COMPUTATION WITH CHAINED CLOSED TIMELIKE CURVES

by

Mert Can Çıkla

B.S., Computer Engineering, İzmir University of Economics, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2019

ABSTRACT

COMPUTATION WITH CHAINED CLOSED TIMELIKE CURVES

Discussions of Closed Timelike Curves (CTC) led to a few computation models that when used in conjunction with a Turing Machine, yield much more efficient computations. CTC assisted computation has the ability to send a piece of information back in time which breaks the time causality and has various paradoxes associated with it. Computation models deal with these paradoxes differently by making different assumptions. Regardless of the practicality of CTCs, studying different computation models has the possibility to grant valuable insight into complexity classes and their relationships among each other.

The first of these models is proposed by Deutsch. The model avoids paradoxes by assuming that the states that enter and exit the machine constitute a probability distribution and that the machine outputs a stationary distribution. The second model, proposed by Lloyd et al. has the ability to discard unwanted outcomes, via the assumption of time-related paradoxes that can be caused by CTC interaction to be impossible. The third model, proposed by Say and Yakaryılmaz improves upon Deutsch's model and deals with some of its shortcomings.

In this thesis, we demonstrate and analyze how these CTC based computation models help solve NP-complete, and some other problems efficiently and propose a more cost-efficient version of one such algorithm from the literature. Lastly, we explore and discuss some odd and interesting properties of calculations in DCTCs.

ÖZET

ZİNCİR HALİNDE KAPALI ZAMANSI EĞRİLER İLE HESAPLAMA

Kapalı Zamansı Eğriler üzerine tartışmalar, bu eğrileri kullanarak Turing Makinesi ile hesaplamaları daha efektif hale getiren hesaplama modellerine yol açmıştır. Bu modeller verileri zamanda geriye gönderme gücüne sahip olmalarından ötürü zamanın nedenselliğini bozarak paradokslara yol açabilmektedirler. Modeller kendi içlerinde bu paradoksların önüne geçmek için yaptıkları varsayımlarına göre farklılık göstermektedir. Farklı hesaplama modelleri üzerine çalışmaların karmaşıklık sınıfları ve bu sınıfların birbirleri ile olan etkileşimleri üzerine açıklık getirebildiği bilinmektedir.

Hesaplama modellerinin ilki Deutsch tarafından önerilmiş olup, paradoks oluşmasını engellemek için zamanda geri gönderilen bitlerin tek bir halde bulunmak yerine bir olasılık dağılımı içinde bulunduklarını varsaymaktadır. İkinci model Lloyd vd tarafından önerilmiştir ve istenilmeyen olasılık çıktılarının sonuç kümesinden çıkarılabilme gücüne sahiptir. Bu gücü ise makinenin her zaman kararlı olması ve makine ile etkileşim sonucu çıkabilecek paradoksların olma ihtimalinin sıfır olduğu varsayımı sayesinde elde etmektedir. Üçüncü model Say ve Yakaryılmaz tarafından önerilmiştir ve Deutsch'un modelini geliştirerek modelin birkaç dezavantajını ortadan kaldıran iyileştirmeler sunmaktadır.

Bu tezde, kapalı zamansı eğriler tabanlı hesaplama modelleri incelenip bu modellerin NP-complete problemlere nasıl efektif çözümler getirdiği analiz edilmektedir. İkinci olarak literatürde geçen bu sınıfa ait bir algoritmanın aynı hatayı daha az devre masrafı ile elde edebilen bir versiyonu önerilmiştir. Son olarak Deutsch'un modeli ile hesaplamalarda ortaya çıkabilen bir durum üzerine tartışma ve analiz yer almaktadır.

TABLE OF CONTENTS

ABSTRACT iii
ÖZET iv
LIST OF FIGURES vi
LIST OF SYMBOLS viii
LIST OF ACRONYMS/ABBREVIATIONS ix
1. INTRODUCTION 1
1.1. Related Work 1
2. PRELIMINARIES 4
2.1. Quantum Computation 6
3. COMPUTATION WITH DEUTSCHIAN CTCS 13
3.1. Primality Testing 13
3.2. Boolean Satisfiability Problem 16
4. COMPUTATION WITH PCTCS 22
4.1. Postselection 22
4.2. PCTCs 23
4.3. Alternative to BW Algorithm for SAT 26
4.3.1. Probability Amplification 29
4.4. Performance Comparison 30
4.4.1. Performance Evaluation 33
5. CHAIN-CTC MODEL 37
5.1. Primality Testing with Chain-CTCs 37
5.2. Chain-CTCs with multiple branchings 40
6. RANDOMIZED COMPUTATIONS WITH CTCS 44
6.1. Chain-CTC Examples 47
7. CONCLUSION 50
REFERENCES 51

