Finding Optimal Implementations of Non-native CNOT Gates using SAT

Philipp Niemann[1,2], Luca Müller[1,2], and Rolf Drechsler[1,2]

[1] Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
[2] Department of Computer Science, University of Bremen, Bremen, Germany
{pniemann,lucam,drechsler}@uni-bremen.de

Abstract. Quantum computer architectures place restrictions on the availability of quantum gates. While single-qubit gates are usually available on every qubit, multi-qubit gates like the CNOT gate can only be applied to a subset of all pairs of qubits. Thus, a given quantum circuit usually needs to be transformed prior to its execution in order to satisfy these restrictions. Existing transformation approaches mainly focus on using SWAP gates to enable the realization of CNOT gates that are not natively available in the architecture. As the SWAP gate is a composition of CNOT and single-qubit Hadamard gates, such methods may not yield a minimal solution. In this work, we propose a method to find an optimal implementation of non-native CNOTs, i.e. using the minimal number of native CNOT and Hadamard gates, by using a formulation as a Boolean Satisfiability (SAT) problem. While straightforward representations of quantum states, gates and circuits require an exponential number of complex-valued variables, the approach makes use of a dedicated representation that requires only a quadratic number of variables, all of which are Boolean. As confirmed by experimental results, the resulting problem formulation scales considerably well—despite the exponential complexity of the SAT problem—and enables us to determine significantly improved realizations of non-native CNOT gates for the 16-qubit IBM QX5 architecture.

1 Introduction

Quantum computers [10] promise to have enormous computational power and, thus, to solve relevant problems significantly faster than their classical counterparts. In recent years, large efforts have been put on their development, but while their mathematical foundations have been widely explored and are mostly quite well understood, the physical realization currently provides the biggest obstacle preventing the widespread use of quantum computers.

While more and more powerful quantum computer architectures have been presented with increasing quantity and quality of the so-called qubits, the basic computational entities in quantum computing, one of the physical constraints that all these architectures have in common is the limited availability of quantum operations/gates. Typically, multi-qubit gates are much harder to realize than

single-qubit gates and in many cases there is only one multi-qubit gate natively available, namely the two-qubit controlled-NOT (CNOT) gate. As there are several universal gate libraries consisting of the CNOT gate and single-qubit gates only, e.g. the Clifford+T library [6], this still allows to perform arbitrary quantum computations. However, in various architectures, the CNOT is only available on a small subset of physically adjacent qubit pairs, which can make computations that require CNOT operations on distant qubits quite complex. Fortunately, there are ways to simulate these logical CNOTs at the physical level and transform a quantum circuit that contains non-native CNOTs to a quantum circuit containing only native gates and, thus, being ready for the execution on the targeted quantum architecture.

Many approaches to find efficient CNOT implementations have been suggested, e.g. in [3, 4, 15, 17–19]. The underlying ideas of these solutions are to use so-called SWAP gates in order to swap the qubits which the CNOT is to be applied to, with ones that a CNOT is available for in the specific architecture, or to use templates of pre-computed sequences of native gates. Since the underlying problem has been shown to be NP-complete in [5], it is not surprising that most approaches do not aim to provide minimal solutions. In fact, only [17] aims for solutions with a minimal number of SWAP and Hadamard gates, but SWAP gates themselves are not elementary gates, but need to be realized as cascades of CNOT and Hadamard gates.

In contrast, we propose an algorithm that determines an optimal implementation of arbitrary non-native CNOT gates using any combination of Hadamard gates and CNOT gates that are native to the underlying architecture. To this end, we formulate the problem as an instance of the Boolean Satisfiability (SAT) problem. The algorithm makes use of the planning problem and constructs a propositional formula which, if satisfiable, provides an implementation for a specific CNOT gate. While the SAT problem itself is NP-complete and straightforward representations of quantum states, gates and circuits require an exponential number of complex-valued variables, the crucial trick here is to make use of a dedicated representation borrowed from the stabilizer circuit formalism [2] that requires only a quadratic number of Boolean variables.

Experimental evaluations of some quantum computer architectures show that the resulting problem formulation scales considerably well—despite the exponential complexity of the SAT problem. Our results indicate that SWAP-based approaches indeed do not yield such optimal solutions for many CNOT gates, as the proposed algorithm determined significantly more efficient implementations.

The remainder of this paper is structured as follows. The next section introduces notations and preliminaries needed in this paper. Section 3 discusses the considered problem and related work, followed by Section 4 presenting our approach to determining optimal implementations of non-native CNOT gates. Experimental results are presented in Section 5. Finally, the paper is concluded in Section 6.

