Classical Simulation of Quantum Stabilizer and Clifford Circuits

Ethan Lee, Jothi Ramaswamy

5/10/2022

One field of quantum computing that has gained traction is how we can efficiently simulate quantum circuits in classical computers. Some current research in this topic has focused on designing efficient algorithms to simulate stabilizer circuits. These circuits are unique in that they have been simulated in polynomial time without imposing restrictions on entanglement. In this paper, we summarize efficient algorithms for computing quantum circuits classically, as described by Aaronson and Gottesman for stabilizer circuits and Bravyi and Gosset for Clifford+T circuits. We also discuss how stabilizer circuits belong to the $\oplus L$ class and how this describes limitations in the power of stabilizer circuits as they are not universal, while although Clifford+T circuits are universal, simulation of them requires exponential runtime. Despite this, existing simulations of stabilizer and Clifford circuits (or modified versions of them) can still be used in applications of fault tolerance hardware and the Hidden Shift Problem.

1 Introduction

Despite the vast impact the field of quantum computing has on modern-day concepts such as cryptography, researchers are still some time away from developing an accurate quantum computer for a large number of qubits[5,7]. Thus, the concept of simulation of quantum circuits has entered the foray as a crucial topic, as it allows scientists to design and better test quantum circuits. Quantum circuits also play an incredibly important role in the BQP (class of functions solvable by a quantum circuit in polynomial time with an error probability bounded by 1/3) vs. BPP problem (class of problems solved by randomized algorithms with with an error probability bounded by 1/3). While we know BPP $\subseteq$ BQP because we can simulate any problem in BPP with a quantum circuit in polynomial time, it is still unknown whether BQP is a strict superset of BPP[1,3]. Simulating quantum circuits on classical computers can be seen as an effort to contain the power of quantum circuits, as successful and universal simulation of quantum circuits would imply that BQP cannot be seen as a strict superset of BPP. This CS 231 final project details the most recent results and algorithms regarding the simulation of quantum circuits, covering preliminary concepts, the Gottesman-Knill Theorem, improved methods for simulation of Stabilizer and Clifford circuits, applications of simulations of quantum circuits, and future directions of study in that order.

2 Preliminaries

One area of research that people have been particularly focusing on is simulating quantum stabilizer circuits on classical computers. In a general context (and as covered in CS 231), quantum codes are defined by sets of Pauli operators, or stabilizers, that they are stabilized by; that is, applying every unitary within a set of stabilizers to a quantum code should leave the quantum code unchanged. For this report, stabilizer circuits are circuits consisting only of CNOT, Hadamard, phase, and measurement gates. Current research has focused on algorithms that can simulate stabilizer circuits efficiently, and in this article we look at a particular algorithm, the Tableau algorithm, that cuts down the time of updating bits after applying a circuit from $O(n^3)$ to $O(n^2)$ time.

A lot of research has also focused on a specific type of stabilizer circuit, the Clifford circuit. Clifford circuits, also referred to as unitary stabilizers, consist only of CNOT, Hadamard, and phase gates. Section 4 discusses the T-gate, defined as $|0\rangle\langle 0| + e^{i\pi/4}|1\rangle\langle 1|$. Adding the non-Clifford T-gates onto a Clifford circuit leads to increased opportunities for addressing quantum computation problems, such as small-scale fault-tolerant quantum computers and the use of classical techniques for compiling quantum circuits; moreover, the T+Clifford gate set is universal. We also discuss magic states along with Clifford + T circuits. Magic states are, in unlimited supply, states that allow for universal quantum computation when used with Clifford operations. For example, magic states can be combined with Clifford operations to apply a T gate. However, it is well-documented that one would need an infinite supply of magic states to have a chance at complete universal computation.