LIST OF FIGURES

Figure 2.1. Markov chain with two states A,B and transition probabilities T_{ij} 5

Figure 2.2. Measurement device on a quantum circuit. Single wire on the left carries a qubit and the two wires on the right represent a classical bit output 7

Figure 2.3. Bloch Sphere for qubit representation. 12

Figure 3.1. Markov Chain representation, CTC transition matrix and the corresponding stationary distribution for CTC in Algorithm 1 if n is prime. 14

Figure 3.2. Markov Chain representation, CTC transition matrix and the corresponding stationary distribution for CTC in Algorithm 1 if n is composite. 15

Figure 3.3. DCTC algorithm for primality testing. 15

Figure 3.4. DCTC algorithm for the SAT problem. 17

Figure 3.5. Markov Chain representation of the CTC used in Algorithm 2, if θ is satisfiable. 18

Figure 3.6. Markov Chain representation, CTC transition matrix and the corresponding stationary distribution for the CTC used in Algorithm 2, if θ is unsatisfiable. 21

Figure 4.1. Brun and Wilde's PCTC circuit for SAT solving. 23

Figure 4.2. Alternative PCTC circuit for SAT. 26

Figure 4.3. Alternative PCTC circuit for SAT with probability amplification. . 30

Figure 4.4. Error reduction per added circuit cost comparison for Algorithms 4.1 and 4.3. $m = \log n$, $s = 10$. 34

Figure 4.5. Error reduction per added circuit cost comparison for Algorithms 4.1 and 4.3. $\mathcal{M}$ axis with logarithmic scaling. $m = n$, $s = 10$. . . . 35

Figure 4.6. Error reduction per added circuit cost comparison for Algorithms 4.1 and 4.3. $\mathcal{M}$ axis with logarithmic scaling. $m = 2^{n-1}$, $s = 10$. . 36

Figure 5.1. Primality testing with Chain-CTCs. 38

Figure 5.2. Chain-CTC algorithm with multiple branchings. 40

Figure 6.1. Simple DCTC algorithm with a coin flip inside its CTC. 44

Figure 6.2. Simple DCTC algorithm with a coin flip outside its CTC. 45

Figure 6.3. Chained version of Algorithm 5. 47

Figure 6.4. Chained CTC algorithm example. 48

LIST OF SYMBOLS

C_b	Left stochastic transition matrix of CTC b
C_b^{ijk}	Left stochastic transition matrix of CTC b with i, j, k branching outcomes
$\mathcal{M}_A$	Error reduction per cost measure for algorithm A
ε_A	Error rate for algorithm A
θ	A boolean formula
λ_A	Circuit cost measure for algorithm A
ϕ_b	Stationary distribution of CTC b
ϕ_b^{ijk}	Stationary distribution of CTC b with i, j, k branching outcomes

LIST OF ACRONYMS/ABBREVIATIONS

CTC	Closed Timelike Curve
PCTC	Closed Timelike Curve based on Postselection
DCTC	Closed Timelike Curve based on Deutsch's model
SAT	Boolean Satisfiability Problem
CRR	Causality Respecting Register

1. INTRODUCTION

Time-travel has long been a subject of many science fiction books and movies. The idea is rightfully associated with fiction and trying to explain it using the best known scientific theorems that explain our universe leads to logical paradoxes, most notably the Grandfather Paradox. Gödel's solution to Einstein's field equations of General Relativity implies the possibility of the existence of Closed Timelike Curves (CTCs) as a region in spacetime that enables time-travel to the past. The possibility is a highly debated topic amongst physicists. CTC-based computers have been proposed to use CTCs to send bits of information back in time to speed up computations [1–3]. Regardless of whether it is possible to build such a device and use it for computational purposes, the theory has insightful implications for complexity classes and their relationships. Similar to nondeterminism or the concept of an oracle machine that can answer decision problem-based questions, CTC-based computation is a useful theoretical device to investigate limits of computation. Previous research indicates that CTC-based computation is exceptionally powerful, capable of solving NP-complete problems efficiently [2]. This extreme computation power suggests the impossibility of closed time-like curves. From a computability perspective, a version of the model with some additional unrealistic assumptions is able to break the Turing barrier, allowing otherwise undecidable problems, such as the halting problem, to become decidable [4].

1.1. Related Work

Novikov's self-consistency principle requires a realizable model of time travel to avoid the Grandfather Paradox. The Grandfather paradox emerges when a time-traveler goes back in time to kill his grandfather whereby the time-traveler is not born which deems the killing of the grandfather impossible which in turn allows the time-traveler to be born. Deutsch, in his seminal paper, proposed a model that resolves the contradiction associated with the grandfather paradox [1].

