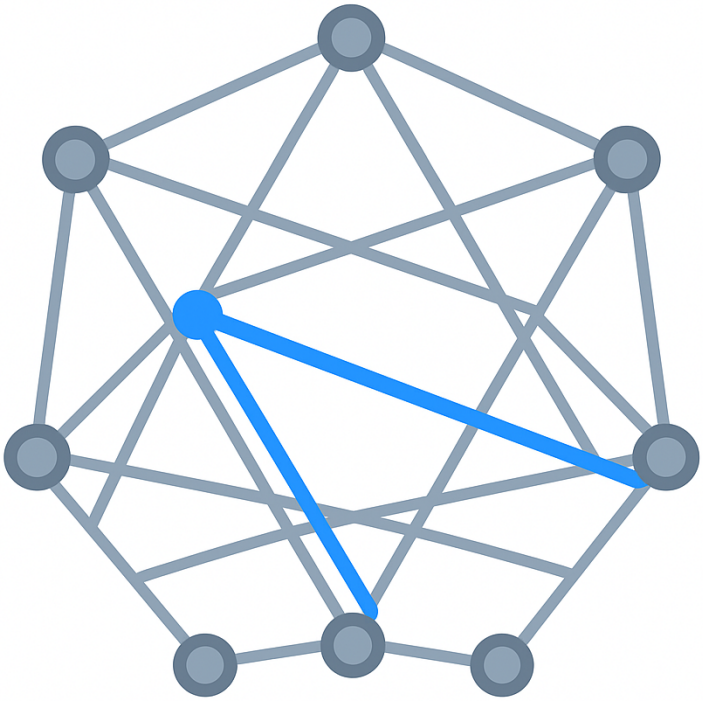
QIST-4200 Project Report

Optimizing Stabilizer Quantum Circuits via Minimal Paths on Cayley Graphs

by

Aarav Ratra
David Riekerk
Izzy van der Giessen
Jerzy Wierzbicki
Tiago Duarte Lopes da Silva



Course Instructor:	Prof. Evert van Nieuwenburg (Leiden University)
Supervisor:	Mr. Dimitrios Thanos (PhD Candidate, Leiden University)
Project duration:	September 2, 2025 – October 31, 2025
Submitted on:	October 31, 2025

TU Delft

Preface

This project represents the culmination of our collaborative efforts in the QIST-4200 course at TU Delft and Leiden University, where we explored and implemented a method for optimizing stabilizer circuits using Cayley graphs. As newcomers to the field of quantum information and computational sciences, and hailing from diverse academic backgrounds, this project has been an insightful experience for all of us.

We would like to extend our heartfelt gratitude to our supervisor, Mr. Dimitrios Thanos, for his valuable discussions and continuous feedback throughout the project, which motivated us to strive for excellence. Additionally, we thank Prof. Alfons Laarman for his detailed and constructive feedback on our work and report. We also appreciate Prof. Evert van Nieuwenburg, the course instructor for QIST-4200, and Mrs. Eva de Haan, the program coordinator for QIST, for providing us with the opportunity to work on this project. Lastly, we are grateful to ITAV TU Delft for their training sessions on teamwork and writing, which were particularly beneficial, especially since many of us are relatively new to collaborative work.

We have responsibly utilized AI-based tools for formatting and writing support, ensuring that all content reflects our own work and understanding.

Beyond striving for a good grade, we hope that this report contributes to ongoing research and development in quantum circuit optimization methods and simulation tools.

Aarav Ratra
David Riekerk
Izzy van der Giessen
Jerzy Wierzbicki
Tiago Duarte Lopes da Silva
Delft, October 2025

Summary

In this work, we study how to minimize Clifford circuits by viewing the problem as finding the shortest path on the Cayley graph of the Clifford group. In this graph, each node represents a Clifford operator, and each edge corresponds to applying one of the generators,

$$S_n = \{H_i, S_i, \mathrm{CX}_{ij}\}.$$

Since all generators are given the same weight, the Cayley graph is unweighted, and Dijkstra's algorithm can be used to find the shortest path from the identity to any target Clifford operator. This path directly gives the shortest, or optimal, Clifford circuit.

For the two-qubit case, we found that the maximum number of gates needed to build any Clifford operation is 11. This matches the known theoretical diameter of the Cayley graph. The optimized circuits also contain fewer two-qubit (CX) gates than single-qubit (H, S) gates, which reflects the internal structure of the Clifford group.

Although our current implementation assumes that all gates have the same cost, the method can easily be extended to weighted graphs, where each gate has its own cost based on factors such as noise or execution time. While the computational cost increases with the number of qubits, the approach works well for small systems and provides a useful basis for scalable Clifford circuit synthesis.

Contents

1

Introduction

Quantum computation relies heavily on efficient manipulation of quantum states through quantum circuits. *Clifford circuits* are an important subset due to their important role in quantum error correction, entanglement manipulation, and fault-tolerant computation. In addition, Clifford circuits can be efficiently simulated classically.

