Quantum Constant Propagation

Yanbin Chen[0000−0002−1123−1432]⋆ and Yannick Stade[0000−0001−5785−2528]⋆

Technical University of Munich
TUM School of Computation, Information and Technology
Boltzmannstr. 3, 85748 Garching, Germany
{chya,stya}@cit.tum.de

Abstract. A quantum circuit is often executed on the initial state where each qubit is in the zero state. Therefore, we propose to perform a symbolic execution of the circuit. Our approach simulates groups of entangled qubits exactly up to a given complexity. Here, the complexity corresponds to the number of basis states expressing the quantum state of one entanglement group. By doing that, the groups need neither be determined upfront nor be bound by the number of involved qubits. Still, we ensure that the simulation runs in polynomial time—opposed to exponential time as required for the simulation of the entire circuit. The information made available at gates is exploited to remove superfluous controls and gates. We implemented our approach in the tool quantum constant propagation (QCP) and evaluated it on the circuits in the benchmark suite MQTBench. By applying our tool, only the work that cannot be carried out efficiently on a classical computer is left for the quantum computer, hence exploiting the strengths of both worlds.

Keywords: quantum computation · constant propagation · simulation · optimization · static analysis

1 Introduction

Current status in quantum computing. Quantum computers have seen rapid improvements in recent years, especially the capability of the physical realizations of quantum computers has increased significantly [5]. There are applications where quantum computers promise an advantage over classical machines [19,8,10,23]. Currently, the provided number of qubits does not reach the order of magnitude required for putting the majority of quantum algorithms into practical use. Moreover, current hardware faces significant problems with noise that perturbs the

⋆ Both authors contributed equally to this research and are ordered alphabetically.

computed results and makes them harder to use as the circuits grow deep [15]. Consequently, the number of gates in the quantum circuit should be reduced as much as possible beforehand. For this purpose, several tools have been developed, e. g., T|ket⟩ [24], pyzx [14], Qiskit [20], staq [2], QGo [27]. More details about existing optimization techniques can be found in Section 7.

Considering initial configuration. A quantum program is usually executed starting from a state where all qubits are $|0\rangle$. Surprisingly, tools listed above take this information only slightly into account and they heavily rely on gate cancellation rules and pattern matching to simplify circuits. When it comes to the initial state, quantum circuits designers provide ad-hoc arguments why particular controls or gates can be omitted based on the initial configuration, e. g., in the context of Shor's algorithm [17,26]. Jang et al. [13] propose to use the knowledge of the initial state to automatically remove superfluous controls of controlled gates. For their optimization, they need to execute the quantum circuit on a quantum machine many times. In our view, executing a circuit on a quantum computer several thousand times to achieve an optimized version of the same circuit seems to be laborious. More in the spirit of our approach is Liu et al. [16], who propose a Relaxed Peephole Optimization (RPO) approach that leverages the information on single-qubit states which could be efficiently determined at compile time. However, their idea of treating qubits as independent systems has the drawback that information on single qubits is lost when a multi-qubit gate is applied, with few exceptions. Our approach avoids this issue by tracing entangled qubits' states up to a given complexity.

Restricted polynomial-time simulation. Since the full simulation of a quantum circuit takes exponential time in the number of qubits [9] in general, simulating the entire quantum circuit is not a viable solution for efficient optimization. For this reason, we propose a restricted simulation of a quantum circuit in Section 3, which simulates the circuit only up to a given complexity. The complexity of an entanglement group (the group of qubits that are entangled) corresponds to the number of basis quantum states. For example, the complexity of the three-qubit state, $\frac{1}{2}(|000\rangle + |001\rangle + |010\rangle + |111\rangle)$, is 4, since there are 4 basis states in the entanglement group, namely $|000\rangle$, $|001\rangle$, $|010\rangle$, and $|111\rangle$. The complexity up to which circuits are simulated is chosen beforehand and thus not depending on the number of qubits. This restriction on complexity ensures that our approach runs in polynomial time, which we will prove in Section 5. Using the idea of restricted simulation, we propose to perform a quantum equivalent to constant propagation [22], called *quantum constant propagation* (QCP). We have implemented our idea into a publicly available tool[1].

Objective of QCP. With QCP we aim to reduce the number of controls and eliminate superfluous controlled gates, following the same objective as Jang et al.. Overall QCP reduces quantum circuits in their costs to be executed on target

[1] The implementation of QCP is accessible under https://github.com/i2-tum/qcp.

platforms. Nevertheless, the circuit processed by QCP still produces the same desired outcome, as what we prove in Section 4.

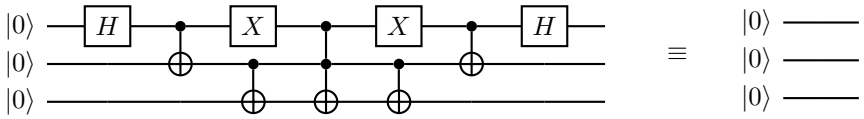


Fig. 1. Our proposed *quantum constant propagation* identifies the doubly controlled not-gate in the middle as superfluous and hence the circuit reduces to the empty circuit on the right.

Effects of QCP. Our proposed optimization technique is capable of identifying the doubly controlled not-gate in the middle of the circuit shown in Figure 1 as superfluous and, hence, the circuit reduces to the empty circuit. Our evaluation in Section 6 MQT Bench [21] demonstrates the impact of our novel optimization technique. Applying our optimization followed by Qiskit using the highest optimization level, we can remove up to 26k more gates compared to just using Qiskit [20], which corresponds to 0.5% of all gates evaluated. When we compare our approach with a similar existing optimization called Relaxed Peephole Optimization (RPO) [16] by running ours after RPO, we can remove 17.2% more gates on the evaluated circuits than just using RPO alone. It shows that this existing optimization even benefits our optimization. We believe that, especially in the future, QCP will become more important when larger circuits are built based on building blocks controlled by one or multiple controls. We comment on this in more detail in Section 8. In the next section, Section 2, we give a brief introduction to quantum computing on aspects important to this article.

2 Preliminaries

In the following, we give a brief introduction to quantum computing in order to make this article as self-contained as possible; for a more in-depth explanation, the interested reader is referred to the textbooks [12,18].

Quantum bits. Instead of bits, quantum computers operate on *qubits* (quantum bits). Those cannot just assume the two basis states $|0\rangle$ and $|1\rangle$ but also every state that can be expressed as their linear combination, $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $\alpha, \beta \in \mathbb{C}$—often called *amplitudes*—satisfy $|\alpha|^2 + |\beta|^2 = 1$. Hence, a qubit can be in a so-called *superposition* of both basis states. Upon measured, it collapses into either $|0\rangle$ or $|1\rangle$ with probability $|\alpha|^2$ or $|\beta|^2$, respectively.

Multiple quantum bits. The state of a multi-qubit quantum system is denoted by a vector in $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 = \mathbb{C}^{2^n}$. The basis vectors are written as $|b_1\rangle \otimes \cdots \otimes |b_n\rangle$ or $|b_1 \dots b_n\rangle$ for short, with $b_i \in \{0, 1\}$. Sometimes the abbreviated notation $|n\rangle$ is used where $n = \sum_{i=0}^{n} b_i \cdot 2^i$.

Gates. Operations on qubits are expressed as *gates*, each of which is denoted by a unitary matrix. A quantum computer only offers a discrete set of basis (parameterized) gates that can be applied to the qubits. Similar to conditioned branches in classical programs, gates can be controlled on the state of one or multiple other qubits. Then the controlled gate is applied to the target qubit if and only if all controlling qubits are in the $|1\rangle$ state.

Entanglement. Entanglement refers to the situation in which the measurement result of one qubit depends on the rest of the quantum system. For example, the circuit in Figure 2 creates an entanglement among three qubits: The *Hadamard-gate* brings the first qubit into maximal superposition $(|0\rangle + |1\rangle)/\sqrt{2}$; then, two controlled not-gates are applied in sequence; when measuring the resulting quantum state $(|000\rangle + |111\rangle)/\sqrt{2}$, there are only two possible outcomes—"000" and "111"—and no other combination of results, e.g., "010" and "110", can be obtained even if the three qubits are not measured simultaneously.

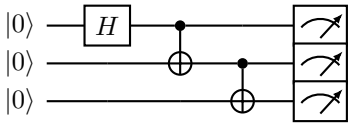


Fig. 2. This circuit creates a GHZ state of three qubits.