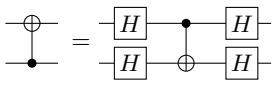

Fig. 1. Swapping control and target of a CNOT using Hadamard gates.

2 Background and Preliminaries

To keep the paper self-contained, this section briefly introduces the basics of quantum computation and the SAT problem.

2.1 Quantum States and Circuits

In contrast to classical bits which can only assume two discrete states, *qubits* can represent any combination of the classical Boolean values 0 and 1. More precisely, the state space of a qubit is a 2-dimensional Hilbert space such that all possible states can be written as $|\psi\rangle = a|0\rangle + b|1\rangle = \binom{a}{b}$ where $|0\rangle, |1\rangle$ denote the computational basis states (associated with the classical Boolean values) and $a, b \in \mathbb{C}$ are complex-valued numbers such that $|a|^2 + |b|^2 = 1$. Analogously, the state space of an n-qubit quantum system has 2^n basis states $(|0\ldots00\rangle, |0\ldots01\rangle, \ldots, |1\ldots11\rangle)$ and the state of such system can be described by a 2^n-dimensional complex-valued vector.

A quantum circuit is a model of quantum computation representing a sequence of quantum operations [10]. Each operation is a unitary transformation and is represented by a quantum gate. The operation of a quantum gate acting on n qubits is uniquely determined by a $2^n \times 2^n$ unitary matrix.

A *stabilizer circuit* is a quantum circuit consisting entirely of gates from the Clifford group which contains controlled-NOT ($CNOT$), Hadamard (H) and Phase (S) gates, represented by the following matrices:

$$CNOT = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1&1 \\ 1&-1 \end{pmatrix}, \quad S = \begin{pmatrix} 1&0 \\ 0&i \end{pmatrix}.$$

A CNOT on two qubits α and β, denoted as CNOT(α, β), performs a NOT operation on the target qubit β if, and only if, the control qubit α is in the $|1\rangle$-state.

Example 1. The left-hand side of Fig. 1 shows the circuit notation of a CNOT. Horizontal lines denote the qubits, the control qubit connection is indicated by a small, filled circle and the target qubit is illustrated by $\oplus$. As shown on the right-hand side, control and target of a CNOT can be swapped by applying Hadamard gates before and after the CNOT gate.

A *stabilizer state* is any quantum state which can be obtained by applying a stabilizer circuit to the initial state $|0\rangle^{\oplus n} = |0\ldots00\rangle$. Stabilizer circuits are not

universal, which means that they cannot conduct all quantum computations. Nonetheless, stabilizer circuits are used in quantum error-correction and many other applications (see [10, Section 10.5.1] for more information).

The advantage stabilizer circuits offer is their efficient simulation on a classical computer, according to the Gottesman-Knill theorem. As shown in [2], a stabilizer state on n qubits as described above can be represented by $n(2n+1)$ binary values, instead of 2^n complex numbers representing the vector which fully describes a quantum state. It can be visualized by a $(2n+1) \times 2n$ matrix, called *tableau*, containing the Boolean variables $x_{i,j}$, $z_{i,j}$, and r_i for all $i \in \{1, ..., 2n\}$ and $j \in \{1, ..., n\}$ (as shown in Fig. 2).

$$\left(\begin{array}{ccc|ccc|c} x_{11} & \dots & x_{1n} & z_{11} & \dots & z_{1n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \dots & x_{nn} & z_{n1} & \dots & z_{nn} & r_n \\ \hline x_{(n+1)1} & \dots & x_{(n+1)n} & z_{(n+1)1} & \dots & z_{(n+1)n} & r_{n+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(2n)1} & \dots & x_{(2n)n} & z_{(2n)1} & \dots & z_{(2n)n} & r_{2n} \end{array}\right)$$

Fig. 2. Tableau representing a stabilizer state [2]

Applications of quantum gates are conducted by updating these tableau entries in polynomial time by means of the following Boolean formulae:

- For a CNOT from control α to target β:

$$\forall i \in 1, ..., 2n : r_i := r_i \oplus x_{i\alpha} z_{i\beta} (x_{i\beta} \oplus z_{i\alpha} \oplus 1);$$
$$x_{i\beta} := x_{i\beta} \oplus x_{i\alpha};\ z_{i\alpha} := z_{i\alpha} \oplus z_{i\beta}$$

- For a Hadamard gate on qubit α:

$$\forall i \in 1, ..., 2n : r_i := r_i \oplus x_{i\alpha} z_{i\alpha};\ x_{i\alpha} := z_{i\alpha};\ z_{i\alpha} := x_{i\alpha}$$

- For a Phase gate on qubit α:

$$\forall i \in 1, ..., 2n : r_i := r_i \oplus x_{i\alpha} z_{i\alpha};\ z_{i\alpha} := z_{i\alpha} \oplus x_{i\alpha}$$

This simulation is apparently much more efficient than a matrix-vector multiplication of the state vector with the transformation matrix of the given gate.