Optimizing Clifford circuits remains a non-trivial problem of both theoretical and practical relevance. Optimization of quantum circuits serves multiple purposes: it reduces overall gate count, minimizes circuit depth, and, most importantly, limits the propagation of quantum noise on near-term devices. Prior research has extensively investigated optimization techniques for Clifford circuits, including both local and graphical methods. There has been work done on graphical optimization of CNOT circuits [1], but this work only deals with a subgroup of the Clifford group.

In this work, we extend these ideas to a broader class of stabilizer and Clifford operations by formulating the *circuit minimization problem* as a *shortest-path search in a Cayley graph*. Each node in this graph corresponds to an element of the Clifford group, and edges represent the application of generators from a chosen gate set. Within this framework, finding an optimal circuit that realizes a given Clifford operation reduces to finding the minimal-length path between the identity element and the target element in the corresponding Cayley graph.

We restrict our analysis to the two-qubit Clifford circuits, which are already rich in non-commutative structure while remaining computationally manageable. Understanding the two-qubit case is essential,as larger circuits can be decomposed into smaller units thath can be optimized individually using these techniques. Overall, this work contributes to the growing range of techniques aimed at finding optimal quantum circuits.

The remainder of the report is structured as follows - Section 2 provides a mathematical introduction to Cayley graphs and reviews Clifford circuits along with their stabilizer tableau representations. Section 3 outlines the proposed approach for generating and storing the Cayley graph, as well as the circuit optimization algorithm. The experimental results obtained are displayed in Section 4. Finally, Section 5 discusses the results, while outlining current limitations and potential directions for future improvements.

2

Preliminaries

This chapter introduces the necessary mathematical and physical foundations for understading the optimization method presented in the report. We first provide an introduction to the Cayley graph, a geometric representation of group structures, followed by a brief discussion of the Clifford group, which defines the gate set relevant to our circuits. Finally, we introduce the tableau representation, a compact formalism that enables efficient classical simulation and computation of Clifford operations. We combine these to map the problem of Clifford circuit optimization to a shortest path finding problem for graphs.

2.1. Cayley Graph - Mathematical Introduction

Before introducing the Cayley Graph, we cover a few definitions regarding group theory.

A group is a pair $(G, \cdot)$, where G is a set and $\cdot : G \times G \to G$ is a (multiplication) map. The pair satisfies the following axioms:

- Associativity: $\forall g, h, i \in G : g \cdot (h \cdot i) = (g \cdot h) \cdot i$
- Neutral element: $\exists e \in G$ s.t. $\forall g \in G : e \cdot g = g \cdot e = g$
- Inverse: $\forall g \in G \; \exists f \in G$ s.t. $f \cdot g = g \cdot f = e$

It is said that a subset $S \subseteq G$ generates G if, for each member g of G, there exists a integer k and $a_1, \ldots, a_k \in S$ such that

$$g = a_1 \circ \ldots \circ a_k.$$

We write this relation as $\langle S \rangle = G$, to indicate that S generates G. The product $a_1, \ldots, a_k$ is referred to as a word in the generators of S, and the number k is called its word length.

A Cayley graph is a graphical method of representing the structure of a group for a given generating set. Formally speaking, for a group $G = \langle S \rangle$ for some generating set S, The Cayley Graph, denoted by $\mathcal{G}(G, S)$ is defined as:

- Nodes: Each $g \in G$ corresponds to a node/vertex.
- Edges: For each $g \in G$ and $s \in S$, there is a directed edge from node g to node gs. Hence, edges represent multiplication by generators.

An example Cayley graph [2] is depicted in Figure 2.1, which represents a subgroup of two-qubit quantum operators generated by set $\{H_1, S_2\}$.

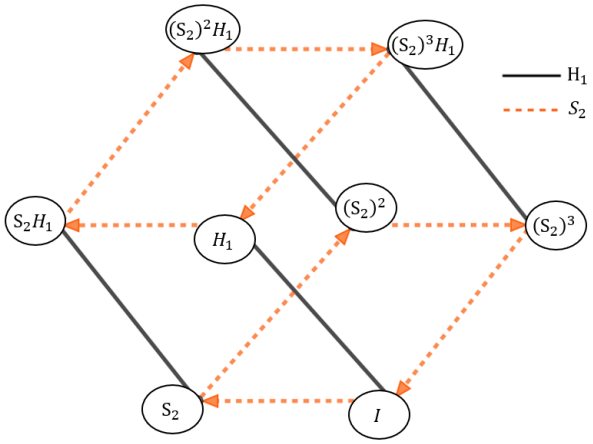


Figure 2.1: Cayley graph of $G = \langle H_1, S_2 \rangle$ (figure taken from [2] and adapted to include representatives as nodes)

By construction, for $g = a_1, \ldots a_k$ each word $a_1, \ldots a_k$ for some $k \in \mathbb{N}$ corresponds one-to-one to a path from the identity element e to g on its Cayley graph $\mathcal{G}$ - and hence, the word length k corresponds to the path length followed. Hence, the smallest number k for which g can be expressed as such a product is equivalent to the shortest path distance from e to g, denoted by $d(e, g)$. In other words - it is the length of the shortest word.