Curse of dimensionality. Entanglement is the root cause of why it is so hard to simulate a quantum computer on a classical machine. As long as all the qubits are separable, i. e., not entangled, one needs to store two complex numbers for each qubit. As soon as some k qubits are entangled with each other, one needs to store potentially $\mathcal{O}(2^k)$ complex numbers, which would immediately lead to exponential running time [1,11].

Concrete semantics. A quantum program consists of a sequence of quantum gates that are applied on an initial configuration of a quantum system, where usually all qubits are in the $|0\rangle$ state. The application of a gate transforms the concrete quantum state, represented as a state vector, according to the unitary matrix associated with the gate via matrix multiplication.

Interface of an optimization. In our setting, a quantum circuit is represented as a list of gates. The optimization is expected to accept a list of gates and output an optimized version. For this purpose, an optimization provides a function `transform : gate list -> gate list`. In the next section, we explain how we implement QCP utilizing a restricted simulation.

3 Methodology

As mentioned in Section 1, up to now, many quantum circuit designers have argued in complex manners about the superfluousness of specific controls and gates. Our approach aims to automate those reasonings and apply them automatically as an optimization pass to a quantum circuit. For that, we simulate the circuit and identify controls and gates that can be dropped without changing the semantics of the circuit. To make our optimization efficient in terms of a polynomial time complexity, we propose a restricted simulation that simulates the circuit only partially but satisfies the required time-bound, with the help of our specially tailored data structures that efficiently represent quantum states.

3.1 Union-Table

Efficient union-find customization. One central idea of our approach to allow polynomial running time is to keep groups of qubits that are not entangled with each other separated as long as possible. For that, we need a data structure that stores a collection of sequences, i. e., qubits, supports the operation `union`, and can retrieve the position of an element in its sequences. Additionally, it needs to maintain extra information associated with each sequence. The required functionality suggests augmenting a union-find data structure. However, we have the advantage that the total number of all elements stored in our structure is constant and known a priori, namely the number of qubits. For this reason, we use a table-like approach, hence, the name *union-table*. To store n elements in a union-table, we use an array of length n, where each field denotes one element. Each field contains a pointer to a value of type `entry` storing all indices `idxs` also pointing to this entry, their number `card`, and the value `elem` associated with this entry. This leads to the following type definitions for the union-table as an OCaml module where `t` is the type of the union-table itself.

```
module UnionTable : sig
  type 'a entry = { card: int; idxs: int list; elem: 'a }
  type 'a t =
    { size: int; perm: int array; content: 'a entry array }
  ...
```

Use of permutation. Note that we use an extra attribute `perm` that stores a permutation serving as a view onto the underlying data structure. If one calls a function that accesses an entry at index `i`, then `i` is first looked up in `perm` that returns a potentially different index `j`; the index `j` is then used for actual access to the union-table structure. More information about the functions to operate on a union-table, especially their running times, can be taken from Table 1.

3.2 Representation of a Quantum State

Bitwise representation. The union-table is polymorphic in the type used to store the extra information for each set. For that, we introduce another module

Function	Description
`make n x` `int -> 'a -> 'a t`	creates a union-table with an initial value `x` in all `n` entries; each entry corresponds to its own set, no entries are united yet into a common set, see also the corresponding paragraph in the text.
`pos_in_group i ut` `int -> 'a t -> int`	returns the position of the qubit referred to by `i` within its entanglement group; this is important to identify its state stored in the quantum state for this group.
`union i j combine ut` `int -> int -> ('a -> 'a -> bool Seq.t -> 'a)-> 'a t -> 'a t`	unites the two entries pointed to by `i` and `j`. To retrieve the new element in the new entry, the `combine` function is used. The boolean sequence passed to `combine` denotes the choices made during the merging of the two sets.
`get i ut` `int -> 'a t -> 'a`	returns the element in the entry pointed to by `i`.
`set i x ut` `int -> 'a -> 'a t -> 'a t`	updates the element in the entry pointed to by `i`.
`swap i j ut` `int -> int -> 'a t -> 'a t`	swaps two elements with each other. It alters the permutation that serves as a view of the content.
`card i ut` `int -> 'a t -> int`	returns the number of elements in the entry pointed to by `i`.
`same i j ut` `int -> int -> 'a t -> bool`	returns `true` if and only if `i` and `j` point to the same entry in the union-table.

Table 1. The table lists all functions provided by the module union-table to access and modify the stored data. The type definition of each function is given in script size below it. All functions in the lower block are implemented to run in $\mathcal{O}(1)$ time. The functions `make` and `pos_in_group` take $\mathcal{O}(n)$ time and `union` needs $\mathcal{O}(n + f(n))$ where n is the size of the union-table and f the running time of `combine`.

`QuantumState` that is a hash table with bit combinations as keys and complex numbers as values inspired by the data structure used in [7]. The bit combinations correspond to basis states, e. g., if there are stored three qubits in a group in the union-table, the binary number 11_{bin} corresponds to the quantum state $|011\rangle$ where the first of the three qubits is in the state $|0\rangle$ and the other two in $|1\rangle$. The values denote the amplitudes for each state. For this to work correctly, the length of keys must not be limited by the number of bits used for an integer, e. g., 32 or 64 bits; instead, we use arbitrary large integers as keys for the hash table. The indices of qubits in each group in the union-table are ordered; this way, one gets a mapping from the global index of a qubit to its position within the state. Figure 3 shows the representation of the state $(|10000\rangle - |10101\rangle + |11000\rangle - |11101\rangle)/2 \otimes (1 + i)/\sqrt{2}\,|0\rangle$ using a union-table and bitwise representation of the quantum states.

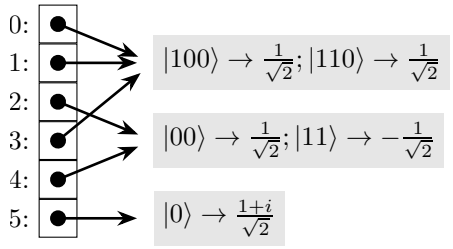


Fig. 3. The representation of a quantum system with six qubits using the union-table data structure (left), where the quantum state of each entanglement group is a hash table with the basis states as keys and their complex amplitudes as values (right). The qubits in each state are indexed from left to right. The represented quantum state is $(|10000\rangle - |10101\rangle + |11000\rangle - |11101\rangle)/2 \otimes (1+i)/\sqrt{2}\,|0\rangle$. Using the usual tensor-product notation, it is not obvious that qubits $\{0, 1, 3\}$ and $\{2, 4\}$ are separable.

Merging entanglement groups. The correct ordering of indices becomes, in particular, tricky when two entanglement groups are merged and, consequently, also their quantum states must be merged. For this purpose, the `union` function requires a `combine` function to combine the two entries, i. e., in our case the two quantum states. The union-table merges the sequences of two groups in one merge step known from the merge-sort algorithm. The combine function receives the order in which the elements from the two former groups were merged, in order to apply the same merging behavior to the quantum states. When the quantum state for n qubits contains k many basis states, the `combine` function requires $\mathcal{O}(k \cdot n)$ steps.

Application of gates. Here, we only describe the application of a single-qubit gate to a state; the approach generalizes to gates operating on multiple qubits. Let U be the matrix representation of some gate that is to be applied on qubit i in a given quantum state where $U = (u_{ij})_{i,j\in\{0,1\}}$. We iterate over the keys in the quantum state: For keys with the i-th bit equal to 0, we map the old key-value pair $(|\Psi\rangle, \alpha)$ to the two new pairs $(|\Psi\rangle, \alpha \cdot u_{11})$ and $(|\Psi'\rangle, \alpha \cdot u_{21})$, where $|\Psi'\rangle$ emerges from $|\Psi\rangle$ by flipping the i-th bit; for keys with the i-th bit equal to 1, we map the old key-value $(|\Phi\rangle, \beta)$ pair to the two new pairs $(|\Phi'\rangle, \beta \cdot u_{12})$ and $(|\Phi\rangle, \beta \cdot u_{22})$, where, again, $|\Phi'\rangle$ emerges from $|\Phi\rangle$ by flipping the i-th bit. Generated pairs with the same key (basis state) are merged by adding their values (amplitudes). For a matrix of dimension 2^d and k states in the quantum state of n qubits, this procedure takes $\mathcal{O}(2^d \cdot k \cdot n)$ steps where d is the number of affected qubits.