2.2 SAT and Planning

The *Boolean Satisfiability Problem*, abbreviated SAT, addresses the following:

Given a Boolean formula ϕ over n variables, does a mapping v from the variables to the Boolean truth values $\{0, 1\}$ exist, such that $\phi(v) = 1$?

Example 2. Consider the following Boolean formula given in Conjunctive Normal Form (CNF):

$$\phi = (a \vee \neg b \vee c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee \neg c) \wedge \neg c$$

Because of the last clause $\neg c$, c must be 0 for ϕ to evaluate to 1. This means that the third clause is also 1. We are now left with the sub-formula $(a \vee \neg b) \wedge (\neg a \vee b)$, which is 1 if $a \mapsto 1$ and $b \mapsto 1$, or if $a \mapsto 0$ and $b \mapsto 0$. Thus, ϕ is satisfiable and $v = \{a \mapsto 1, b \mapsto 1, c \mapsto 0\}$ is one mapping that satisfies ϕ.

SAT is an NP-complete problem, as proven by [7] and many reasoning engines have been developed in order to solve SAT for arbitrary Boolean formulae. One application for SAT is the planning problem, which is described in [14]. An instance $\pi = <A, I, O, G>$ of the planning problem consists of the set of state variables A, the initial state I, the operators O and the goal state G. In essence, the problem is to find a sequence of operators that transform a system from an initial state to a defined goal state. It can be expressed as a propositional formula $\phi(t)$, so that ϕ is satisfiable if, and only if, there exists a sequence of actions of length t, so that the system is transformed from the initial state to its goal state. This allows us to conveniently solve the planning problem, by testing ϕ for satisfiability.

3 Considered Problem

Finding optimal implementations for all CNOT gates of a given quantum computer architecture is essential in order to improve the performance of algorithms that are run on said hardware, as every additional gate increases execution time and the probability of errors. If we define the cost of a CNOT gate to be the number t of gates which are native to the architecture that have to be applied in order to realize that gate, we can find an optimal implementation for a particular CNOT by determining the minimum value for t. For available gates like Hadamard gates and native CNOT gates, this cost will obviously be 1, while others will be far beyond that. For instance, the best-known realization of several CNOTs in the IBM QX5 architecture have a cost of more than 50 gates as determined by [3]. One reason this is an important problem is that native/physical CNOTs are only scarcely available in many quantum computer architectures.

Considering IBM's QX5 architecture, which is part of the *IBM Q* project available at [1], only 22 CNOTs are native to the architecture, as illustrated in Fig. 3. More precisely, an arrow between two qubits indicates that the CNOT gate whose control qubit is at the base of the arrow and whose target qubit is at the tip of the arrow is natively available. For instance, CNOT($Q1$, $Q2$), i.e. a CNOT with control on $Q1$ and target on $Q2$ is available on QX5, but not vice versa.

However, there are $\binom{16}{2} \cdot 2 = 240$ logical CNOTs for this 16-qubit system and $\binom{n}{2} \cdot 2$ in the general case of an n-qubit system, which need to be emulated to implement arbitrary quantum algorithms.

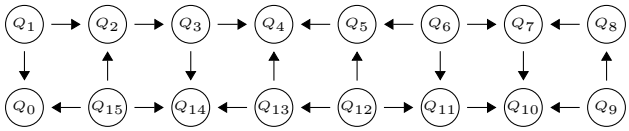

Fig. 3. IBM QX5 architecture.

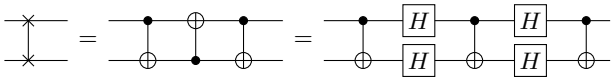

Fig. 4. SWAP gate realized by Clifford group gates.