The maximum of these distances, taken over all elements of the group, defines a key graph-theoretic property — the diameter of the Cayley graph.

$$\mathrm{diam}(\mathcal{G}(G, S)) = \max_{g \in G} d(e, g),$$

Intuitively, the diameter measures the largest minimal distance required to reach any element of the group from the identity using the available generators - and hence provides a measure on how efficiently the group can be navigated.

2.2. Clifford Operations

The Quantum Circuit Model is one of the most widely used frameworks for describing quantum computation. In this model, a computation consists of a sequence of initialized qubits, unitary quantum gate operations and quantum measurements, which together implement the desired transformation or algorithm.

Within the set of quantum gates, Clifford operators are of particular relevance. The Clifford group of operators [3] is a group of unitary operators that normalizes the Pauli group $\mathbf{P_n}$, i.e., maps Pauli operators to other Pauli operators (up to a global phase) via conjugation [1]:

$$\mathbf{C_n} = \{U \in \mathbf{U_{2^n}} | UPU^\dagger \in \mathbf{P_n} \forall P \in \mathbf{P_n}\}$$

Operators in the n-qubit Pauli group, $\mathbf{P_n}$ are n-fold tensor products of the Pauli matrices, $\{I, X, Y, Z\}$, multiplied by a phase, $\{\pm 1, \pm i\}$, to ensure closure of the group under multiplication.

The generators of the 1-qubit Clifford group $\mathbf{C_1}$ consist of the Hadamard (H) and the phase (S) operators:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

Extending to two-qubits, the generators that form the two-qubit Clifford group $\mathbf{C_2}$ (which normalises the 2-qubit Pauli group $\mathbf{P_2}$) include the single-qubit operators H, S, and the two-qubit operator CX, which is represented as:

$$CX_{01} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

[1] The global phase is physically irrelevant and so we usually consider the Clifford group modulo global phases i.e. $\mathbf{C_n}/U(1)$.

Similarly, the CX_{10} is represented as $CX_{10} = I \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1|$ and has a different matrix representation.

Thus, the generators of the two-qubit clifford group $\mathbf{C_2}$ can be described as

$$\langle S_2 \rangle = \{H_0, H_1, S_0, S_1, CX_{01}, CX_{10}\}.$$

In general, the gate set $\{H, S, CX\}$ serves as the generators for the Clifford group, and any quantum operation in $\mathbf{C_n}$ can be represented as a composition of these gates. Therefore, these gates are often referred to as the Clifford gates.

Clifford circuits are better understood in terms of *stabilizer states*. Stabilizer states are a subset of quantum states that are invariant under Pauli operations. Stabilizer circuits, then, map stabilizer states of the Pauli group to other stabilizer states. These are of particular importance because, according to the Gottesman-Knill theorem, stabilizer circuits can be simulated efficiently (in polynomial time) on a classical computer.

A discussion on the number of elements in the Clifford group $\mathbf{C_n}$ can be controversial since different authors choose different bases to define the Clifford set. Many authors prefer to quotient the group by the global phase, and hence define the Clifford group as $\mathbf{C_n}/U(1)$ [4]. Hence, the group size can be written as:

$$N(n) = |\mathbf{C_n}/U(1)| = 2^{n^2+2n} \prod_{j=1}^{n} (2^{2j} - 1)$$

This equates to 24, 11520, and 92897280 elements for $n = 1, 2, 3$, respectively.

The Cayley graph $\mathcal{G}(\mathbf{C_n}, s)$ is of interest to us, as it can map various quantum circuits representing an operation $g \in \mathbf{C_2}$ to different paths from the identity element e.

For $n = 2$, the diameter of the Cayley graph of the Clifford group $\mathbf{C_2}$ is 11 [2]. Therefore, we can propose that any two-qubit Clifford operation can be expressed as a composition, up to a global phase, of, at most, 11 Clifford gates.

2.3. Tableau Representation of Clifford Operators

A part of the reason why Clifford circuits have efficient classical simulation stems from the fact that, rather than representing these operations as $2^n \times 2^n$ unitary matrices, we may represent them by their action on the Pauli group $\mathbf{P_n}$ modulo global phase, using elements $P \in \mathbf{P_n}$ as the basis. Thus, Pauli operators can be encoded in a $2n$-dimensional vector space. Therefore, we may represent the Clifford circuit as a *symplectic tableau* over $\mathbb{F}_2$ [5], which allows for polynomial-time storage and computation.

The tableau representation [5] encodes the action of a Clifford operator, U, on the standard Pauli basis, $\{X_i, Z_i\}, 1 \leq i \leq n$, for n qubits. Each row of the *tableau* corresponds to a Pauli operator, while each column states whether there is a presence or absence of a certain component in the transformed operator. There is also an additional column that keeps track of the global phase of the operator.

For an n-qubit system, the tableau representation requires $2n(2n + 1)$ bits, one per entry, which is exponentially smaller than the $2^{2n} * 64$ bits required for the full matrix representation, assuming IEEE 754 double-precision (64-bit) floating-point numbers for the entries.

Importantly, the tableau can be efficiently updated under Clifford gates using simple bitwise operations, enabling polynomial-time classical simulation of stabilizer circuits.

Table 2.1 compares the size of both matrix and tableau representations for 2 to 5 qubits.