3.3 Restricted Simulation

Restrict complexity. The state in Figure 2 contains only two basis states as opposed to $2^3 = 8$ possible ones for which a complex number needs to be stored each. Exploiting this fact, the key features of the restricted simulation are

(i) to keep track of the quantum state as separable entanglement groups of qubits, where qubits are included in the same entanglement group if and only if they are entangled, and
(ii) to limit the number of basis states representing the quantum state of an entanglement group by a chosen constant.

Note that the number of basis states allowed in the quantum state of one entanglement group corresponds to the number of amplitudes required to be stored; all other amplitudes are assumed to be zero.

Reaching maximum complexity. The careful reader may ask how to proceed when the maximum number of allowed basis states is reached. We set the state of the entanglement group of which the limit is exceeded to $\top$, meaning that we no longer track any information about this group of qubits. By doing so, we can continue simulating the remaining entanglement groups until they may also end up in the $\top$. For this, we utilize a flat lattice that consists of either an element representing a concrete quantum state of an entanglement group, $\top$, or $\bot$ (not used) satisfying the partial order in Figure 4. The following definitions establish the relation between the concrete quantum states and their abstract description.

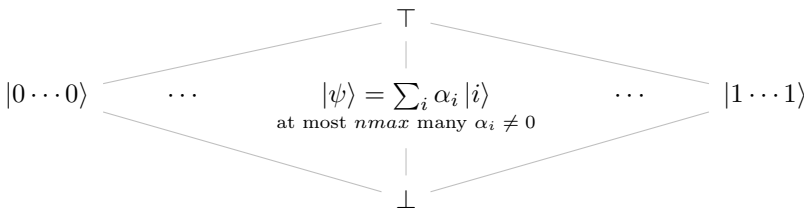


Fig. 4. Lattice for the abstract description of quantum states.

Definition 1 (Abstract state). *The abstract state s is an abstract description of a concrete quantum state if and only if $s = \top$ or $s = |\psi\rangle$ where $|\psi\rangle$ is a concrete quantum state consisting of at most nmax many non-zero amplitudes.*

Definition 2 (Abstract description relation). *Let Δ denote the description relation between quantum states and their abstract description. Furthermore, let $|\psi\rangle$ be a quantum state and s be an abstract description. The quantum state $|\psi\rangle$ is described by s, formally $|\psi\rangle\ \Delta\ s$, if and only if $s = \top$ or $s = |\psi\rangle$.*

Consequently, the entry in the union-table is not the quantum state itself but an element of the flat lattice, an abstract description of the quantum state. Definition 3 defines a concretization operator for abstract states.

Definition 3 (Concretization operator). *Let γ be the concretization operator, and s be an abstract description. Then $\gamma\ s = \{|\psi\rangle \mid |\psi\rangle\ \Delta\ s\}$.*

Next, we define the abstract effect for gates acting on quantum states.

Definition 4 (Abstract gate). *Let $[\![U]\!]^\sharp$ denote the abstract effect of the quantum gate U. For an abstract description s, the abstract effect of U is:*

$$[\![U]\!]^\sharp s = \begin{cases} U\,|\psi\rangle & \text{if} \quad s = |\psi\rangle \\ \top & \quad\;\; s = \top \end{cases}$$

Theorem 1 justifies of the above-defined abstract denotation of quantum states. It follows directly from Definition 1, Definition 2, and Definition 4.

Theorem 1 (Correctness of abstract denotation). *For any quantum state $|\psi\rangle$ and abstract description s satisfying $|\psi\rangle\ \Delta\ s$, $U\,|\psi\rangle\ \Delta\ [\![U]\!]^\sharp s$ holds.*

Operating with separated states. In the beginning, every qubit constitutes its own entanglement group. Single-qubit gates can be applied without further ado by modifying the corresponding amplitudes accordingly; the procedure behind is matrix multiplication which can be implemented in constant time given the constant size of matrices. The case where multi-qubit gates are applied is split into two subcases. When the multi-qubit gate is applied only to qubits within one entanglement group, the same argument for applying single-qubit gates still holds. Applying a multi-qubit gate across several entanglement groups will most likely entangle those; hence, we need to merge the affected groups into one.

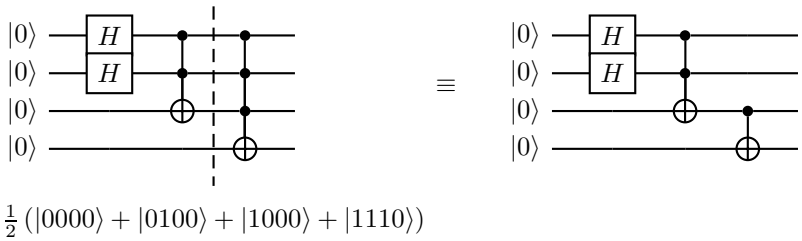
Applying multi-qubit gates. In the case of an uncontrolled multi-qubit gate, such as an echoed cross-resonance (ecr) gate, we first merge all affected entanglement groups into one. If one of those groups is already in the $\top$ state, we must set the entire newly formed entanglement group to $\top$. Otherwise, we can apply the merging strategy of the involved quantum states as described in Section 3.2. Afterward, matrix multiplication is performed to reflect the expected transformation of the state. A special case is the swap gate: we leave the entanglement groups as is and keep track of the effect of the swap gate in the permutation embedded in the quantum state structure. Before we apply a controlled gate, we perform *control reduction*—the central part of the optimization—which we outline in the next section, to remove superfluous controls.

3.4 Control Reduction

Classically determined qubits. The central task of quantum constant propagation is to remove superfluous controls from controlled gates. First, we identify and remove all classically determined qubits, i. e. those that are either in $|0\rangle$ or $|1\rangle$. If we find a qubit always in $|0\rangle$, the controlled gate can be removed. If we find qubits always in $|1\rangle$, those controls can be removed since they are always satisfied.

Satisfiable combination. By filtering out classically determined qubits as described above, a set of qubits may remain in some superposition. Even then, for the target operation to be applied, there must be a basis-state where each of the controls is satisfied, i. e., each is in $|1\rangle$. If no such combination exists, the gate can be removed entirely.

Implied qubits. When a combination with all controls in the $|1\rangle$ state was found, there can still be some superfluous controls among the remaining qubits. Consider the situation in Figure 5. Here, the upper two qubits are both in $|1\rangle$ state when the third qubit is as well; hence, the third qubit implies the first and second one. The semantics of the controlled gate remains unchanged when we remove the two upper controls. To generalize this idea, we consider every group of entangled qubits separately since there can not be any implications among different entanglement groups. Within each entanglement group, we look for implications, i. e., whether one qubit being in $|1\rangle$ state implies that other qubits are also in the $|1\rangle$ state. Those implied qubits can be removed from the list of controls.



$\frac{1}{2}(|0000\rangle + |0100\rangle + |1000\rangle + |1110\rangle)$

Fig. 5. For the rightmost gate, the third qubit implies the first and second qubit, hence the first and second control qubits can be removed from it.

Further optimization potential. In some cases, there might be an equivalence relation between two qubits; here, either one or the other qubit can be removed. This choice is made arbitrarily right now; by considering the circuit to the left or right of the gate, more optimization potential could be exploited. Moreover, the information of more than one qubit might be needed to imply another qubit. Here, we limit ourselves to the described approach because of two reasons: First, in currently common circuits [21] multi-controlled gates with more than two controls rarely occur, and for two controls, our approach finds all possible implications; second, to find the minimal set of controlling qubits is a computationally expensive task that is to the best of our knowledge exponential in the number of controls.

Handle the abstract state $\top$. If some of the entanglement groups covered by the controls are in $\top$, the optimization techniques can be applied nevertheless. Within groups that are in $\top$ no classically determined qubits or implications between qubits can be identified; however, this is still possible in all other groups. To check whether a satisfiable combination exists across all entanglement groups, we assume one within each group that is $\top$. This is a safe choice: It does not lead to any unsound optimizations since there could be a satisfiable combination in such groups.