The model based on the many-worlds interpretation, imposes that the traveled state is not a single deterministic one. Deutsch explains the model using a probability distribution that leads to consistency by allowing the grandfather to live with probability $\frac{1}{2}$ which leads to the traveler being born with probability $\frac{1}{2}$ hence able to successfully travel. Brun demonstrated that a CTC-based computation is capable of solving NP-complete or even PSPACE-complete problems in a short amount of time [2]. The demonstration, however, lacked a concrete computation model to be used in further studies. Deutsch's model has gained more acceptance in relevant work for its self-consistency.

Another model that gained acceptance in the field, establishes a framework without the aforementioned inconsistencies in which CTCs can exist. The PCTC model, based on postselected quantum teleportation, is suggested by Lloyd et al. [5]. The Postselection model inherently avoids the Grandfather Paradox but is computationally weaker than Deutsch's model [6]. In Ref. [6] Brun and Wilde also showed that a single postselected CTC is sufficient to make any quantum measurement with certainty. A third model called Transition probability CTC (TCTC) was proposed by Allen [7] to deal with some of the shortcomings of previous models. Brun and Wilde showed that TCTCs are equivalent to PCTCs in terms of computational power. Bacon showed that 1-qubit CTCs allow NP problems to be solvable in polynomial time [3] . Later, Aaronson and Watrous proved $\mathsf{P}_{\mathsf{CTC}} = \mathsf{BQP}_{\mathsf{CTC}} = \mathsf{BPP}_{\mathsf{CTC}} = \mathsf{PSPACE}$, namely, given a polynomial width CTC, the computation power increases to be able to solve any problem in PSPACE in polynomial time and that the powers of classical and quantum polynomial time computers become equivalent with CTCs [8].

Say and Yakaryılmaz showed that weaker models of computation such as finite and push-down automaton also gain power when augmented with CTCs [9]. They also showed that CTCs give the power of limited nondeterminism to deterministic machines. They examined the power granted by a single CTC bit and showed that $\mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PP}$ and $\mathsf{BPP}_{\mathsf{CTC}[1]} = \mathsf{BPP}_{\mathsf{path}}$. O'Donnell and Say showed that increasing the number of CTC bits up to $\log n$ does not grant any additional computation power [10] and they generalized the result to logarithmically many qubits instead of classical and proved the stronger result $\mathsf{BQP}_{\mathsf{QCTC}[\log]} = \mathsf{BQP}_{\mathsf{CTC}[1]} = \mathsf{PP}$ [11].

Aaronson et al. used a looser model to examine the computability aspect of CTC-based computation and showed that problems that are Turing-reducible to the infamous halting problem are solvable by classical or quantum computers with CTCs [4]. They also examined the less powerful PCTC model and showed that postselection does not grant any additional power in terms of computability when the computation is required to halt.

2. PRELIMINARIES

Markov Property. A stochastic process with random variables X_i and state space x_i has the Markov property if

$$P(X_{n+1} = x | X_n = x_n) = P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n).$$

In other words, the Markov property holds if the probability distribution of the next state of a process is dependent only on the last state preceding it.

Markov Process. A stochastic process is a Markov process if it satisfies the *Markov Property.*

Picking a ball from a bag that contains infinitely many, different colored balls is a Markov process. However, if the number of balls with a certain color is finite, the process does not hold the Markov property because selecting the next ball depends on how many of the balls of the same color were picked previously.

Markov Chain. Markov Chain is a Markov process with a countable state space S of size n, outcomes from random variables X_i defined with a column stochastic state transition matrix T where

$$T_{ij} = P(X_{n+1} = x_i | X_n = x_j),$$

$$\sum_{i=0}^{n} T_{ij} = 1 \text{ for all } j.$$

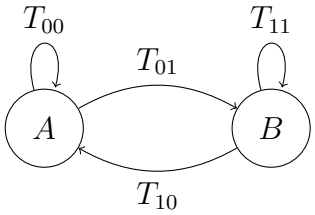


Figure 2.1: Markov chain with two states A,B and transition probabilities T_{ij}

Stationary Distribution. Let ϕ be a probability distribution over the state space S of a Markov chain with transition matrix T. ϕ is a stationary of this Markov chain if and only if $\phi = \phi T$

BPP. Bounded-error polynomial time, BPP, is the class of problems decidable by a Turing machine in polynomial time with error probability $\varepsilon < \frac{1}{3}$.