Existing approaches for the efficient realization of non-native CNOTs have mainly focused on inserting SWAP gates, which are compositions consisting of CNOT and Hadamard gates as shown in Fig. 4. To illustrate this idea, consider the realization of a CNOT($Q3, Q0$) in IBM QX5. In order to implement this non-native CNOT, one could simply swap $Q3$ with $Q2$ and $Q2$ with $Q1$ using SWAP gates, transferring the state of $Q3$ to that of $Q1$, then apply the native CNOT($Q1, Q0$), and finally undo the SWAPs to restore the original positions of $Q1, Q2$, and $Q3$. However, each SWAP introduces an additional cost of 7 gates (c.f. Fig. 4) resulting in a total cost of $4 \cdot 7 + 1 = 29$ gates, but complete SWAPs may not be required. Almeida et al. [3] identified several movements of control and target qubits which can be realized with reduced costs—resulting in a realization of CNOT($Q3, Q0$) using only 20 gates (shown in Fig. 5).

As proven in [5], the underlying problem, like SAT, is NP-complete, which lead the authors of [17] to propose a SAT-based approach for determining the minimal number of SWAP and Hadamard gates given that the Hadamard gates are only used within SWAP gates or to invert the direction of a native CNOT. On the one hand, this limitation simplifies the problem to a purely classical-combinatorial problem and eliminates all aspects of quantum computations. On the other hand, the optimized movements in [3] suggest that it is likely to obtain further reductions if one allows for an unrestricted use of Hadamard and CNOT gates. However, this generalization significantly increases the search space which then also includes quantum circuits realizing true quantum operations and, thus, poses severe obstacles to their representation and the formulation as a SAT problem.

Luckily, CNOT and Hadamard gates do not unleash the full power of quantum computation that would require exponentially large complex-valued vectors and matrices to be dealt with, but only give rise to stabilizer circuits for which the polynomial size *tableau* representation can be employed that consists of Boolean variables only. As we will describe in the next section, this allows to use the planning problem as convenient way to express a quantum circuit as a propositional formula $\phi(t)$ to be solved for satisfiability.

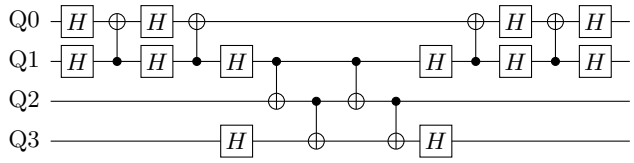

Fig. 5. Realization of CNOT($Q3, Q0$) in IBM QX5 according to [3].

4 SAT Formulation

To determine optimal implementations for non-native CNOTs, we formulate an instance $\pi = <A, I, O, G>$ of the planning problem and then convert it into a propositional formula $\phi(t)$ for a given number t, as explained in [14]. This formula shall be satisfiable if, and only if, there is a sequence of t native CNOT and Hadamard gates that realizes the desired non-native CNOT.

In the context of the considered stabilizer circuits, the initial state I of the planning problem is an arbitrary stabilizer state $|\psi\rangle$, an operator $o \in O$ represents a single quantum gate and the goal state G is the state $|\psi\rangle$ is transformed to after the application of the non-native CNOT for which we want to find an optimal implementation. The set A of state variables contains all Boolean variables that make up the tableau for a stabilizer state as reviewed in Section 2.1, namely x_{ij},z_{ij}, and r_i for $i = 1, \ldots, 2n$ and $j = 1, \ldots, n$.

With the knowledge of how π can represent a stabilizer circuit in mind, we can now construct ϕ by considering I, O and G individually.

Constructing the initial state I The initial state is simply encoded as the conjunction over A^0 in the initial state of the system: $\bigwedge_{a \in A^0} a$. Note that the number in the superscript represents the number of operators which have been applied to the system so far, so A^0 represents the initial state, where no operators have been applied yet, while A^n represents the state after exactly n operators have been applied. For our purposes, the standard initial tableau as defined in [2] is used as the initial state, where $x^0_{ij} = 1$ if $i = j$ and 0 otherwise, $z^0_{ij} = 1$ if $i - n = j$ and 0 otherwise and all $r^0_{ij} = 0$ (c.f. Figure 6a for the case $n = 2$).

$$\left(\begin{array}{cc|cc|c} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right) \qquad \left(\begin{array}{cc|cc|c} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right)$$

(a) Initial tableau (b) Goal state for CNOT(1,2)

Fig. 6. Tableaus for initial and goal state for $n = 2$ [2].

Constructing the operators O The operators are identified by the effect they have on a given state A to transform it into the successor state A'. These operators are the gates which are natively available in the specific architecture to be used, so while Hadamards may be used on all qubits, only some CNOTs are available. According to [14], any operator o is represented as follows:

$$\tau_o = p \wedge \bigwedge_{a \in A} [(EPC_a(o) \vee (a \wedge \neg EPC_{\neg a}(o))) \leftrightarrow a']$$

where p represents the operator's precondition, which needs to be 1 in order for the operator to be applicable.