Application of controlled gates. Before applying a controlled gate, we assume that all superfluous controls are already removed according to the approach described in Section 3.4. Like the application of an uncontrolled multi-qubit gate explained in Section 3.2, all involved entanglement groups must be merged. Then, all states that satisfy all remaining controls after the control reduction are filtered. To those, the gate is applied to the target qubits via matrix multiplication, whereas the amplitudes of all other states remain unchanged. However, if one of the controls belongs to an entanglement group in $\top$, the resulting state cannot be determined, and we set the merged entanglement group to $\top$.

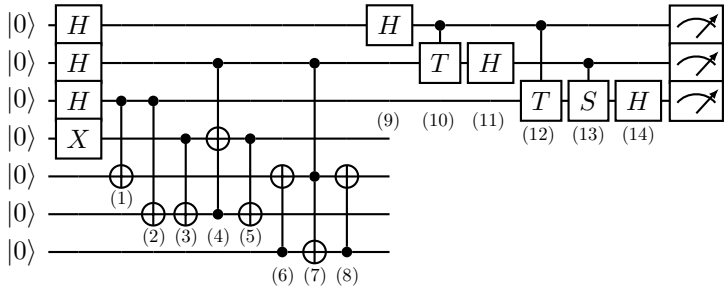


Fig. 6. Quantum constant propagation removes the control from the gate (3) and the gates (6), (10), and (12) entirely. For an explanation, see Example 1

Example 1. We will demonstrate the effect of our optimization on an example taken from [26]. Vandersypen et al. perform Shor's algorithm on 7 qubits to factor the number 15. In this process, they design the circuit from Figure 6. From the gate labeled with (3), the optimization will remove the control because the state of the controlling qubit is known to be $|1\rangle$ at this point. Gate (6) will be removed entirely because the controlling qubit is known to be in the $|0\rangle$ state. Also, gates (10) and (12) are removed since their controlling qubit will be in the $|0\rangle$ state. These optimizations seem to be trivial. However, the difficult part when automating this process is to scale it to larger and larger circuits without sacrificing efficient running time. Here, we provide the right tool for that with our proposed restricted simulation. [2] ※

4 Correctness of Control Reduction

In this section, we complement the intuitive justification for the correctness of the optimization with a rigorous proof. We first establish the required definitions

[2] Note that our optimization will not remove redundant gates such as the two Hadamard gates on top; we leave this step for other optimization tools since we already have enough to perform this task sufficiently well.

to characterize the concrete semantics of controlled operations. Similar reasoning about the correctness is contained in [13]; we see our style as more comprehensible since it argues only over the superfluousness of one qubit at a time but is still sufficient to show the correctness of the optimization.

Definition 5 (Controlled gate). *Let $U \in \mathbb{C}^{2^n \times 2^n}$ for $n \in \mathbb{N}$ be a unitary matrix of a gate. Let $C^m(U)$ denote the matrix representing the m-controlled version of this gate (the application of gate U is controlled on m qubits).*

Example 2. Consider the X-gate. The corresponding matrix is given by

$$X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right).$$

The doubly-controlled version $C^2(X)$ (the Toffoli-gate), amounts to

$$\begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 0 & 1 & \\ & & & & & 1 & 0 & \end{pmatrix} \in \mathbb{C}^{8\times 8}.$$

※

Definition 6 (Superfluousness of controls). *Given a state $|\Psi\rangle \in \mathbb{C}^{2^{m+n}}$ and a unitary $U \in \mathbb{C}^{2^n \times 2^n}$. Let $\mathbb{I} \in \mathbb{C}^{2\times 2}$ denote the identity matrix. The first one of m controls is superfluous with respect to $|\Psi\rangle$ if*

$$C^m(U)\,|\Psi\rangle = \mathbb{I} \otimes C^{m-1}(U)\,|\Psi\rangle\,. \tag{1}$$

For the following, we assume without loss of generality that the first m qubits are the controlling ones for a gate applied to the following n qubits.

Theorem 2 (Superfluousness of controls). *With the notation from Definition 6 and $|\Psi\rangle = \sum_{i=0}^{2^m-1} \sum_{j=0}^{2^n-1} \lambda_{i,j}\,|i\rangle \otimes |j\rangle$, the condition from Definition 6 is equivalent to*

$$\left.\begin{pmatrix} \lambda_{i,0} \\ \vdots \\ \lambda_{i,2^n-1} \end{pmatrix}\right|_{i=2^{m-1}-1}$$

being an eigenvector of U for the eigenvalue 1 or the 0 vector. [3]

Proof. When we write out the left-hand side of Equation (1) in Definition 6 using the definition of $|\Psi\rangle$, we get the following equation:

$$C^m(U)\,|\Psi\rangle = \sum_{j=0}^{2^n-1} \left(\sum_{k=0}^{2^n-1} u_{j,k}\lambda_{i,k}\,|i\rangle\,|j\rangle \right)\Bigg|_{i=2^m-1} + \sum_{i=0}^{2^m-2} \sum_{j=0}^{2^n-1} \lambda_{i,j}\,|i\rangle\,|j\rangle \text{ [4]} \tag{2}$$

[3] We index elements in matrices starting with 0 as opposed to the mathematical convention with 1 such that the λ corresponding to the basis state $|0\rangle$ has index 0.

[4] For simplicity we omitted the $\otimes$ sign to multiply the states.

We do the same with the right-hand side of Equation (1), which results in:

$$C^m(U)\,|\Psi\rangle = \sum_{i\in\{2^{m-1}-1,2^m-1\}} \sum_{j=0}^{2^n-1} \left(\sum_{k=0}^{2^n-1} u_{j,k}\lambda_{i,k}\,|i\rangle\,|j\rangle \right)\Bigg|_{i=2^m-1} + \sum_{\substack{i=0 \\ i\notin\{2^{m-1}-1,2^m-1\}}}^{2^m-1} \sum_{j=0}^{2^n-1} \lambda_{i,j}\,|i\rangle\,|j\rangle \tag{3}$$

Such that Equation (1) in Definition 6 is satisfied, both, Equation (2) and 3 must be equal, which gives us:

$$\sum_{j=0}^{2^n-1} \left(\sum_{k=0}^{2^n-1} u_{j,k}\lambda_{i,k} \right) |i\rangle\,|j\rangle \stackrel{!}{=} \sum_{j=0}^{2^n-1} \lambda_{i,j}\,|i\rangle\,|j\rangle \Bigg|_{i=2^{m-1}-1}$$

By performing a summand-wise comparison, this reduces to:

$$\sum_{k=0}^{2^n-1} u_{j,k}\lambda_{i,k} = \lambda i, j \Bigg|_{i=2^{m-1}-1} \qquad \forall j \in \{0,\ldots,2^n-1\}$$

This is equivalent to $(\lambda_{i,0},\ldots,\lambda_{i,2^n-1})^\top$ with $i = 2^{m-1}-1$ being an eigenvector to the eigenvalue 1 or being the zero-vector, concluding the proof. □

From Theorem 2 we can derive a corollary that brings this result in a closer relationship with our optimization using the following definition.

Definition 7 (Implied control). *Using the notation from Definition 5, we say the first control is* implied *by the other controls if* $\lambda_{i,j} = 0$ *for* $i = 2^{m-1}-1$ *and all* $j \in \{0,\ldots,2^n-1\}$.

If one interprets the basis states as equal-length bitstrings representing variable assignments of $m+n$ truth variables, then this condition intuitively states that the implication $x_1 \wedge \cdots \wedge x_{m-1} \implies x_0$ holds.

Corollary 1 (Sufficient condition for a control to be superfluous). *If the first control is implied by the other controls, it is superfluous.*

The following main theorem shows that each of the three possible modifications, as described in Section 3.4, does not change the semantics of the circuit.

Theorem 3. *Quantum constant propagation does not change the semantics relative to the initial configuration with all qubits in the* $|0\rangle$ *state.*

Proof. Without loss of generality, we can assume that the optimization pass detects the first one of m controlling qubits as superfluous. Depending on the state of the first qubit, the optimization continues in three different ways.

(i) *The first qubit is in* $|0\rangle$: Here, different from the other two cases, not just the controlling qubit is removed from the controlled gate, rather than the entire gate is removed. Thus, we need to show

$$C^m(U)\,|\Psi\rangle = |\Psi\rangle\,.$$

Since all $\lambda_{i,j} = 0$ where $i = 2^m - 1$ the sum on the right of Equation (2) reduces to 0 and the claim follows.