Qubits (n)	Matrix representation (bits)	Tableau representation (bits)	Reduction factor
2	$2^4 \cdot 64 = 1{,}024$	$2 \cdot 2 \cdot (2 \cdot 2 + 1) = 20$	$51\times$
3	$2^6 \cdot 64 = 4{,}096$	$2 \cdot 3 \cdot (2 \cdot 3 + 1) = 42$	$97.5\times$
4	$2^8 \cdot 64 = 16{,}384$	$2 \cdot 4 \cdot (2 \cdot 4 + 1) = 72$	$228\times$
5	$2^{10} \cdot 64 = 65{,}536$	$2 \cdot 5 \cdot (2 \cdot 5 + 1) = 110$	$596\times$

Table 2.1: Comparison of memory requirements between full matrix and tableau representations for different numbers of qubits. Matrix size assumes double-precision (64-bit) entries.

This representation is not only of theoretical interest, since practical quantum computing frameworks, such as IBM's Qiskit library [6], use tableau-based methods to efficiently simulate Clifford gates and stabilizer circuits.

3

Methods

3.1. Implementation of the Cayley graph

Cayley graphs can be generated by recursively applying generators until all states are covered. Without any filters, this approach would generate the full unquotiented Cayley graph. Since we do not differentiate between gates that only differ by a global phase, a brute force Cayley graph generating algorithm is highly inefficient.

As explained in section 2.3, we make use of the tableau representation of Clifford gates. This representation uniquely encodes Clifford gates modulo phase. Since all our generators are Clifford gates, we can use the tableau representation for every node in our graph.

As a starting point, we take the identity gate and apply each generator in our generator set, S_2, obtaining $|S_2|$ new Clifford gates (nodes). Then, we check if any of these nodes are equivalent to existing nodes in the graph, i.e., if their tableau representations match that of an existing node. If no equivalence is found, the new gate is added as a node to the graph. This process is repeated until every node has been identified, i.e., no new nodes are added in an iteration.

The pseudo-code for the algorithm is provided below.

Algorithm 1 Build Cayley Graph

Require: Generators $G = \{(s, g)\}$, starting Clifford I, optional limit m_{max}
Ensure: Set of nodes and labeled edges representing the Cayley graph

```
start ← I
nodes ← {start}
queue ← [ start ]
while queue ≠ ∅ do
    current ← queue.pop(0)
    for all g ∈ G do
        new ← current · g
        if new ∉ nodes then
            nodes.add(new)
            queue.add(new)
        end if
        add edge (current, s, new)
    end for
    if m_max is defined and |nodes| ≥ m_max then
        break
    end if
end while
return nodes and labeled edges
```

3.2. Shortest Path using Dijkstra's Algorithm

Every group element $g \in G$ can be expressed as a finite product of generators $a_1, \ldots, a_k \in S$, corresponding to a path of length k from the identity e to g in the Cayley graph $\mathcal{G}(G, S)$. The smallest such k defines the distance between e and g.

3.2.1. Motivation

Breadth-first search (BFS) suffices when all generators have equal cost, since the Cayley graph is then unweighted. However, in many applications, such as finding optimal circuits in quantum computing or minimizing transformation cost in algebraic computations, each generator $a \in S$ carries a specific non-negative weight $w(a)$. The edge $x \to xa$ in the Cayley graph is then assigned weight $w(a)$, and the total cost of a product $a_1 \cdots a_k$ is $\sum_i w(a_i)$. Dijkstra's algorithm naturally extends to this setting to compute the minimal-cost path from e to any g, yielding an *optimal circuit* representation of g. We have chosen Dijkstra's algorithm with the idea of extending our research to finding minimal cost circuits, as they are usually preferred over minimal length circuits. In our case, however, all gates have the same cost, and therefore all edges in the Cayley graph have equal weight.

3.2.2. Algorithm

Dijkstra's algorithm maintains, for each vertex v, the best-known distance $d(v)$ from the identity e. Initially, all distances are set to infinity, $d(v) = \infty$, except for the source $d(e) = 0$. The algorithm repeatedly selects the vertex x with minimal tentative distance and relaxes all outgoing edges $x \to xa$ with cost $w(a)$:

$$d(xa) \leftarrow \min(d(xa),\, d(x) + w(a)).$$

Each relaxation tests whether a cheaper sequence of generators reaches xa. The process terminates when the target element g is extracted from the priority queue, yielding both the minimal cost $d(g)$ and a predecessor chain from e to g. When all edge weights are equal, Dijkstra's method reduces to a breadth-first search (BFS).

3.2.3. Optimal Circuit Interpretation

Each path in $\mathcal{G}$ corresponds to a product of generators. Hence, the shortest (minimum-cost) path from e to g directly encodes an optimal generator sequence, i.e., an optimal circuit implementing g. This circuit minimizes the total cost function, which is determined by the assigned generator weights. The correctness of Dijkstra's algorithm ensures that no alternative sequence of generators with non-negative costs can achieve a smaller total cost.

Complexity and Remarks

For an explicit graph, Dijkstra's algorithm runs in $O(|G| \cdot |S| + |G| \log |G|)$ using a priority queue. Dijkstra's algorithm provides a systematic method for finding optimal, cost-aware circuits on Cayley graphs.