3 Gottesman-Knill Theorem

Theorem 3.1. *The Gottesman-Knill Theorem uses the result*[2] *that the following statements are equivalent for any n-qubit state $|\psi\rangle$:*

- $|\psi\rangle$ *can be reached starting from $|0\rangle^{\otimes n}$ using only CNOT, Hadamard, phase, and measurement gates.*
- $|\psi\rangle$ *is stabilized by exactly 2^n Pauli operators.*
- $|\psi\rangle$ *can be uniquely identified by its stabilizer group.*

Group theory tells us that any finite group of items G has a generating set, or subset of the group that can represent all members of the group through linear combinations, of size at most $\log(|G|)$. Thus, the generating set of the stabilizer group of some n-qubit state $|\psi\rangle$ will be of size at most $\log(2^n) = n$. Furthermore, since each generator can be specified by $2n + 1$ bits, the state $|\psi\rangle$ can be represented by $n(2n + 1)$ bits. Gottesman-Knill identifies a way to update the bits specifying $|\psi\rangle$ in polynomial time every time a CNOT, Hadamard, phase, or measurement gate is applied to $|\psi\rangle$. The following section gives specific methods for performing these bit updates in an improved version of Gottesman-Knill, in which the runtime of bit updates after measurement is improved from worst-case $O(n^3)$ to $O(n^2)$.

4 Improved Simulation of Stabilizer Circuits

We can further build on the Gottesman-Knill Theorem to improve the simulation of stabilizer circuits, as Aaronson and Gottesman have done[2]. Stabilizer circuits are unique in that they can be effectively simulated using classical computers in polynomial time without any restrictions on entanglement. This is unlike other circuits, which specifically require entanglement to be polynomially bounded under a measure related to Schmidt rank in order to simulate them in polynomial time.

Prior to this research, we were still able to update each of the $n(2n+1)$ bits representing a state after each gate is applied in polynomial time. For unitary stabilizers (stabilizers without any measurement gates), we are able to update each bit in $O(n)$ time. Measurement gates are trickier and in practice take $O(n^3)$ time to update each bit, as current methods use Gaussian elimination and matrix inversion. However, this would be infeasible to simulate on a classical computer if we have, for example, 2000 qubits, given that measurements are frequent. As a result, this paper provides a more efficient *tableau algorithm* to update bits after applying measurement gates in $O(n^2)$ time.

4.1 Tableau Algorithm

4.1.1 Tableau and Rowsum

The Tableau algorithm introduces a time efficient way to update bits after applying each gate in a quantum circuit by using n stabilizers and n associated "destabilizers". These destabilizer generators are Pauli operators that together with the stabilizer generators generate the full Pauli group $\mathcal{P}_n$.

This algorithm uses a tableau that looks like the example in Appendix Section 3. The structure of the tableau is as follows:

1. Each entry either consists of binary variables x_{ij}, z_{ij}, where $i \in \{1, \cdots 2n\}$ and $j \in \{1, \cdots n\}$, or r_i (defined in step 4) for all $i \in \{1, \cdots 2n\}$.
2. Rows 1 to n of the tableau represent the destabilizer generators $R_1, ...R_n$.
3. Rows n+1 to 2n of the tableau represent the stabilizer generators $R_{n+1}, ...R_{2n}$.
4. r_i represents the phase of R_i; $r_i = 1$ if the phase is negative and 0 if the phase is positive.
5. $R_i = \pm P_1 \cdots P_n$ such that (x_{ij}, z_{ij}) determine which Pauli operator P_j is. Specifically, a combination of 00 corresponds to $P_j = I$, 01 corresponds to $P_j = X$, 11 corresponds to $P_j = Y$, 10 corresponds to $P_j = Z$.

This algorithm also uses a function rowsum(h, i), which takes the generator h and returns $i + h$. It keeps track of phase bit r_h, and all the factors of i that appear when multiplying Pauli matrices. The time complexity of rowsum is $O(n)$.

4.1.2 The Algorithm

For each gate, the algorithm updates the bits for initial state $|0\rangle^{\otimes n}$, which has $r_i = 0$ for all $i \in \{1, \dots, 2n+1\}$, and also has $x_{ij} = \delta_{ij}$ and $z_{ij} = \delta_{(i-n)j}$ for all $i \in \{1, \dots, 2n+1\}$ and $j \in \{1, \dots, n\}$ where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, as follows:

Algorithm 1 Tableau Algorithm

1. **CNOT from control a to target b** : For all $i \in \{1, \cdots 2n\}$, $r_i := r_i \oplus x_{ia} z_{ib} (x_{ib} \oplus z_{ia} \oplus 1)$. $x_{ib} := x_{ib} \oplus x_{ia}$. $z_{ia} := z_{ia} \oplus z_{ib}$.
2. **Hadamard on qubit a** : For all $i \in \{1, \cdots 2n\}$, $r_i := r_i \oplus x_{ia} z_{ia}$. Swap x_{ia} and z_{ia}.