(ii) *The first qubit is in* $|1\rangle$: Then the amplitude of all basis state with the first qubit in $|0\rangle$ are equal to 0, i. e., $\lambda_{i,j} = 0$ where $i = 2^{m-1} - 1$ and for all $j \in \{1, \ldots, 2^n - 1\}$. Consequently, the condition in Theorem 2 is satisfied, and the first control qubit can safely be removed.

(iii) *Otherwise*: This case can only occur if the optimization found another qubit j among the controlling ones such that the first qubit is only in $|1\rangle$ if the j-th qubit is also in $|1\rangle$. Hence, the sufficient condition from Corollary 1 is satisfied and here the first control qubit can be safely removed.

Altogether, this proves the correctness of the QCP optimization pass. □

We continue with an analysis to show that QCP runs in polynomial time.

5 Running Time Analysis

Variable definition. For the rest of this section, let m be the number of gates in the input circuit and n the number of qubits. Furthermore, let k be the maximum number of controls attached to any gate in the circuit. Each entanglement group is limited in the number of basis states by the custom constant n_{max}. The achieved asymptotic running time of our QCP is then established by the following lemmas and the main theorem of this section.

Lemma 1. *Control reduction runs in* $\mathcal{O}(k^2 \cdot n)$ *time.*

Proof. As described in Section 3.4, the control reduction procedure consists of three steps. First, scanning for classically determined qubits takes $O(n \cdot k)$ time since the state of all controlling qubits needs to be determined and the entanglement group contains at most n_{max} basis states, which is constant. The factor of n comes from retrieving the position and later the state of a specific qubit within the entanglement group which comprises $\mathcal{O}(n)$ qubits, see also Table 1.

Second, the check for a combination where every controlling qubit is in $|1\rangle$, requires splitting the controlling qubits into groups according to their entanglement groups and then checking within each such group whether a combination of all controlling qubits in $|1\rangle$ exists. There can be $\mathcal{O}(k)$ groups containing each $\mathcal{O}(k)$ qubits in the worst case. For each such group, a basis state among the at most n_{max} basis states where all contained controlling qubits are in $|1\rangle$, needs to be found. This requires retrieving the position and then the state of the individual controlling qubits, which takes $\mathcal{O}(n)$ for each of those. Together, this step runs in $\mathcal{O}(k^2 \cdot n)$.

For the third step of finding implications between qubits, we need to consider every pair of qubits in each group already calculated for the previous step. For each pair, we need to retrieve the position and state of the corresponding qubits again, which takes $\mathcal{O}(n)$ times. Since there are $\mathcal{O}(k^2)$ pairs to consider, this gives us a running time of $\mathcal{O}(k^2 \cdot n)$ for this step.

Combined, the running time of the entire control reduction is $\mathcal{O}(k^2 \cdot n)$. □

To perform the control reduction, the current quantum state needs to be tracked. The running time required for that is given by the next lemma.

Lemma 2. *The application of one gate requires $\mathcal{O}(n)$ time.*

Proof. For multi-qubit (un- and controlled) gates, first the affected entanglement groups need to be merged. With the results mentioned in Section 3, this requires $\mathcal{O}(n)$ time considering that n_{max} is constant.

For uncontrolled gates, there are only single-qubit and two-qubit gates available in current quantum programming tools, hence, we consider the size of the unitary that defines the transformation of those as constant. We first check whether the number of basis states would exceed n_{max} after the application of the gate; this can be done in $\mathcal{O}(n)$ by iterating over the basis states in the entanglement group and counting the states relevant for the matrix multiplication.

For the application of the associated unitary, one must iterate over the states in the entanglement group and add for each the corresponding states with their modified amplitudes as described in Section 3.2. Since the number of states in an entanglement group is bound by n_{max} and the unitary is constant in size, this requires $\mathcal{O}(n)$ time. Checking the state of a specific qubit in a basis state within the entanglement group comprising $\mathcal{O}(n)$ qubits requires $\mathcal{O}(n)$ time.

For the controlled case, the procedure is slightly more complicated, since the unitary transformation shall only be applied to basis states where all controlling qubits are satisfied. This can be done by filtering out the right states and then applying the same procedure as above. Hence, since there are at most n_{max} states, this does not change the overall running time. Consequently, the whole application of one gate can be performed in $\mathcal{O}(n)$ time. □

Theorem 4. *QCP runs in $\mathcal{O}(m \cdot k^2 \cdot n)$.*

Proof. Lemma 1 and Lemma 2 show together, that processing one gate takes $\mathcal{O}(k^2 \cdot n + n) = \mathcal{O}(k^2 \cdot n)$ time. With m the number of gates present in the input circuit, this gives us the claimed result. □

In particular, this shows that the entire QCP runs in polynomial time which we consider important for an efficient optimization. This is due to the restriction of the number of states in each entanglement group since this number could otherwise grow exponentially in the number of qubits, i. e., would be in $\mathcal{O}(2^n)$.

6 Evaluation

The QCP, we propose, only applies control reduction and gate cancellation because of unsatisfiable controls. This may facilitate the elimination of duplicate gates or rotation folding afterward— optimizations which we leave for existing tools capable of this task. In more detail, with the evaluation presented here, we pursue three objectives:

(i) Measure the effectiveness of QCP in terms of its ability to facilitate widely used quantum circuit optimizers.
(ii) Show that QCP extends existing optimizations that also use the idea of constant propagation, namely the Relaxed Peephole Optimization (RPO) [16].
(iii) Demonstrate the efficiency (polynomial running time) of QCP even when processing circuits of large scale.

In the following, we describe the experiments performed to validate our objectives, and afterward, we show and interpret their results. The corresponding artifact [4] provides the means to reproduce the results reported here.

6.1 Experiments

The benchmark suite. To provide realistic performance numbers for our optimization, we evaluate it on the comprehensible benchmark suite MQTBench [21]. This benchmark contains circuit representations of 28 algorithms at different abstraction levels; most are scalable in the number of qubits ranging from 2 to 129 qubits. We use the set of circuits at the target-independent level compiled with Qiskit using optimization level 1. This results in a total number of 1761 circuits of varying sizes.

Representation of numeric parameters. Due to considerations of practicability and to avoid dealing with symbolic representations of numeric parameters of gates, we convert the parameters to floats and introduce a threshold[5] of $\varepsilon = 10^{-8}$; numbers that differ by less than this threshold are treated as equal, especially numbers less than ε are treated equal to zero. Consequently, some gates in the input circuits reduce to the identity gate; we remove those from the benchmark circuit in a preprocessing step.

Test settings. For purpose (i), we evaluate the influence of QCP with different values for n_{max} on optimization passes provided by three well-established and widely accepted circuit optimizers—PyZX [14], Qiskit [20], and T|ket⟩ [24]. For that, we let those passes run on all benchmark circuits without QCP to create results for a baseline; these numbers are compared with those resulting from first processing the circuits with QCP for different n_{max} values and then applying

[5] Based on personal discussion with Johannes Zeiher from the Max Planck Institute for Quantum Optics, gate parameters can be realized with a precision of $\pi \cdot 10^{-3}$.

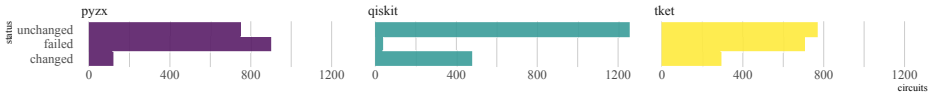


Fig. 7. This shows how many circuits remained unchanged, changed, or failed due to timeout (of one minute) or another error when first applying QCP with $n_{max} = 1024$ and then the corresponding optimization tool.

those passes. For purpose (ii), we compare the results of the optimization composed by RPO and Qiskit with those when placing QCP before or after RPO into this pipeline. The above comparisons are both conducted for two metrics, namely gate count and control count. For purpose (iii), we record the running times of QCP alone on each input circuit. All experiments are executed on a server running Ubuntu 18.04.6 LTS with two Intel® Xeon® Platinum 8260 CPU @ 2.40GHz processors offering in total 48 physical cores.

Pre-processing to fit circuit optimizer. Each circuit optimizer supports only a specific gate set. Therefore, certain pre-processing is required to adapt the circuits to the circuit optimizer. This pre-processing includes parameter formatting, gate substitution, and gate decomposition. The latter modification leads to a larger gate count than the input circuit. However, this larger gate count will already be included in our baseline for each circuit optimizer and hence, will not lead to a deterioration of the gate count through the optimization.

