, 2020; pp. 72–82.
37. Zhang, Z.; Paredes, R.; Sundar, B.; Quiroga, D.; Kyrillidis, A.; Duenas-Osorio, L.; Pagano, G.; Hazzard, K.R. Grover-QAOA for 3-sat: Quadratic speedup, fair-sampling, and parameter clustering. *arXiv* **2024**. [CrossRef]
38. Weidenfeller, J.; Valor, L.C.; Gacon, J.; Tornow, C.; Bello, L.; Woerner, S.; Egger, D.J. Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware. *Quantum* **2022**, *6*, 870. [CrossRef]
39. Blekos, K.; Brand, D.; Ceschini, A.; Chou, C.H.; Li, R.H.; Pandya, K.; Summer, A. A review on quantum approximate optimization algorithm and its variants. *Phys. Rep.* **2024**, *1068*, 1–66. [CrossRef]
40. Govia, L.C.G.; Poole, C.; Saffman, M.; Krovi, H.K. Freedom of the mixer rotation axis improves performance in the quantum approximate optimization algorithm. *Phys. Rev. A* **2021**, *104*, 062428. [CrossRef]
41. Bravyi, S.; Kliesch, A.; Koenig, R.; Tang, E. Obstacles to variational quantum optimization from symmetry protection. *Phys. Rev. Lett.* **2020**, *125*, 260505. [CrossRef] [PubMed]
42. Golden, J.; Bärtschi, A.; O'Malley, D.; Eidenbenz, S. Threshold-based quantum optimization. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*; IEEE: New York, NY, USA, 2021; pp. 137–147.
43. Wang, Z.; Rubin, N.C.; Dominy, J.M.; Rieffel, E.G. XY mixers: Analytical and numerical results for the quantum alternating operator ansatz. *Phys. Rev. A* **2020**, *101*, 012320. [CrossRef]
44. Vijendran, V.; Das, A.; Koh, D.E.; Assad, S.M.; Lam, P.K. An expressive ansatz for low-depth quantum approximate optimisation. *Quantum Sci. Technol.* **2024**, *9*, 025010. [CrossRef]
45. Sarmina, B.G.; Sun, G.H.; Dong, S.H. Principal component analysis and t-distributed stochastic neighbor embedding analysis in the study of quantum approximate optimization algorithm entangled and non-entangled mixing operators. *Entropy* **2023**, *25*, 1499. [CrossRef] [PubMed]
46. Yarkoni, S.; Raponi, E.; Bäck, T.; Schmitt, S. Quantum annealing for industry applications: Introduction and review. *Rep. Prog. Phys.* **2022**, *85*, 104001. [CrossRef]
47. Morita, S.; Nishimori, H. Mathematical foundation of quantum annealing. *J. Math. Phys.* **2008**, *49*, 125210. [CrossRef]
48. Ebendt, R.; Fey, G.; Drechsler, R. *Advanced BDD Optimization*; Springer: Dordrecht, The Netherlands, 2005; ISBN 978-0387254531.
49. Wille, R.; Drechsler, R. Effect of BDD optimization on synthesis of reversible and quantum logic. *Electron. Notes Theor. Comput. Sci.* **2010**, *253*, 57–70. [CrossRef]
50. Karimi, N.; Elyasi, S.N.; Yahyavi, M. Implementation and measurement of quantum entanglement using IBM quantum platforms. *Phys. Scr.* **2024**, *99*, 045121. [CrossRef]
51. Wille, R.; Van Meter, R.; Naveh, Y. IBM's Qiskit tool chain: Working with and developing for real quantum computers. In *2019 Design, Automation & Test in Europe Conf. & Exhibition (DATE)*; IEEE: New York, NY, USA, 2019; pp. 1234–1240.
52. Georgopoulos, K.; Emary, C.; Zuliani, P. Modeling and simulating the noisy behavior of near-term quantum computers. *Phys. Rev. A* **2021**, *104*, 062432. [CrossRef]
53. Rao, P.; Yu, K.; Lim, H.; Jin, D.; Choi, D. Quantum amplitude estimation algorithms on IBM quantum devices. In *Quantum Communications and Quantum Imaging XVIII*; SPIE: St Bellingham, WA, USA, 2020; Volume 11507, pp. 49–60.
54. Simonis, H. Sudoku as a constraint problem. In *CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems*; Citeseer: Sitges, Spain, 2005; pp. 13–27.
55. Lynce, I.; Ouaknine, J. Sudoku as a SAT problem. In Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics (AI&M), Fort Lauderdale, FL, USA, 4–6 January 2006.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.