3. **Phase on qubit a** : For all $i \in \{1, \cdots 2n\}$, $r_i := r_i \oplus x_{ia} z_{ia}$. $z_{ia} := z_{ia} \oplus x_{ia}$.

4. **Measurement of qubit a in the standard basis** : This is where it starts to get a little more complicated. First check whether there is a $p \in \{n+1, \ldots, 2n\}$ such that $x_{pa} = 1$.

   (a) If there is such a p, we should update the state as the measurement outcome is random.

      i. For all $i \in \{1, \cdots 2n\}$ where $x_{ia} = 1$ and $i \neq p$, call rowsum(i, p).

      ii. Set the $(p-n)$th row equal to the pth row.

      iii. Set the pth row to be zero except for r_p which is 0 or 1 with equal probability and z_{pa} which should be 1.

      iv. Return r_p as the final measurement.

   (b) If there is not such a p, we just have to determine whether 0 or 1 is measured as the measurement is determinate.

      i. Set the (2n + 1)th row to be identically 0.

      ii. Apply rowsum(2n + 1, i + n) for all $i \in \{1, \cdots n\}$ where $x_{ia} = 1$.

      iii. Return r_{2n+1} as the final measurement.

We can observe that the time complexity of this is largest when we are applying a measurement gate, and in this case the time complexity is $O(n^2)$ because in the worse case scenario, we run n iterations of rowsum, a linear time algorithm, along with other constant time actions. This algorithm is thus convenient because we don't have to use any Gaussian elimination. However, we do have a tradeoff in the space complexity of the algorithm. Because we also include n destabilizer generators, we double the space we need for the tableau from $n(2n+1)$ to $2n(n+1)$.

4.2 Implementation in CNOT-Hadamard-Phase

Aaronson and Gottesman also created a C program to perform the tableau algorithm, named *CNOT-Hadamard-Phase* (CHP). CHP was designed to have a high performance with a large number of qubits and measurement gates. Aaronson and Gottesman were able to practically simulate circuits with up to 3000 qubits, and experienced their biggest limitation in memory available. CHP was able to handle a maximum of 20000 qubits on a 256 MB RAM machine before needing virtual memory.

The biggest obstacle in efficiently simulating stabilizer circuits with large inputs seems to be the execution of measurement gates. As expected, when looking at the runtimes of each gate, applications of measurement gates dominated the total execution time. This makes sense because in our complexity analysis of each gate, the execution time of CNOT, Hadamard, and Phase gate updates only grew linearly in the numer of qubits in the inputs, while the execution time of measurement gate updates grew quadratically.

4.3 Limitations of simulating stabilizer circuits

Aaronson and Gottesman further their discussion of stabilizer circuits in their analysis of Gottesman-Knill's computational complexity. In particular, they provide a logarithmic-space reduction [1] from

[1]A logarithmic-space reduction is when we can transform all instances of problem A to an instance of problem B (and thus all instances of problem B since they are functionally equivalent) using extra space that only grows

Gottesman-Knill to the complexity class $\oplus L$ to demonstrate how Gottesman-Knill is $\oplus L$-complete. $\oplus L$ is the class of all problems that reduce to simulating a polynomial-size CNOT circuit (a circuit composed entirely of NOT/CNOT gates) acting on an initial state $|0 \cdots 0\rangle$. By this logic, we can see how Gottesman-Knill is $\oplus L$-hard. Since CNOT circuits are a subset of stabilizer circuits, we can apply the identity function itself to reduce a problem in $\oplus L$ to an application of Gottesman-Knill.