For our purposes, the only condition is that a single quantum gate is to be applied by each action. In order to eliminate the possibility of two operators being applied at the same time, let the precondition p of an operator o be $\sigma_o \wedge \bigwedge_{q \in \{O \setminus \{o\}\}} \neg \sigma_q$, where the variable σ_o can later be used to identify which operators have been used, if ϕ is satisfiable.
$EPC_a(o)$ is the effect precondition of o on the state variable a. It corresponds to the formula which, if it evaluates to 1, sets the value of the literal a to 1. Since in our case the operators set a given state variable a to the binary value they evaluate to, we can prove that $\neg EPC_{\neg a}(o) = EPC_a(o)$. To this end, consider the effect the Hadamard gate on a qubit α has on the state variable r_i: $\forall i \in \{1, ..., 2n\} : r_i := r_i \oplus x_{i\alpha} z_{i\alpha}$.

This means that for an arbitrary but fixed i, we have $EPC_{r_i}(o) = r_i \oplus x_{i\alpha} z_{i\alpha}$ and $EPC_{\neg r_i}(o) = \neg(r_i \oplus x_{i\alpha} z_{i\alpha})$. We can deduce that:

$$\begin{aligned} \neg(EPC_{\neg r_i}(o)) &= \neg(\neg(r_i \oplus x_{i\alpha} z_{i\alpha})) \\ &= r_i \oplus x_{i\alpha} z_{i\alpha} \\ &= EPC_{r_i}(o) \end{aligned}$$

This obviously also holds for all combinations of operators and state variables other than a Hadamard on qubit α and r_i. With this knowledge we can simplify the formula for τ_o:

$$\begin{aligned} \tau_o &= p \wedge \bigwedge_{a \in A} [(EPC_a(o) \vee (a \wedge \neg EPC_{\neg a}(o))) \leftrightarrow a'] \\ &= p \wedge \bigwedge_{a \in A} [(EPC_a(o) \vee (a \wedge EPC_a(o))) \leftrightarrow a'] \\ &= p \wedge \bigwedge_{a \in A} EPC_a(o) \leftrightarrow a' \end{aligned}$$

Note that for all state variables a which are unaffected by an operator o, $EPC_a(o)$ simply corresponds to a, making the given sub-formula for a: $a \leftrightarrow a'$.
Inserting our precondition p, an arbitrary operator o may be encoded as:

$$\tau_o = (\sigma_o \wedge \bigwedge_{q \in \{O \setminus \{o\}\}} \neg \sigma_q) \wedge (\bigwedge_{a \in A} EPC_a(o) \leftrightarrow a')$$

For any given step, the formula for choosing an operator to apply to a state A is represented as:

$$T(A, A') = \bigvee_{o \in O} \tau_o$$

Constructing the goal state G The goal state is again encoded as the conjunction over A^t: $\bigwedge_{a \in A^t} a$. This is the tableau state which is created by applying the CNOT to be implemented to the standard initial tableau. Since most entries are 0 in the initial tableau, this reduces to updating

$$x^t_{\alpha\beta} := x^0_{\alpha\beta} \oplus x^0_{\alpha\alpha} = 0 \oplus 1 = 1 \text{ and}$$

$$z^t_{(n+\beta)\alpha} := z^0_{(n+\beta)\alpha} \oplus z^0_{(n+\beta)\beta} = 0 \oplus 1 = 1$$

for a CNOT gate on control qubit α and target qubit β. Figure 6b shows the goal state tableau for the realization of a CNOT on control qubit 1 and target qubit 2 for the standard initial tableau from Fig. 6a.

With all these representations defined, the complete propositional formula is of the following form:

$$\phi(t) = A^0 \wedge \bigwedge_{i=0}^{t-1} T(A^i, A^{i+1}) \wedge A^t$$

where the operators and state variables are super-scripted with the step i they belong to. $\phi(t)$ is satisfiable if, and only if, there is an implementation for the given CNOT using t gates. The operators used are identified by the variables σ; as for each $i \in \{1, \ldots, t\}$ there is only exactly one ${\sigma_o}^i$ for which ${\sigma_o}^i = 1$, the operator used for step i can be identified unambiguously. This means that if $\phi(t)$ is satisfiable, there is a sequence of operators $o^1, ..., o^t$ which is an implementation of the CNOT using t quantum gates. In order to determine the cheapest implementation, the minimum value for t such that $\phi(t)$ is still satisfiable has to be found.