6.2 Results

Statistics of the benchmark suite. As mentioned in the previous section, we evaluate the QCP on 1761 circuits using between 2 and 129 qubits. The smallest circuits comprise only two gates, whereas the largest circuit contains almost 4.9 million gates. However, except for 16 circuits, the majority contain less than 50 thousand gates. The entire benchmark comprises 23.3 million gates and 22.5 million controls, of which approximately 17 thousand belong to a doubly controlled X-gate and the rest to single-controlled gates. The preprocessing of the circuits to make them suitable for the different circuit optimizers must be considered a best-effort approach. Consequently, some circuits still could not be parsed by the corresponding circuit optimizer. Figure 7 shows exemplarily how many of the 1761 circuits failed either due to a timeout of one minute or another error, remained unchanged regarding their gate count, or changed when first applying QCP for $n_{max} = 1024$ and then the corresponding optimizer.

Improvement of standard optimizers. Figure 8 shows a summary of the first experiment; the plots show how many more gates and controls, respectively, could be removed in total over the entire benchmark utilizing QCP than just using the corresponding optimizer alone. The plots for Qiskit and T|ket⟩ show that the reduction of gates and controls increases gradually with the value of

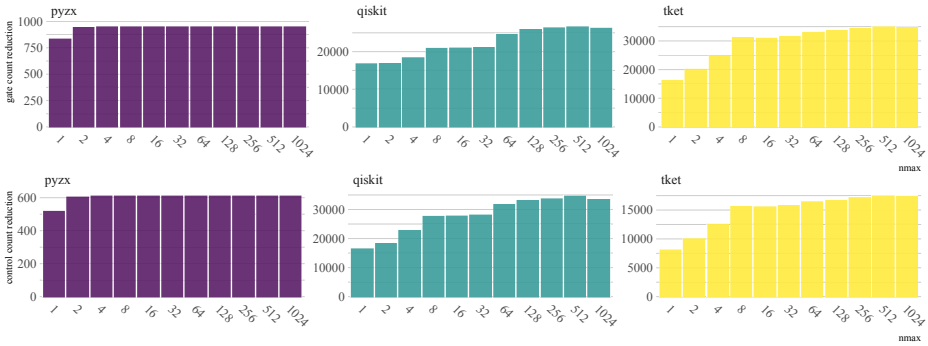


Fig. 8. This plot depicts the aggregated number of gate count (top) and control count (bottom) reduction relative to the baseline, respectively, when applying QCP with different values for n_{max} (x-axis) and then the corresponding optimizer. Note that a y-value greater than 0 corresponds to an improvement over the baseline of only performing the corresponding optimization alone (i. e., PyZX, Qiskit, or T|ket⟩).

n_{max}. Note that the absolute numbers for PyZX are smaller since PyZX fails on a lot more circuits compared to the other two optimization tools. In any case, it is evident from the plot that QCP improves the result of each optimizer.

Distributon of relative improvement. To show the impact of QCP in more detail, we calculate the relative gate reduction for each circuit by dividing the absolute gate reduction by the total gate count before optimization; analogously, we calculate the relative control reduction for every gate. Only for those circuits that fall into the category changed in Figure 7, we plot the respective distribution of the relative gate and control count reduction. Figure 9 shows the histograms when applying QCP with $n_{max} = 1024$ before each circuit optimizer. In those plots, the width of each bin amounts to 0.02. We only plot these plots for $n_{max} = 1024$ because they look almost identical for other values of n_{max}. These plots show that the impact of QCP is small on the majority of circuits. However, some circuits benefit considerably, especially when applying the optimizer T|ket⟩ afterward, which looks for patterns to replace with fewer gates; apparently, QCP modifies the circuit such that more of those patterns occur in the circuit.

Interaction with RPO. RPO [16] propagates the initial state through the circuit as long as the single qubits are in a pure state, see also Section 7. To achieve this type of state propagation in our framework, a value for n_{max} of two suffices. Still, QCP with $n_{max} = 2$ can track more information as RPO since also two basis states can suffice to express multiple qubits that are in a superposition of two basis states. Figure 10 and Figure 11 depict the mutual influence of RPO and QCP. For values 1 and 2 for the parameter n_{max}, QCP does deteriorate the results of RPO when applied before RPO. This is because RPO also implements some circuit pattern matching together with circuit synthesis; when QCP destroys such a pattern, this optimization can not be applied at this posi-

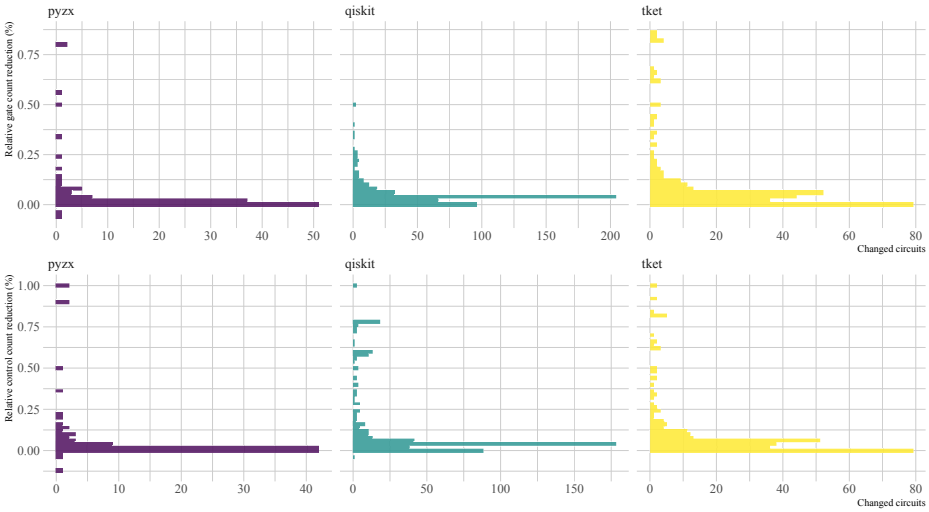


Fig. 9. The relative reduction of gates (top) and controls (bottom) of the circuits that appear in the category changed in the plot from Figure 7.

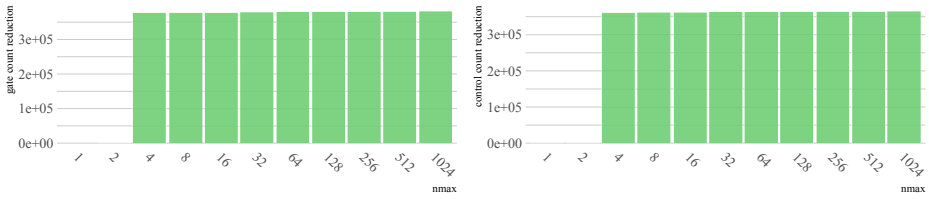


Fig. 10. Those two plots show the reduction of gates and controls, respectively, when applying QCP with different values for n_{max} (x-axis) after RPO and finally Qiskit.

tion anymore. However, for larger values for n_{max}, those plots show that QCP finds additional optimization potential and is therefore not subsumed by RPO. When looking at Figure 10, one can see that RPO even benefits QCP: In this setting, approximately 10 times more gates can be removed compared to only using QCP with Qiskit afterward. These remarkable results are mainly due to two circuit families, namely `qpeexact` and `qpeinexact`, where RPO removes some controlled gates with their technique in the first place and facilitates that QCP can remove even more controlled gates.

Analysis of QCP alone. QCP only fails on six circuits, of which one is a timeout, and five produce an error because of an unsupported gate. QCP needs the most time on the `grover` and `qwalk` circuits; on all other circuits, it finishes processing after at most 3.6 seconds. In general, the running time of QCP is high if it must track high entanglement for many gates. Accordingly, Figure 12 shows the running time of QCP on the circuits that belong to the family of Quantum

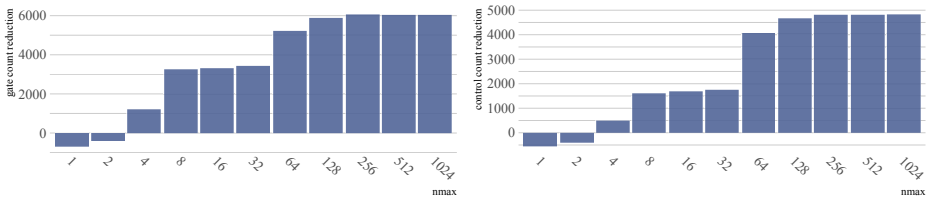


Fig. 11. Those two plots show the reduction of gates and controls, respectively, when applying RPO after QCP with different values for n_{max} (x-axis) and finally Qiskit.