4

Experimental Setup and Results

In this section, we dive into the experimental pipeline and the results obtained.

4.1. Test Cases and Pipeline

To test our circuit optimizer and to assess its capabilities, we need to perform, some known test cases to verify our circuit for basic problems, as well as do a benchmarking against a large random dataset of quantum circuits.

4.1.1. Random Circuits

An algorithm was created to generate random quantum circuits, given a certain list of generators, a desired number of gates and a number of qubits (limited to two for our analysis). Then, we recorded the initial and final circuits, as well as the time elapsed in various subprocesses (i.e. computation of `Clifford(qc)` and the path-finding process) and the reduction factors obtained.

Algorithm 2 Random Circuit Generation and Optimization Pipeline

1: Initialize empty DataFrame `Dat` with required columns
2: **for** GateCount in 5 to 50 **do**
3: **for** exp = 1 to 20 **do**
4: Generate random 2-qubit circuit `qc` with `GateCount` gates
5: Record statistics of `qc` (QASM, gate counts, depth, individual gate counts)
6: `(qc_opt, times) = opt_subroutine_with_time(qc)` (Optimize and record runtime for individual subprocesses)
7: Record statistics of `qc_opt`
8: Add `qc`, `qc_opt`, their statistics, and `t` to `Dat`
9: **end for**
10: **end for**
11: Export `Dat` as `random_dat.pkl` and `out.csv`

Note that the total optimization subroutine includes importing the Cayley graph from memory, computation of the Clifford tableau, evaluating the shortest path, and reconstructing the circuit. The runtime for each of these depends on different variables. Computing the Clifford tableau for a given circuit should ideally have linear time complexity $O(n)$ (where n is the gate count of the original circuit), while performing the actual optimization using the Cayley graph itself does not depend on n but on the overall graph size, as mentioned in Section 3.2.2.

4.1.2. Demo Examples

In addition to random circuits, we tested the algorithm using some known test cases for demonstration, based on the following identities:

1. $H_0 H_1 CX_{0,1} H_0 H_1 = CX_{1,0}$

2. $(CX_{10}H_0)^4 = S_1^2$

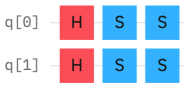
3. $CX_{10}S_0CX_{10}S_0CX_{10}S_0^2CX_{10}H_0CX_{10}S_1H_0 = \mathbb{1}$

The first example is a simple case of switching the control and target qubits for a CX gate, and the other two were taken from reference [2].

4.2. Statistical Analysis - Random Circuits

The algorithm in Section 4.1.1 was employed to effectively create a dataset of 920 Clifford circuits (with gate count varying from 5 to 50) and perform the optimization process for each. For each circuit size, ranging from 5 to 50 gates, 20 circuits were generated. An example is displayed in figure 4.1.

(a) Original Circuit (Gate Count = 22)



(b) Optimized Circuit (Gate Count = 6)

Figure 4.1: A (randomly generated) quantum circuit and an optimized version of the same

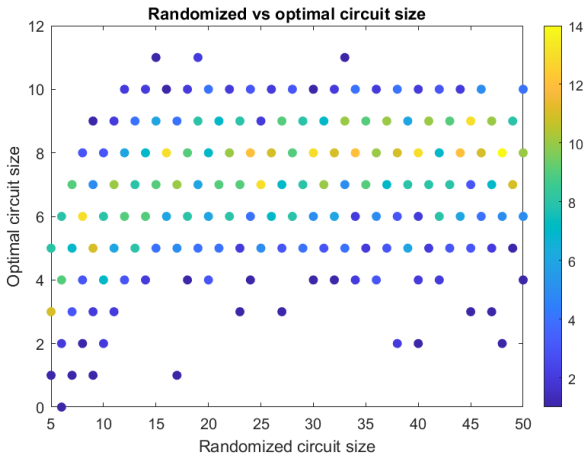


Figure 4.2: Scatterplot of optimal circuit size against initial randomized circuit size

Figure 4.2 shows a scatterplot which plots the gate-count of the optimized circuits against the gate-count of the random circuits. We observe that the number of gates does not exceed 11, which happens to be the diameter [2] of the Cayley graph of the $\mathbf{C_2}/U(1)$ group. Furthermore, the density plot suggests that the Cayley graph that, if the initial random circuit size exceeds 14, most of the optimal circuit sizes lie between 7 and 9.

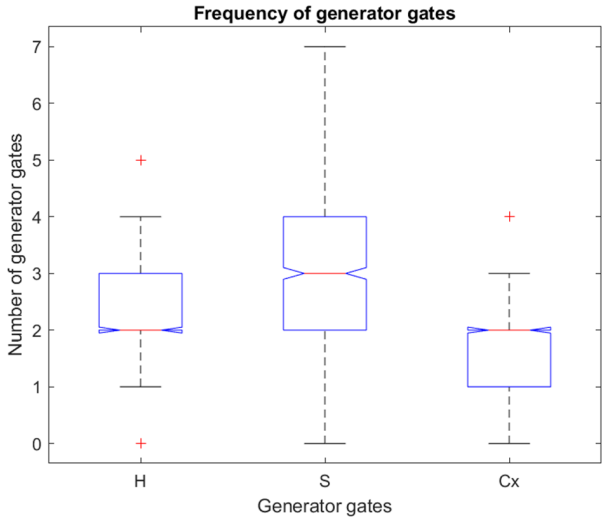


Figure 4.3: Boxplot for frequency of generator gates

Figure 4.3 suggests that the most common gates used in the optimized circuits are the S-gates, thereafter H-gates, whil the least used gates are the CX-gates.