Aaronson and Gottesman provide a method to reduce Gottesman-Knill to $\oplus L$ by using a logarithmic-space machine M with an oracle for simulating CNOT circuits, which they describe as a sufficient method of proving the complexity. Since Gottesman-Knill is $\oplus L$ and $\oplus L$-hard, it is also $\oplus L$-complete. This means that problems that reduce to simulating stabilizer circuits are functionally equivalent to problems that reduce to simulating CNOT circuits. This brings up certain limitations because we know that CNOT circuits are not universal! We cannot simulate AND functions using CNOT circuits, and this non-universality extends to Gottesman-Knill too. Thus, the lack of universality is probably the largest limitation of this quantum circuit simulation algorithm.

5 Simulation of Clifford+T Circuits

Bravyi and Gosset[4] provided a further extension on Gottesman-Knill by formulating an algorithm for the approximation of the output probability of some state x, $P_{out}(x)$, when applying a Clifford + T circuit (which consists of Clifford and T gates) to an initial state $|0^n\rangle$, as well as an algorithm for sampling some x from P_{out} with error ϵ. The authors choose to add on the T gate due to the increased computational power it allows for, as it leads to universal computation[6]. However, these algorithms do have exponential runtime, showing a tradeoff between universality and efficiency.

A simple explanation of the algorithms now follows; the specific detailed steps of the first algorithm are contained in the Appendix Section 2. Our goal in the first algorithm is to apply some Clifford + T circuit U to an initial state $|0^n\rangle$ and estimate the output probability of some state x, $P_{out}(x)$, by measuring some fixed register Q_{out}, which will output some random string x with probability $P_{out}(x) = \langle 0^n| U^\dagger \Pi(x) U |0^n\rangle$, where $\Pi(x)$ is the projection of Q_{out} onto $|x\rangle$. First, we replace each T-gate with the "gadget" shown in Appendix Section 2, Step 1. In the gadget, $|A\rangle$ represents a magic state that allows us to access non-Clifford gates; thus, we must add t (where t is the number of T-gates in our circuit) copies of $|A\rangle$ to our initial state, making it $|0^n A^{\otimes t}\rangle$. Then, we can replace all measurements within the gadgets with random post-selection bits. Next, the magic state $|A\rangle^{\otimes t}$ in our initial state can be written as a linear combination of χ stabilizer states, and the actions of the gadgetized circuit (which we will call V) on this linear combination can be simulated with the original Gottesman-Knill theorem. Thus, the final state before running the measurement of Q_{out} can be written as a linear combination of χ stabilizer states, and the measurement of Q_{out} on this state can also be simulated for each term in the linear combination, as measurements map stabilizers to stabilizers. Now, the norm of the post-measurement state can be measured; this norm is clearly related to $P_{out}(x)$, where x was the output string from the measurement of Q_{out}. Bravyi and Gosset show how to calculate this norm with error ϵ in runtime $O(\chi t^3 \epsilon^{-2})$, and they further explain how this overall simulation algorithm can also be used to sample from the output distribution P_{out} with a small statistical error.

The primary problem with these algorithms is that they have exponential runtime in relation to the number of T-gates (as the optimal value of χ turns out to be exponential in t). The authors

logarithmically in the size of the inputs. This can similarly be interchanged with polynomial time reductions, etc, which would instead require that the reduction occur in time that grows polynomially in the size of the inputs.

point out that many problems requiring quantum circuits can still be solved using just a small number of T-gates, making this exponential time somewhat negligible; however, we can still see that an efficient simulation of universal quantum circuits would thus be a daunting task.

6 Applications and Future Directions

6.1 Application: Fault Tolerance Hardware

Stabilizer circuits are likely to be used for fault tolerance hardware, especially since current applications of fault-tolerance constructions consist of stabilizer codes. This application requires us to actually build the hardware, and induces many questions about the architecture of what these circuits will look like. Since we are specifically dealing with quantum computer architecture, one of the biggest issues we may encounter is decoherence, or a loss of information as quantum states change when they interact with their environments. As a result, it is imperative that we design these stabilizer gates to have the fewest number of gates possible. Motivated by applications in fault-tolerance systems, Aaronson and Gottesman proved the *Canonical Form Theorem*, which is very useful for stabilizer circuit synthesis.