Fourier Transform. Those produce maximum entanglement among the qubits where all possible basis states are represented at the end of the circuit. The plot displays the running time against the number of qubits. Note that the number of gates, and therefore the size of the circuit, grows quadratically with the number of qubits. A full simulation of those circuits would result in exponential running time. The plots indicate that QCP circumvents the exponential running time by limiting the number of basis states to express the state of an entanglement group by n_{max}.

Explanation of outliers. The plot in Figure 12 shows outliers, especially for larger values for n_{max}. Those outliers indicate an exponential running time cut-off at a specific qubit count depending on the value of n_{max}. Considering the circuits reveals that due to the generation pattern of those circuits, the chunk of gates executed on the maximal possible entanglement gradually increases in size until the qubit count where the running time drops again. For example, the maximum outlier in the plot for $n_{max} = 4096$ is reached for 112 qubits. In this circuit, 271 gates are executed on the maximum entanglement comprising 4096 basis states without increasing the entanglement before the gate that increases the entanglement above the limit is processed. In the circuit for one more qubit, i. e., 113 qubits, just 13 gates are executed on the largest possible entanglement. This is due to the order in which the gates in the input file are arranged. In summary, those practical running time measurements underpin our theoretical statements from Section 5 since the exponential growth would continue unrestrained otherwise. The results provided indicate differences from existing optimizations. In the next section, we compare our proposed optimization with those and other optimization techniques on a broader basis.

7 Related Work

Other (peephole) optimizations. Existing optimization tools [24,2,20] mostly look for known patterns consisting of several gates that can be reduced to a smaller number of gates with the same effect. A special case of those optimizations is *gate cancellation* that removes redundant gates: Many of the common gates are hermitian, i. e., they are self-inverse; when they appear twice directly after each

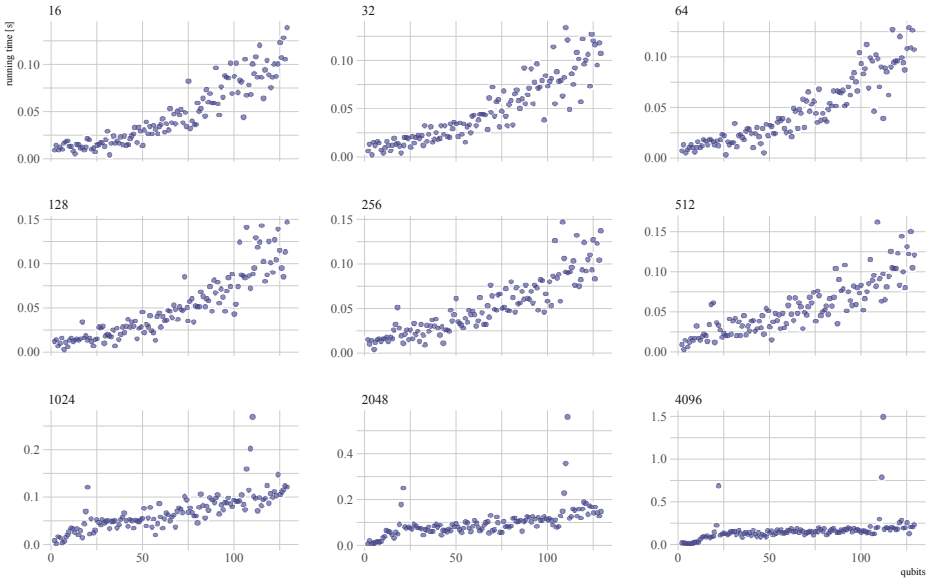


Fig. 12. This plot shows the running time of QCP for different values of n_{max} against the number of qubits (x-axis). The outliers occur due to the structure in which the circuits are generated; more details can be found in the text.

other, both can be dropped without influencing the semantics of the program. When we applied the optimization tools mentioned at the beginning of this paragraph on the circuit shown in Figure 1, none of those could reduce the circuit to the equivalent empty circuit.

Bitwise simulation. As already mentioned in Section 3.2, the idea to use a hash table to store the quantum state goes back to a simulator developed by Da Rosa et al. [7]. They use a hash table in the same way as we described in Section 3.2 with the basis states as keys and their associated amplitudes as values. However, our approach improves upon theirs by keeping qubits separated as long as they are not entangled following the idea in [3] and, hence, be able to store some quantum states even more efficiently. In contrast, Da Rosa et al. use one single hash table for the entire quantum state. Since they want to simulate the circuit and not optimize it as we aim for, they do not change to $\top$ if the computed quantum state becomes too complex. Consequently, their simulation runs still in exponential time even though it is not exponential in the number of qubits but rather in the degree of entanglement [7].

Initial-state optimizations. Circuit optimization tools developed by Liu et al. [16] and Jang et al. [13] both take advantage of the initial state. Liu et al. leverage the information on the single-qubit state which could be efficiently determined at compile time [16]. They implement state automata to keep track of the single-

qubit information on each pure state for circuit simplifications. Single-qubit information is lost though when a multi-qubit gate is applied except for a few special cases since a pure state could then turn into a mixed state. To tackle this issue, users are allowed to insert annotations from which some single-qubit information can be recovered. Our approach, however, avoids treating qubits as independent of each other and tries to trace the global state of the quantum system, enabling us not to lose all the information on qubits even after applying a multi-qubit gate on them. The circuit optimizer proposed by Jang et al. aims to remove redundant control signals from controlled gates based on the state information [13]. Instead of classical simulation, they repeatedly perform quantum measurements at truncation points to determine state information. Besides, in order to consider the noise of quantum computers, they set thresholds depending on gate errors and the number of gates and drop observations that are below the thresholds. Although their approach is lower in computational cost compared to classical simulation, the fact that quantum measurements are needed disallows their tool to run at the compile time only, since shipping circuits to the quantum runtime is necessary for performing measurements. Additionally, in their scheme, it is assumed that the controlled gate in the circuit is either a Toffoli gate or a singly-controlled unitary operation denoted to avoid computations growing exponentially, therefore gate decompositions are needed to guarantee that the assumption holds. In contrast, our approach runs statically at compile time and no prior assumption or pre-processing is required for the success of the analysis. In addition, Markov et al. [17] and Vandersypen et al. [26] optimize their circuits manually using arguments based on initial-state information.

Quantum abstract interpretation. Another point of view within the static analysis of quantum programs was established by Yu and Palsberg [28]. They introduce a way of abstract interpretation for quantum programs to verify assumptions on the final state reached after the execution of the program, hence their focus is not on the optimization of the circuit but rather to verify its correctness. Interestingly, their approach to focus on a particular set of qubits mimics our separation of entanglement groups, or to put it the other way around, our separation can be seen as one instantiation of their abstract domain just that we allow to alter the groups during simulation of the circuit instead of keeping them fixed over the entire computation as they do. Consequently, our approach dynamically adapts to the current circuit whereas Yu and Palsberg need to fix the set of qubits to focus on statically for their quantum abstract interpretation.

Classical constant propagation. When designing our optimization we were inspired by constant propagation known from classical compiler optimizations for interprocedural programs such as C/C++ programs [22]. However, our QCP differs fundamentally from classical constant propagation: In our case, we just need to pass the information along a linear list of instructions (the gates); the problem here is the sheer mass of information that needs to be tracked. In the classical case, the challenge is to deal with structural program elements such as loops and conditional branches that prevent linearly passing information about

values. Here, a constraint system consisting of equations over an abstract domain is derived from the program which then needs to be solved.