4.2.1. Run-time analysis

The average time of a circuit optimization problem was 0.00651 ± 0.00026 (95% CI) seconds.

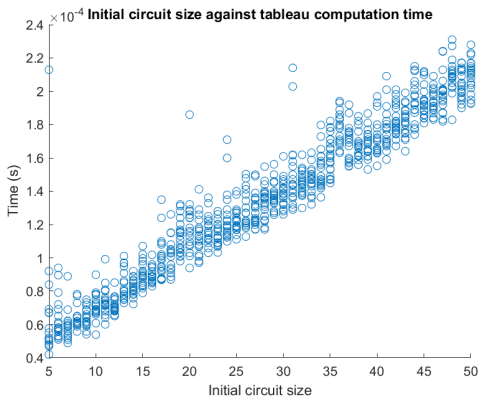


Figure 4.4: Tableau computation runtime v/s circuit size

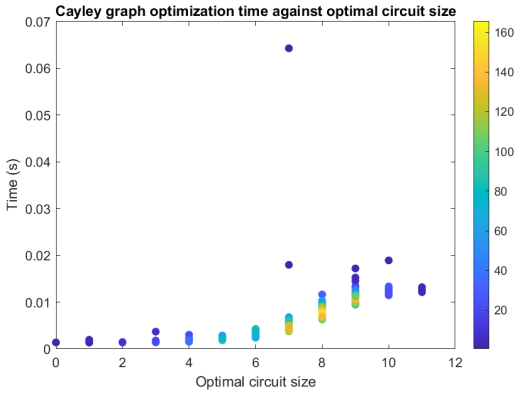


Figure 4.5: Optimization process runtime v/s final circuit size

Figure 4.4 shows a scatter plot of Tableau computation runtime against initial circuit size. A linear relationship is suggested.

Figure 4.5 shows a scatter plot optimization runtime against final word length / circuit size. We observe a non-linear positive correlation, which shows an initial exponential increase in growth but that tapers out as the Cayley graph gets exhausted.

Hence, these results show some general trends obtained over a random sample.

4.3. Demo Examples

In this section, we look through some demo examples mentioned in Section 4.1.2. The results have been displayed in Table 4.1.

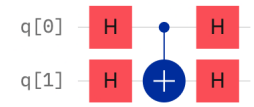
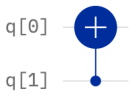
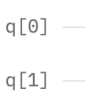
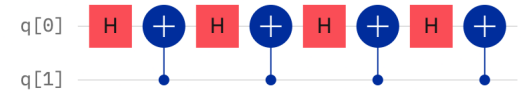
Original Circuit	Optimized Circuit
	
	
	

Table 4.1: Original and Optimized Circuits - Some known examples

As seen, the optimized circuits are in accordance with the identities mentioned in Section 4.1.2, hence validating the algorithm for circuit optimization.

5

Discussion

5.1. Interpretation of Results

We learn from section 4 that using Dijkstra's algorithm to find the shortest path on the Cayley graph for the Clifford group provides an effective method for circuit optimization, as shown in the demonstrations in Section 4.3.

5.1.1. Maximum Gate Count

Figure 4.2 illustrates that, in all tested cases, the maximum gate count for a 2-qubit circuit optimized using this method is 11. This observation is a direct empirical verification of the group-theoretic result established in [2]: namely, that the diameter of the Cayley graph for $\mathbf{C_2}/U(1)$ is 11. Hence, our optimizer guarantees that no circuit with more than 11 gates will be produced, as dictated by this fundamental group-theoretic limit.

5.1.2. Runtime Analysis

The runtime analysis of various subprocesses of our optimizer showcases the following results.

As per figure 4.4, the tableau computation process is shown to have a runtime that is linear in the number of initial gates, which matches the theoretical runtime $O(n)$.

For circuits with n going up to 50, the absolute runtime for this subprocess is in the order of $10^{-4} - 10^{-5}$ seconds (however, this depends on the machine). This, however, is significantly lower than the runtime of the optimization process, which is obtained to be of the order of 10^{-2} seconds, which is independent of n but is bounded by the size of the graph. While it is technically safe to say that the optimizer is linear in time, in our experimental settings, its runtime is not comparable to the optimization process. We can estimate that for larger circuits with greater than 10^3 gate operations, we might observe a linear trend.

An analysis of the runtime of the optimization process with respect to the final number of gates (Figure 4.5) reveals a non-linear yet increasing correlation, where it appears to show a somewhat rapid growth in the beginning, but eventually tapers in the end, resembling a logistic curve. This logistic-like correlation arises from the fact that the search space grows roughly exponentially for smaller distances, but is globally bounded by the finite group size and its diameter, resulting in eventual saturation in runtime.