A naive approach for finding a minimum t would be to start at $t = 1$ and increment t until $\phi(t)$ is satisfiable. Alternatively, one may take previously suggested minima as an upper bound and decrement t until $\phi(t)$ is no longer satisfiable. In fact, showing that no solution exists for some t implies that no solution exists for any smaller t. This is because the Hadamard and CNOT gates are self-inverse such that two consecutive gates on the same qubit(s) cancel out and do not have an effect to the entire circuit functionality. Thus, proving that no solution exists for t steps directly implies that there is no solution for $t-2$, $t-4$, $t-6$ steps and so on. In order to also cover the remaining cases $t-1$, $t-3$, etc., we allowed an identity operator in the last time step which has no effect to the state tableau.

Table 1. Feasibility Study

			Boolector		Z3	
Control	Target	t	Result	Run-time	Result	Run-time
Q1	Q3	3	UNSAT	9.2	UNSAT	1.2
Q1	Q3	4	SAT	14.5	SAT	2.0
Q1	Q4	7	UNSAT	24.2	UNSAT	7.2
Q1	Q4	8	SAT	29.3	SAT	10.0
Q0	Q2	8	UNSAT	28.5	UNSAT	4.0
Q0	Q2	9	UNSAT	37.5	UNSAT	10.7
Q0	Q2	10	SAT	45.0	SAT	16.3
Q0	Q4	17	UNSAT	429.1	UNSAT	141.3
Q0	Q4	18	SAT	355.3	SAT	36.6
Q8	Q13	23	UNSAT	774.0	UNSAT	716.0
Q8	Q13	24	SAT	564.2	SAT	149.8

5 Experimental Results

The algorithm above has been implemented in C++. It takes the number of qubits of the considered architecture, a list of natively available CNOTs, and the CNOT to be implemented as inputs and outputs the resulting instance of the planning problem in the SMT-LIB v2 format [13] which can then be given to any compatible SMT solver. We implemented direct interfaces to Boolector 3.2.1 [12] and Z3 4.8.8 [8]. All experiments were conducted on an Intel Core i5-7200 machine with 32 GB of main memory running Linux 4.15.

5.1 Feasibility Studies

To start with, we performed some feasibility studies to check whether the constructed instances of the planning problem are solvable in reasonable run-time and verify that the obtained quantum circuits indeed realize the desired non-native CNOTs. For this purpose, we used the IBM QX5 architecture and some non-native CNOTs for which a realization with less than 30 native gates was known. The results are provided in Table 1 where the first two columns denote the control and target qubit of the considered non-native CNOT, t denotes the number of steps and the remaining columns denote the outcome (SAT or UNSAT) and run-time in CPU seconds for the two considered SMT solvers Boolector and Z3.

The numbers indicate that for smaller values of t the run-time does not much depend on the result (SAT or UNSAT). For larger values of t, both solvers require significantly more run-time to prove that no solution exists for t steps than to determine one of the possible solutions. But still, the results clearly demonstrate the power of the proposed approach since the search space contains $(22+16)^t$ different possible realizations of native gates to be ruled out in the case of UNSAT. Overall, Z3 solver performed much quicker than Boolector and was solely used for future runs. To verify the correctness of the determined stabilizer circuits, we performed equivalence checking based on QMDDs [11].

Table 2. Experimental results for IBM QX5

Qubit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	–	5	10	14	18	28	32	38	54	44	34	30	24	20	10	5
1	1	–	1	4	8	18	22	28	38	54	32	28	22	18	8	10
2	10	5	–	1	4	14	18	24	34	38	28	24	18	14	4	5
3	14	8	5	–	1	10	14	20	30	34	24	20	14	10	1	8
4	18	12	8	5	–	5	8	14	24	28	18	14	8	5	10	12
5	24	18	14	10	1	–	5	10	20	24	14	10	5	10	14	18
6	28	22	18	14	4	1	–	1	10	14	4	1	10	14	20	22
7	38	32	28	24	14	10	5	–	5	8	1	10	14	20	24	32
8	44	38	34	30	20	16	10	1	–	5	4	14	18	24	28	38
9	40	42	38	34	24	20	14	4	1	–	1	10	14	20	24	34
10	34	42	32	28	18	14	8	5	8	5	–	5	8	14	18	28
11	30	32	28	24	14	10	5	10	14	10	1	–	5	10	14	24
12	20	22	18	14	4	1	10	14	18	14	4	1	–	1	4	14
13	16	18	14	10	1	10	14	20	28	24	14	10	5	–	1	10
14	10	12	8	5	10	14	24	28	32	28	18	14	8	5	–	5
15	1	10	1	4	8	18	22	28	38	34	24	20	14	10	1	–

5.2 Non-native CNOTs on IBM Q architectures

Having confirmed the general feasibility and correctness of the proposed approach, we turned to the problem of determining optimal implementations of non-native CNOTs on IBM Q architectures.