Theorem 6.1. *Canonical Form Theorem: Given any circuit consisting of CNOT, Hadamard, and phase gates, there exists an equivalent circuit that applies a round of Hadamard gates only, then a round of CNOT gates only, and so on in the sequence H-C-P-C-P-C-H-P-C-P-C.*

One useful corollary of the Canonical Form Theorem is that any Clifford circuit of n qubits has an equivalent circuit containing $O(n^2 \log n)$ gates. This upper bound proves useful when constructing efficient stabilizer circuits for hardware uses.

6.2 Application: Hidden Shift Problem

The Hidden Shift Problem is defined such that there are two oracles f and g, as well as a hidden shift string s such that $g(x) = f(x + s)$. The goal is to identify the hidden string s in as few calls to f and g as possible. Using quantum methods, one can determine s in one call each to f and g; Bravyi and Gosset applied an implementation of their Clifford Circuit simulation algorithm to this problem, resulting in the plots in Appendix Section 4 charting their success. As can be seen, the implementation correctly identified the bits of the hidden shift string with high probabilities for each bit, showing a potential area of application for successful simulation of quantum circuits.

6.3 Future Directions

Both papers by Aaronson/Gottesman and Bravyi/Gosset have promising findings but can definitely be built off of in future research. One potential area of future research is simplifying the theorems and algorithms discussed in the Aaronson and Gottesman paper. For example, the tableau algorithm has a runtime of $O(n^2)$, but can we find an algorithm that simplifies this runtime to $O(n)$? Similarly, the Canonical Form Theorem describes an 11-gate sequence that circuits can condense down to. This is a complicated sequence, so future research may focus on how we can simplify this to a shorter sequence. Another potential area of research surrounds measurement gates and how we can manipulate them to consolidate quantum circuits. Future research may focus on the effects of post-processing and delaying measurement gates until the end of each circuit, as some evidence points to classical post-processing potentially making universal quantum circuits possible[2].

References

[1] Aaronson, Scott. “BQP and the Polynomial Hierarchy.” Proceedings of the 42nd ACM Symposium on Theory of Computing - STOC ’10, 2010, https://doi.org/10.1145/1806689.1806711.

[2] Aaronson, Scott, and Daniel Gottesman. “Improved Simulation of Stabilizer Circuits.” Physical Review A, vol. 70, no. 5, Nov. 2004, p. 052328. arXiv.org, https://doi.org/10.1103/PhysRevA.70.052328.

[3] E. Bernstein and U. Vazirani. ”Quantum complexity theory.” SIAM J. Comput., 26(5):1411–1473, 1997. First appeared in ACM STOC 1993.

[4] Bravyi, Sergey, and David Gosset. “Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates.” Physical Review Letters, vol. 116, no. 25, June 2016, p. 250501. arXiv.org, https://doi.org/10.1103/PhysRevLett.116.250501.

[5] Fowler, Gary. “Council Post: When Will Quantum Computers Impact Our Day-to-Day?” Forbes, Forbes Magazine, 28 Apr. 2021, https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2021/04/28/when-will-quantum-computers-impact-our-day-to-day/?sh=2f559ae943d9.

[6] Kliuchnikov, Vadym. “Synthesis of Unitaries with Clifford+T Circuits.” ArXiv:1306.3200 [Quant-Ph], June 2013. arXiv.org, http://arxiv.org/abs/1306.3200.

[7] “Quantum Computing Use Cases Are Getting Real–What You Need to Know.” McKinsey amp; Company, McKinsey amp; Company, 14 Dec. 2021, https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/quantum-computing-use-cases-are-getting-real-what-you-need-to-know: :text=Five%20manufacturers%20have%20announced%20plans,many%20use%20cases%20by%20then.