Hence, we can approximately the overall time complexity as linear tableau cost plus the bounded, saturating optimization time that depends on the final circuit length:

$$T(n, n_{\text{final}}) \equiv O(n) + O\left(\frac{1}{1 + e^{-k(n_{\text{final}} - x_0)}}\right)$$

5.1.3. Gate Distribution in Optimized Circuits

Another subtle observation from the individual gate-counts of the optimization circuit (Figure 4.3) is that the CX gates are found less frequently. This is not surprising, thanks to the graphical structure of the Clifford group. The CX gate is the only two-qubit gate in the Clifford set and is therefore used exclusively

to represent non-local interactions, while most single-qubit operations can be efficiently synthesized from H and S gates. Intuitively, this intrinsic structural property of the Clifford group explains why the optimized circuits naturally exhibit reduced CX counts.

5.2. Limitations

It is worth pointing out that, using this approach, the shortest path found in the Cayley graph is not unique, i.e., there may be several paths of the same length leading from the identity element e to g. These paths may be equivalent in gate-count, but differ in other factors, such as overall depth or ancillary demand.

In practice, some decompositions are more relevant because some gates are more noise-tolerant. This could be considered by assigning a weight to each gate that quantifies the noise tolerance of the gate, which can be easily implemented using Dijkstra's algorithm. Thus, the algorithm could return the most implementable sequence of gates, rather than the shortest path, combining circuit minimization with physical implementation.

Finally, as described in Section 2.2, the Clifford group grows exponentially with circuit size. This makes it difficult to deal with circuits with more qubits as computing, storing and traversing the entire Cayley graph is no more a viable option. One possible route to go about this problem is to identify subsystems acting on at most 2 qubits within the larger circuit and optimize those individual chunks locally. While intricate and beyond the current scope of this project, this modular approach forms a critical avenue for extending scalable Clifford circuit optimization in future work.

5.3. Comparison with ZX-calculus

It is instructive to compare our algorithm with techniques that use the ZX-calculus framework. ZX-calculus [7] provides a powerful tool for analyzing and optimizing quantum circuits by representing unitary transformations (quantum gates) as tensor networks, Z-spiders, and X-spiders, which can be simplified using well-defined graphical simplification rules. This circuit optimization approach has proven particularly effective for circuits composed of the universal gate set, specifically the Clifford + T gate set. For purely Clifford circuits, however, the ZX-rewrite system does not generally yield deeper simplifications beyond known stabilizer identities.

Regarding pure Clifford circuits, the implementation approach presented has notable advantages. First, the Cayley-graph method works directly within the algebraic structure of the Clifford group, so that every transformation results indeed in a valid Clifford gate. Therefore, the optimization process is exact and deterministic: for any Clifford circuit input, it produces a circuit of minimal gate count within the Clifford group, assuming a complete generating set.

Second, since the Cayley graph has a fixed diameter (11 for 2 qubits), our approach is guaranteed to terminate in bounded time and to return a globally minimal circuit. In contrast, while ZX-calculus simplification procedures typically terminate, this is not guaranteed to be the case. Furthermore, they do not guarantee global optimality, as the rewrite system may reach different local minima depending on the sequence of rules applied.

In summary, while ZX-calculus might be a powerful tool for circuit simplification in the general, universal case, our optimizer operates directly with the algebraic structure of the Clifford group, allowing for deterministic optimization with guaranteed minimality and bounded runtime within the Clifford domain.

6

Conclusion

In this work, we proposed a solution to the Clifford circuit minimization problem by transforming it into a shortest path problem within a Cayley graph. In this graph, each node represents a Clifford operator, while each edge corresponds to a generator of the Clifford group (i.e., a Clifford gate). Therefore, any path from the identity to a desired Clifford operator represents a circuit implementation of that operator. We utilized Dijkstra's algorithm to find the shortest path, which allows for adaptations to identify the minimal cost circuit based on different cost metrics.

Our experiments show that the upper bound of a shortest Clifford circuit for two qubits is 11, in accordance with the diameter of the Cayley graph. Moreover, we showed that the shortest Clifford circuit generated by the H, S, and CX gates generally contains more single-qubit gates than 2-qubit gates, due to the structure of the Cayley graph. An overall runtime analysis showed that the tableau computation scales linearly with the initial circuit size and is negligible compared to the optimization process, with the latter being bound but exhibiting a non-linear correlation with the final gate count reflecting rapid growth in early stages, followed by saturation due to the finite size and diameter of the associated Cayley graph.

While this work offers a promising approach to cost-based circuit optimization, it does have some limitations. A significant challenge is the exponential scaling of the Cayley graph, which complicates the effectiveness of our method. However, it can still be utilized as a subroutine for larger multi-qubit circuits by breaking them down into smaller two-qubit subsystems that can be optimized locally. Tackling these larger circuits is a separate challenge that is beyond the scope of this project, but it remains a promising goal for the future.

Bibliography

[1] J. E. Christensen, S. F. Jørgensen, A. Pavlogiannis, and J. v. d. Pol, *On Exact Sizes of Minimal CNOT Circuits*, _eprint: 2503.01467, 2025. [Online]. Available: https://arxiv.org/abs/2503.01467.