For the 16-qubit QX5 architecture, we took the results from [3] as the starting point and iteratively decremented the number of steps until the solver returned UNSAT for some t which, as discussed at the end of Section 4, implies that there is no realization with $k \leq t$ gates.

Tables 2 and 3 show the results for all CNOTs in QX5. In both tables, the rows denote the control qubit and the columns denote the target qubit of the CNOT. Each entry in Table 2 represents the cost of the implementation as defined earlier, i.e., the total number of native gates required in order to realize the CNOT, while Table 3 shows the absolute improvement over the best-known constructions from [3]. Note that our cost metric differs from the one used in [3], which expresses the overhead introduced by the implementation, making it less by one in all cases. This difference has been accounted for in Table 3, but should also be considered when comparing Table 2 to the results from [3].

Exact minima have been determined for all CNOTs of the QX5 architecture. Overall, there are 67 CNOTs between qubits with a maximum distance of 4 for which the constructions from [3] are indeed optimal, while for 50 CNOTs our approach determined that they can be improved by at least 12 gates. For instance, Fig. 7 shows an optimal realization of CNOT$(Q3, Q0)$ requiring only 14 gates as compared to the realization from Fig. 5 using 20 gates which was discussed in Section 3.

Table 3. Improvements compared to [3]

Qubit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	–	0	1	0	2	2	8	14	8	12	12	12	6	1	1	0
1	–	–	–	0	2	2	8	14	14	8	22	14	8	2	2	1
2	1	0	–	–	0	0	6	12	12	18	18	12	6	0	0	0
3	6	0	0	–	–	1	6	12	12	12	12	12	6	1	–	0
4	8	2	0	0	–	0	0	6	6	6	6	6	0	0	1	2
5	12	8	6	1	–	–	0	1	1	0	0	1	0	1	0	6
6	18	12	6	0	0	–	–	–	1	0	0	–	1	0	6	12
7	18	12	6	0	0	1	0	–	0	0	–	1	6	4	12	12
8	24	18	12	6	6	7	1	–	–	0	0	0	6	12	18	18
9	18	24	18	12	12	6	0	0	–	–	–	1	6	12	12	12
10	12	12	12	6	6	6	0	0	0	0	–	0	0	6	6	6
11	6	12	6	0	0	1	0	1	6	1	–	–	0	1	0	0
12	6	12	6	0	0	–	1	0	6	0	0	–	–	–	0	0
13	7	8	6	1	–	1	6	4	6	0	0	1	0	–	–	1
14	1	2	0	0	1	6	6	6	12	6	6	6	0	0	–	0
15	–	1	–	0	2	2	8	14	14	12	12	12	6	1	–	–

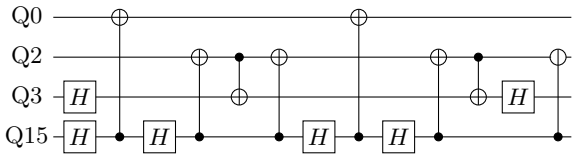

Fig. 7. Implementation of CNOT($Q3, Q0$) as calculated by our algorithm.

For the 20-qubit Q20 architecture [1], where all CNOTs are available in both directions, we were able to prove the minimality of the construction proposed in [9] using $4 \cdot (d-1)$ CNOTs where d is the distance between control and target.

5.3 Effect on Circuit Transformation

In order to evaluate the impact of the determined improvements on quantum circuit transformation, we considered a suite of benchmarks taken from [16] and the naive qubit mapping which maps i-th qubit of the circuit to qubit Qi in the QX5 architecture.

The results are shown in Table 4. Here, the first two columns describe the benchmark in terms of its name (ID) and its number of qubits (L). The next two columns denote the overhead using the original constructions from [3] and the improved/optimal CNOT implementations determined by the proposed approach. The last column lists the relative improvement which is in the range between 10% and 20%.