7 Appendix

7.1 Prerequisites for Algorithm for Approximating Output Probabilities

In this section, we describe additional information and notations about stabilizer groups. We define $\mathcal{P}_n$ to be the n-qubit Pauli group. We call $\mathcal{G} \subseteq \mathcal{P}_n$ a stabilizer group if $-I \notin \mathcal{G}$ and if $\mathcal{G}$ is abelian. A codespace of $\mathcal{G}$ consists of the states that are stabilized by $\mathcal{G}$, and we denote a projector Π as $\Pi_{\mathcal{G}}$ if Π projects onto the codespace of $\mathcal{G}$. Lastly, denote $\mathcal{S}_n$ to be the set of all n-qubit stabilizer states; then, we later use the fact that this set is "2-design", or that

$$|\mathcal{S}_n|^{-1} \sum_{\psi \in \mathcal{S}_n} |\psi\rangle\langle\psi|^{\otimes 2} = \int d\mu(\phi)\, |\phi\rangle\langle\phi|^{\otimes 2},$$

where the integral is with respect to the Haar measure on the set of all normalized n-qubit states ϕ.

7.2 Algorithm for Approximating Output Probabilities

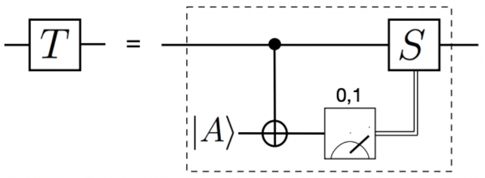
Recall that our goal for this algorithm is to take a Clifford + T circuit U, simulate its effects on input state $|0^n\rangle$, and measure some fixed register Q_{out} on the output, which produces some random string x of length $w = |Q_{out}|$; we would like to find the probability of getting that x from this measurement, which we denote as

$$P_{out}(x) = \langle 0^n| U^\dagger \Pi(x) U |0^n\rangle,$$

where $\Pi(x)$ projects Q_{out} on to $|x\rangle$. For this algorithm, let n denote the number of qubits acted on by U, t be the number of T-gates in U, and c be the number of Clifford gates in U.

Algorithm 2 Bravyi and Gosset's Algorithm for Approximating Output Probabilities

1. First, all T-gates in the circuit U are replaced with the "gadget" shown below, made up of a CNOT, $(0,1)$-measurement, and S-gate.



2. Since each gadget shown above "consumes" a magic state $|A\rangle$, we must add t ancillary qubits so that our initial state is now $|0^n A^{\otimes t}\rangle$. Now, each measurement in the gadget is replaced "post-selection" by $|0\rangle\langle 0|$, which removes the need for the S-gates above.

3. Now, we identify the measurement projector for projecting Q_{out} onto x, which we see is $\Pi = (|x\rangle\langle x|_{Q_{out}} \otimes I_{else}) \otimes |0^t\rangle\langle 0^t|$; this value comes from combining the original measurement projector with the $|0\rangle\langle 0|$ projectors acting on the t ancillary qubits of our initial state. Thus, we currently have

$$P_{out}(x) = 2^t \langle 0^n A^{\otimes t}| V^\dagger \Pi V |0^n A^{\otimes t}\rangle,$$

where V is the new version of the original circuit U (with the T-gates replaced and with each measurement gate replaced).

4. Now, we know that for some $\mathcal{W} \subseteq \mathcal{P}_{n+t}$ with dimension $w+t$ (refer to Appendix Section 1 for prerequisite information on this notation), we must have that the measurement projector $\Pi = \Pi_{\mathcal{W}}$. Then, if $q(j)$ is the j-th qubit of Q_{out}, we know that $\mathcal{W}$ is made up of some generators which we will denote as R_j for $1 \le j \le w$ and R_{w+j} for $1 \le j \le t$, where w is as defined in the problem statement (the exact values of the generators can be found but are not crucial to this overview of the algorithm). Then, define $\mathcal{V}$ as the stabilizer group containing generators $R'_j = V^\dagger R_j V$ (we know these generators can be found this way since applying V maps Pauli operators to Pauli operators). We can then compute each generator R'_j in runtime $O(c+t)$ and thus the group $\mathcal{V}$ in runtime $O((w+t)(c+t))$. Furthermore, we have that
$$V^\dagger \Pi_{\mathcal{W}} V = \Pi_{\mathcal{V}}.$$