[2] C. Keeler, W. Munizzi, and J. Pollack, *Clifford orbits from cayley graph quotients*, 2023. arXiv: 2306.01043 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2306.01043.

[3] D. Gottesman, *The heisenberg representation of quantum computers*, 1998. arXiv: quant-ph/9807006 [quant-ph]. [Online]. Available: https://arxiv.org/abs/quant-ph/9807006.

[4] K. Mastel, *The clifford theory of the n-qubit clifford group*, 2023. arXiv: 2307.05810 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2307.05810.

[5] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A*, vol. 70, no. 5, Nov. 2004, ISSN: 1094-1622. DOI: 10.1103/physreva.70.052328. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.70.052328.

[6] A. Javadi-Abhari, M. Treinish, K. Krsulich, *et al.*, *Quantum computing with qiskit*, 2024. arXiv: 2405.08810 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2405.08810.

[7] J. van de Wetering, *Zx-calculus for the working quantum computer scientist*, 2020. arXiv: 2012.13966 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2012.13966.

A

Appendix

A.1. Original Code and Tables

The original GitHub code and the data tables have been uploaded on GitHub: `https://github.com/jrzwrz/qcircuits-opt`

A.2. Contribution by Individual Members

A.2.1. Project Execution

We followed a SCRUM-inspired approach with weekly/bi-weekly sprints, and 1=2 meetings per week where the chairman and notetaker roles were rotated on a weekly basis. The following roles were served by each of the members:

1. **Aarav Ratra:** Conducting an investigation into the Tableau Method for Clifford Circuits, which includes developing test cases, a data-collection pipeline, and execution processes. Coordinating the presentation.
2. **David Riekerk:** Exploring the mathematical dimensions of the project, examining alternative techniques, and performing a statistical analysis of the data and results.
3. **Izzy van der Giessen:** Overseeing the implementation and storage of the Cayley Graph, integrating all components, and organizing the repository effectively.
4. **Jerzy Wierzbicki:** Implementing Dijkstra's algorithm to optimize Clifford circuits and managing the organization of the repository.
5. **Tiago Duarte Lopes da Silva:** Investigating the Tableau Method for Clifford Circuits, focusing on the creation of test cases and random circuit generation. Also served as the overall editor for the report.

A.2.2. Report Writing Division

Each member has individually worked on each of the subsections, and the division of work has been listed below:

1. **Introduction:** Jerzy Wierzbicki
2. **Preliminaries:**
   (a) **Cayley Graph - Mathematical Introduction:** David Riekerk
   (b) **Clifford Operations:** Aarav Ratra
   (c) **Tableau Representation of Clifford Operations:** Tiago Duarte Lopes da Silva
3. **Methods:**

   (a) **Implementation of the Cayley graph:** Izzy van der Giessen
   (b) **Shortest Path using Dikjstra's Algorithm:** Jerzy Wierzbicki

4. **Results:**
   (a) **Test Cases and Pipeline:** Tiago Duarte Lopes da Silva, Aarav Ratra [1]
   (b) **Statistical Analysis - Random Circuits:** David Riekerk, Aarav Ratra [2]
   (c) **Demo Examples:** Aarav Ratra
5. **Discussion:**
   (a) **Interpretation of Results:** Aarav Ratra
   (b) **Limitations:** David Riekerk
   (c) **Comparison with ZX Calculus:** Tiago Duarte Lopes da Silva
6. **Conclusion:** Izzy van der Giessen

Other than the above, Tiago Duarte Lopes da Silva served as the overall editor for the report.

A.2.3. Time Distribution

The table below presents an approximate breakdown of the time contributions made by each member throughout the project. Most of the implementation and coding work took place in September, while October saw a shift toward report writing and analysis activities. The self-study values encompass hours dedicated to material on how to write, background research, and familiarization with concepts, scaled to reflect the total project duration. Miscellaneous activities include meetings, presentation preparation, lectures, brainstorming sessions, and other project-related tasks not specifically mentioned.

Author	Report Writing (hrs)	Self-Study (hrs)	Coding/Implementation (hrs)	Misc. (hrs)
David	12	24	5	28
Aarav	16	18	12	22
Izzy	12	20	10	20
Jerzy	13	22	10	20
Tiago	15	20	6	20

A.3. AI Usage Statement

The general use of AI has been limited to clarification and assistance, though in some sections, it has been used for restructuring/rephrasing and formatting.

- The author Aarav has used AI to assist in structuring and LaTeX formatting for some parts of the report, specifically Sections 4.1 and 5.1. The AI used for this purpose is Perplexity Pro. Note that no AI-generated text has been copied and pasted into the report in these sections.
- The author Jerzy has only used AI for refining wording and grammar.
- The author Tiago has used AI to assist with error correction and rephrasing while editing the report.
- The authors David and Izzy have not used AI in the preparation of this work.

Further, the image on the cover page was created using AI software.

[1] Equal Contribution
[2] Graphical figures and overall statistical analysis were done by David, while circuit diagrams were contributed by Aarav Ratra. The written sections were equal contributions by both.