Table 4. Circuit Transformation for IBM QX5

Benchmark		Transformation Overhead		
ID	L	[3]	Proposed	Δ
sym6_316	14	3015	2409	-20.10 %
rd53_311	13	3174	2507	-21.01 %
hwb5_53	6	6140	5240	-14.66 %
wim_266	11	6195	5049	-18.50 %
f2_232	8	6319	5198	-17.74 %
rd53_251	8	8976	7134	-20.52 %
cm42a_207	14	9045	7619	-15.77 %
dc1_220	11	10523	8891	-15.51 %
cm152a_212	12	15228	11610	-23.76 %
sym6_145	7	19688	16058	-18.44 %
z4_268	11	23280	18549	-20.32 %
hwb6_56	7	38747	30779	-20.56 %

6 Conclusions

In this work, we proposed a method to determine optimal implementations of non-native CNOTs based on a formulation as a SAT problem. This formulation only becomes possible, since the considered gates (CNOT and Hadamard) are part of the Clifford group library for which a dedicated tableau representation can be employed that only requires $O(n^2)$ Boolean variables. While we restrict to CNOT and Hadamard gates, the approach can steadily be extended to support all Clifford group gates. As confirmed by experimental results, the resulting problem formulation scales considerably well and enabled us to determine significantly improved realizations of non-native CNOT gates for the 16-qubit IBM QX5 architecture, while for Q20 the known construction could be proven to be minimal.

References

1. IBM Q. `https://www.research.ibm.com/ibm-q/`, accessed 14-10-2020
2. Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. Physical Review A **70**(5) (Nov 2004). https://doi.org/10.1103/physreva.70.052328
3. de Almeida, A.A.A., Dueck, G.W., da Silva, A.C.R.: CNOT gate mappings to Clifford+T circuits in IBM architectures. In: Int'l Symp. on Multiple-Valued Logic. pp. 7–12. IEEE (2019). https://doi.org/10.1109/ISMVL.2019.00010
4. Ash-Saki, A., Alam, M., Ghosh, S.: QURE: Qubit re-allocation in noisy intermediate-scale quantum computers. In: Design Automation Conf. pp. 141:1–141:6. ACM, New York, NY, USA (2019)
5. Botea, A., Kishimoto, A., Marinescu, R.: On the complexity of quantum circuit compilation. In: SOCS. pp. 138–142. AAAI Press (2018)
6. Boykin, P., Mor, T., Pulver, M., Roychowdhury, V., Vatan, F.: A new universal and fault-tolerant quantum basis. Information Processing Letters **75**, 101–107 (08 2000). https://doi.org/10.1016/S0020-0190(00)00084-3

7. Cook, S.A.: The complexity of theorem proving procedures. In: Symposium on Theory of Computing. pp. 151–158 (1971)
8. de Moura, L.M., Bjørner, N.: Z3: An Efficient SMT Solver. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340 (2008)
9. Nash, B., Gheorghiu, V., Mosca, M.: Quantum circuit optimizations for nisq architectures. Quantum Science and Technology **5**(2), 025010 (Mar 2020)
10. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press (2010). https://doi.org/10.1017/CBO9780511976667
11. Niemann, P., Wille, R., Miller, D.M., Thornton, M.A., Drechsler, R.: QMDDs: Efficient quantum function representation and manipulation. IEEE Trans. on CAD **35**(1), 86–99 (2016). https://doi.org/10.1109/TCAD.2015.2459034
12. Niemetz, A., Preiner, M., Biere, A.: Boolector 2.0. J. Satisf. Boolean Model. Comput. **9**(1), 53–58 (2014). https://doi.org/10.3233/sat190101
13. Ranise, S., Tinelli, C.: The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org` (2006)
14. Rintanen, J.: Planning and SAT. Handbook of Satisfiability **185**, 483–504 (2009)
15. Siraichi, M.Y., Santos, V.F.d., Collange, S., Pereira, F.M.Q.: Qubit allocation. In: International Symposium on Code Generation and Optimization. pp. 113–125. CGO 2018, ACM, Vienna (2018). https://doi.org/10.1145/3168822
16. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: RevLib: an online resource for reversible functions and reversible circuits. In: Int'l Symp. on Multiple-Valued Logic. pp. 220–225 (2008)
17. Wille, R., Burgholzer, L., Zulehner, A.: Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In: Design Automation Conf. pp. 142:1–142:6. ACM (2019)
18. Zhou, X., Li, S., Feng, Y.: Quantum circuit transformation based on simulated annealing and heuristic search. IEEE Trans. on CAD pp. 1–1 (2020). https://doi.org/10.1109/TCAD.2020.2969647
19. Zulehner, A., Paler, A., Wille, R.: An efficient methodology for mapping quantum circuits to the IBM QX architectures. IEEE Trans. on CAD **38**(7), 1226–1236 (July 2019). https://doi.org/10.1109/TCAD.2018.2846658