5. Then, if $\mathcal{V}_0$ is the subgroup of $\mathcal{V}$ such that it includes all Pauli operators that use only I or Z on the first n qubits, we can find a generating set for $\mathcal{V}_0$ in $O((n+t)^3)$. Then, if the dimension of $\mathcal{V}_0$ is v, then we have
$$P_{out}(x) = 2^t \left\langle 0^n A^{\otimes t}\right| V^\dagger \Pi V \left|0^n A^{\otimes t}\right\rangle = 2^t \left\langle 0^n A^{\otimes t}\right| \Pi_{\mathcal{V}} \left|0^n A^{\otimes t}\right\rangle = 2^{v-w} \left\langle 0^n A^{\otimes t}\right| \Pi_{\mathcal{V}_0} \left|0^n A^{\otimes t}\right\rangle,$$
since $\langle 0^n | P | 0^n \rangle = 0$ for all P in $\mathcal{V}$ but not in $\mathcal{V}_0$.

6. Lastly, we define the group of generators $\mathcal{G}$ consisting of $G_i = \langle 0^n | Q_i | 0^n \rangle$, where Q_i are the generators of $\mathcal{V}_0$. Using some casework, we can calculate $\mathcal{G}$ efficiently and we thus have a final value for
$$P_{out}(x) = 2^{v-w} \left\langle A^{\otimes t}\right| \Pi_{\mathcal{G}} \left|A^{\otimes t}\right\rangle,$$
calculated in $O((w+t)(c+t) + (n+t)^3)$ total runtime.

Through this part of the algorithm, we are able to arrive at the group of generators $\mathcal{G}$ such that
$$P_{out}(x) = 2^{v-w} \left\langle A^{\otimes t}\right| \Pi_{\mathcal{G}} \left|A^{\otimes t}\right\rangle.$$

Next, we would like to estimate the $P_{out}(x)$ value itself, through the following steps:

1. Suppose we have that the magic state $\left|A^{\otimes t}\right\rangle$ can be written as a linear combination of χ stabilizer states, $\sum_{a=1}^{\chi} y_a |\rho_a\rangle$ where each $|\rho_a\rangle$ is some stabilizer state in $\mathcal{S}_t$, the set of stabilizer states with t qubits. Then, for each a, we compute b_a, p_a, and $|\phi_a\rangle$ such that
$$\Pi_{\mathcal{G}} |\rho_a\rangle = b_a 2^{-p_a/2} |\phi_a\rangle,$$
where $|\phi_a\rangle$ is another stabilizer state in $\mathcal{S}_t$; this computation can be done in $O(\chi t^3)$ time for all a.

2. We now have
$$P_{out}(x) = \|\psi\|^2,$$

where $|\psi\rangle = \sum_{a=1}^{\chi} 2^{-(w-v+p_a)/2} y_a b_a |\phi_a\rangle$. To estimate the norm of $|\psi\rangle$, we can use a randomized algorithm, where we choose random stabilizer states θ from $\mathcal{S}_t$. Noting that

$$E_\theta |\langle\theta|\psi\rangle|^2 = \frac{\|\psi\|^2}{d}, E_\theta |\langle\theta|\psi\rangle|^4 = \frac{2\|\psi\|^4}{d(d+1)},$$

where $d \equiv 2^t$ by using the information on $\mathcal{S}_n$ in Appendix Section 1, we can see that the expectation of the random variable $\nu = \frac{d}{L}\sum_{i=1}^{L} |\langle\theta_i|\psi\rangle|^2$ is equal to $\|\psi\|^2$, and the standard deviation of ν is $\sqrt{\frac{d-1}{d+1}}\frac{1}{\sqrt{L}}\|\phi\|^2$, which can be approximated to be roughly $\frac{1}{\sqrt{L}}\|\psi\|^2$ if t is large (since recall that $d \equiv 2^t$).

3. By the Chebyshev Inequality,

$$\Pr[|\nu - E(\nu)| \geq 2\sigma] \leq \frac{1}{4},$$

and so we would have $(1-\epsilon)\|\psi\|^2 \leq \nu \leq (1+\epsilon)\|\psi\|^2$ with probability $\frac{3}{4}$ as long as $L = 4\epsilon^{-2}$.

4. To achieve this kind of accuracy level for any failure probability p, we could simply compute $O(\log(p))$ estimates of ν and use their median as our estimate, giving us an approximation for $\|\psi\|^2$ and thus also $P_{out}(x)$ for some pre-determined failure probability and error.