8 Conclusions

Summary. In our work, we take the idea of utilizing the most common execution condition of quantum circuits where the initial states of all qubits are in $|0\rangle$ and propose our optimization, QCP, which simulates circuits in a restricted but computationally efficient way and has demonstrated its power in one of the circuit optimization tasks, namely control reduction. In addition, QCP works in harmony with quantum computers: QCP runs in polynomial time and hence can be executed efficiently on classical computers, the output of QCP, optimized circuits, which cannot be efficiently simulated on classical computers, are submitted to quantum computers for execution. That is, we let the classical computer do all where it is good at and leave only the rest for the quantum computer. The success of QCP not only proves the value that resides within initial state information but also contributes to the research on quantum circuit optimization based on methods of static analysis running on classical computers. It is already clear that quantum circuits are expected to grow larger and larger, where building blocks containing multi-controlled gates will be heavily used. For example, OpenQASM 3.0, a highly accepted Quantum assembly language for circuit description, allows users to write arbitrarily many controls for gates [6]. Therefore, it is likely that our QCP will play to its strengths even more in the future.

Future Work. It is worthwhile to consider other abstract domains, e. g., the one used by Yu and Palsberg [28] that keep partial information about the state and still maintain the efficiency we desire. Additionally, it could be useful for QCP to consider an abstract state of meta-superposition which stores possible states after the measurement in a probability distribution. The use of meta-superposition would allow QCP to simulate circuits with intermediate measurements, i. e., measurements that happen not at the end of the circuit. We also plan to incorporate and evaluate the idea of the threshold from [13], so that QCP will be able to discard basis states that are not significant to the simulation and will be indistinguishable from noise on a real quantum computer. Besides, currently QCP is not able to detect when qubits become separable again after they were entangled. Implementing such detection facilitates keeping more state information and thus performs better optimizations. Another direction is to increase the capability of the control reduction itself: For this, we want to generalize the ideas proposed in [16] that use only pure state information of single qubits, to our setting. This includes replacing fully simulated parts of the circuit by means of circuit synthesis methods, such as KAK decomposition [25]. It is possible that performing QCP causes a loss of opportunities for other optimizations. So, one might also be interested to study how to determine the optimal order to perform different optimization passes.

Acknowledgements

We thank our reviewers for their tremendous efforts and their helpful suggestions. This has greatly improved the quality of this article. We are grateful to our supervisor Helmut Seidl for many fruitful discussions and his support at all times. Johannes Zeiher from Max Planck Institute of Quantum Optics has also provided us with precious advice. This work has been supported in part by the Bavarian state government through the project Munich Quantum Valley with funds from the Hightech Agenda Bayern Plus.

References

1. Aaronson, S., Chen, L.: Complexity-Theoretic Foundations of Quantum Supremacy Experiments (2016). https://doi.org/10.48550/ARXIV.1612.05903
2. Amy, M., Gheorghiu, V.: Staq – A full-stack quantum processing toolkit. Quantum Sci. Technol. **5**(3), 034016 (Jun 2020). https://doi.org/10.1088/2058-9565/ab9359
3. Bauer-Marquart, F., Leue, S., Schilling, C.: symQV: Automated Symbolic Verification of Quantum Programs. In: Chechik, M., Katoen, J.P., Leucker, M. (eds.) Formal Methods, vol. 14000, pp. 181–198. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-27481-7_12
4. Chen, Y., Stade, Y.: Artifact for Quantum Constant Propagation (May 2023). https://doi.org/10.5281/zenodo.8033829
5. Chow, J., Dial, O., Gambetta, J.: IBM Quantum breaks the 100-qubit processor barrier. https://research.ibm.com/blog/127-qubit-quantum-processor-eagle (Feb 2021)
6. Cross, A.W., Javadi-Abhari, A., Alexander, T., de Beaudrap, N., Bishop, L.S., Heidel, S., Ryan, C.A., Sivarajah, P., Smolin, J., Gambetta, J.M., Johnson, B.R.: OpenQASM 3: A broader and deeper quantum assembly language. ACM Transactions on Quantum Computing **3**(3), 1–50 (Sep 2022). https://doi.org/10.1145/3505636
7. Da Rosa, E.C.R., De Santiago, R.: Ket Quantum Programming. J. Emerg. Technol. Comput. Syst. **18**(1), 1–25 (Jan 2022). https://doi.org/10.1145/3474224
8. Farhi, E., Goldstone, J., Gutmann, S., Zhou, L.: The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size. Quantum **6**, 759 (Jul 2022). https://doi.org/10.22331/q-2022-07-07-759
9. Feynman, R.P.: Simulating physics with computers. Int J Theor Phys **21**(6), 467–488 (Jun 1982). https://doi.org/10.1007/BF02650179
10. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proc. Twenty-Eighth Annu. ACM Symp. Theory Comput. - STOC 96. pp. 212–219. ACM Press, Philadelphia, Pennsylvania, United States (1996). https://doi.org/10.1145/237814.237866
11. Haferkamp, J., Hangleiter, D., Bouland, A., Fefferman, B., Eisert, J., Bermejo-Vega, J.: Closing Gaps of a Quantum Advantage with Short-Time Hamiltonian Dynamics. Phys. Rev. Lett. **125**(25), 250501 (Dec 2020). https://doi.org/10.1103/PhysRevLett.125.250501
12. Hidary, J.D.: Quantum Computing: An Applied Approach. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-83274-2

13. Jang, W., Terashi, K., Saito, M., Bauer, C.W., Nachman, B., Iiyama, Y., Okubo, R., Sawada, R.: Initial-State Dependent Optimization of Controlled Gate Operations with Quantum Computer. Quantum **6**, 798 (Sep 2022). https://doi.org/10.22331/q-2022-09-08-798
14. Kissinger, A., van de Wetering, J.: PyZX: Large Scale Automated Diagrammatic Reasoning. Electron. Proc. Theor. Comput. Sci. **318**, 229–241 (May 2020). https://doi.org/10.4204/EPTCS.318.14
15. Knill, E.: Quantum Computing with Very Noisy Devices. Nature **434**(7029), 39–44 (Mar 2005). https://doi.org/10.1038/nature03350
16. Liu, J., Bello, L., Zhou, H.: Relaxed Peephole Optimization: A Novel Compiler Optimization for Quantum Circuits. In: 2021 IEEEACM Int. Symp. Code Gener. Optim. CGO. pp. 301–314. IEEE, Seoul, Korea (South) (Feb 2021). https://doi.org/10.1109/CGO51591.2021.9370310
17. Markov, I.L., Saeedi, M.: Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation (Apr 2015)
18. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, first edn. (Jun 2012). https://doi.org/10.1017/CBO9780511976667
19. Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.H., Zhou, X.Q., Love, P.J., Aspuru-Guzik, A., O'Brien, J.L.: A variational eigenvalue solver on a photonic quantum processor. Nat Commun **5**(1), 4213 (Jul 2014). https://doi.org/10.1038/ncomms5213
20. Qiskit contributors: Qiskit: An open-source framework for quantum computing (2023). https://doi.org/10.5281/zenodo.2573505
21. Quetschlich, N., Burgholzer, L., Wille, R.: MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing (Sep 2022). https://doi.org/10.48550/arXiv.2204.13719
22. Seidl, H., Wilhelm, R., Hack, S.: Compiler Design: Analysis and Transformation. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-17548-0
23. Shor, P.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proc. 35th Annu. Symp. Found. Comput. Sci. pp. 124–134. IEEE Comput. Soc. Press, Santa Fe, NM, USA (1994). https://doi.org/10.1109/SFCS.1994.365700
24. Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., Duncan, R.: T$|$ket$\rangle$: A Retargetable Compiler for NISQ Devices. Quantum Sci. Technol. **6**(1), 014003 (Jan 2021). https://doi.org/10.1088/2058-9565/ab8e92
25. Tucci, R.R.: An Introduction to Cartan's KAK Decomposition for QC Programmers (Jul 2005)
26. Vandersypen, L.M.K., Steffen, M., Breyta, G., Yannoni, C.S., Sherwood, M.H., Chuang, I.L.: Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. Nature **414**(6866), 883–887 (Dec 2001). https://doi.org/10.1038/414883a
27. Wu, X.C., Davis, M.G., Chong, F.T., Iancu, C.: QGo: Scalable Quantum Circuit Optimization Using Automated Synthesis (2020). https://doi.org/10.48550/ARXIV.2012.09835
28. Yu, N., Palsberg, J.: Quantum abstract interpretation. In: Proc. 42nd ACM SIGPLAN Int. Conf. Program. Lang. Des. Implement. pp. 542–558. ACM, Virtual Canada (Jun 2021). https://doi.org/10.1145/3453483.3454061