In the above algorithm, each $\langle\theta_i|\psi\rangle$ can be calculated in $O(\chi t^3)$, and so the estimate for $\|\psi\|^2$ can be calculated in $O(\chi t^3 \epsilon^{-2}\log(p))$ time for error ϵ and failure probability p. Thus, the overall runtime of the total algorithm for approximating $P_{out}(x)$ is $O((w+t)(c+t)+(n+t)^3+\chi t^3\epsilon^{-2}\log(p))$. As we can see, this runtime depends primarily on the number of T-gates t and the number of stabilizers used to represent the magic state $|A^{\otimes t}\rangle$, and the choice of χ. Overall, this result allows for an estimation of the output probability of some random string when applying some Clifford+T circuit to an initial state by means of quantum circuit simulation through creative manipulations of the circuit and application of Gottesman-Knill.

Bravyi and Gosset then provide an extension of the above algorithm for sampling some x from the distribution P_{out} with error level ϵ. The algorithm, which computes the necessary values for calculating $P_{out}^{y}(z|x_1,\ldots,x_{j-1})$ with ϵ error each and with failure probability set to $O(\epsilon w^{-1})$ for optimization reasons, has runtime $O(w(w+t)(c+t)+w(n+t)^3+\chi w^3 t^3\epsilon^{-2}\log(w\epsilon^{-1}))$. Again, the number of T-gates t takes responsibility for most of the runtime complexity here.

Another note on these algorithms is that they are polynomial-time with regards to the number of Clifford gates in U, but exponential-time with regards to the number of T-gates in U (as the optimal value of χ turns out to be exponential in terms of t). Thus, the overall universality of these simulation algorithms is balanced by the lack of efficiency in them.

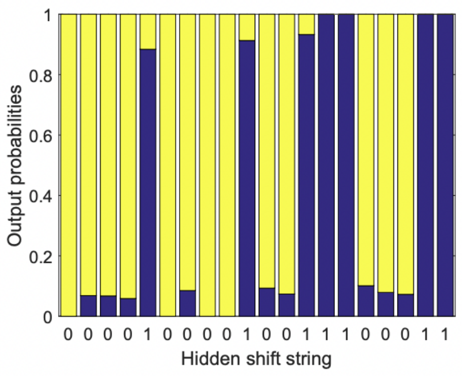
7.3 Tableau Visualizations

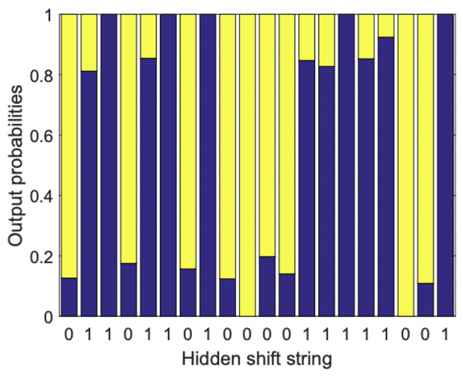
$$\left(\begin{array}{ccc|ccc|c}
x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} & r_1 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
x_{n1} & \cdots & x_{nn} & z_{n1} & \cdots & z_{nn} & r_n \\
\hline
x_{(n+1)1} & \cdots & x_{(n+1)n} & z_{(n+1)1} & \cdots & z_{(n+1)n} & r_{n+1}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

Tableau setup as required to update bits in quadratic time after each gate. Each (x_{ij}, z_{ij}) pairing corresponds to the jth Pauli matrix in the product that determines stabilizer/destabilizer R_i. Each r_i determines the phase of R_i. The overall size is $2n(2n+1)$, which corresponds to the space complexity of the Tableau algorithm.

7.4 Hidden Shift Problem Success Plots

As can be seen in these plots (where colors denote what value for each bit of the hidden shift string was estimated, yellow for 0 and blue for 1), Bravyi and Gosset's implementation of their algorithm for the Hidden Shift Problem was largely successful, with output probabilities essentially matching to the correct bit for each bit of the hidden shift string.


Output probabilities
Hidden shift string


Output probabilities
Hidden shift string