PP. Probabilistic polynomial time, PP, is the class of problems decidable by a Turing machine in polynomial time with error probability $\varepsilon < \frac{1}{2}$. Note that the bound of $\frac{1}{3}$ is arbitrary for BPP and can be any $p \leq \frac{1}{2} - c$ for a positive constant c. This bound allows probability amplification to be applied to BPP to decrease error rate by running multiple instances of the same algorithm and taking the majority decision, whereas it is not possible to do so for PP.

BQP. Bounded-error quantum polynomial time, BQP, is the class of problems decidable by a quantum Turing machine in polynomial time with error probability $\varepsilon < \frac{1}{3}$.

$\mathsf{P}_{\mathsf{CTC}}$. Class of problems decidable by a Turing machine assisted with a Deutschian CTC that uses classical bits in polynomial time.

$\mathsf{P}_{\mathsf{QCTC}}$. Class of problems decidable by a Turing machine assisted with a Deutschian CTC that uses quantum bits in polynomial time.

$\mathsf{BPP}_{\mathsf{path}}$. Class of problems decidable by a Turing machine in polynomial time with error probability $\varepsilon < \frac{1}{3}$ and all computational paths of the machine are of the same length.

$\mathsf{BPP}_{\mathsf{CTC[n]}}$. Class of problems decidable by a classical Turing Machine with bounded error with access to a Deutschian CTC that uses n classical bits.

2.1. Quantum Computation

Qubit. Basic unit of information analogous to a classical binary digit under quantum mechanical rules. Due to the nature of quantum mechanics, the state is described with a probability distribution based on amplitudes. Qubits are described using the bra-ket notation as a superposition of two orthonormal basis states of which the most used are $|0\rangle$ and $|1\rangle$.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

A qubit $|q\rangle$ is described in $|0\rangle, |1\rangle$ basis as follows:

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

α and β are called amplitudes and α^2 gives the probability of the qubit being in state 0 and β^2 gives the probability of the qubit being in state 1 when measured. After the measurement, the qubit obeys classical probability laws hence, $\alpha^2 + \beta^2 = 1$.

Measurement. A single qubit which can hold an infinite amount of information on its amplitudes loses the information it carries after the measurement as described above. While the type of measuring device changes depending on the circuit's implementation, any measurement attempt done on a qubit collapses any superpositioned state into one of its pure basis states. The cause of collapse is not the intent or act but the measurement device's destructive effect on the qubit itself. An oversimplified example is shining a ray of light that carries several photons to observe a particular photon's polarization. Even if a single photon were to be used to measure it, the process is still destructive.

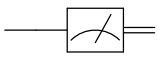


Figure 2.2: Measurement device on a quantum circuit. Single wire on the left carries a qubit and the two wires on the right represent a classical bit output

Quantum Gate. Basic unit of transformation used in quantum circuits. An n bit quantum gate is described with a $2^n \times 2^n$ unitary matrix.

Pauli-X Gate. Quantum analogue of the classical NOT logic gate in the standard basis. Operates on a single bit and is described with the unitary matrix

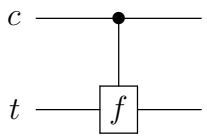
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

When applied to a qubit $|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, it flips the probability amplitudes of $|0\rangle$ and $|1\rangle$

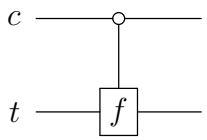
$$X\,|q\rangle = \beta\,|0\rangle + \alpha\,|1\rangle$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Controlled Gates. A controlled gate acts on two sets of qubits, control and target. Control qubit is used as an if conditional and depending on its value, the gate applies a transformation to the second set of qubits, the target. In circuit form, the operation "apply f to t if $c = 1$" is represented as follows:



An anti-control gate that checks c for 0 is represented with:



The circuit above applies f to t if $c = 0$.

Controlled-NOT gate. Most commonly referred to as the CNOT gate, it flips the value of its second qubit if the first qubit is 1, otherwise, the second qubit's value remains the same. Two qubit CNOT gate has the following matrix representation.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Two different circuit representations of the CNOT are given below,

Hadamard Gate. Most often used as a single qubit gate to transform the qubit into a superposition of its two states in equiprobability. It is represented by H and has the unitary matrix,

$$\begin{aligned} H =& \frac{1}{\sqrt{2}} \Big(\big(|0\rangle + |1\rangle \big) \langle 0| + \big(|0\rangle - |1\rangle \big) \langle 1| \Big) \\ =& \frac{1}{\sqrt{2}} \left(\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 \end{bmatrix} + \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \begin{bmatrix} 0 & 1 \end{bmatrix} \right) \\ =& \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \right) \\ =& \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}. \end{aligned}$$

After applying the gate to a $|0\rangle$,

$$H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

the qubit has probability $\left(\frac{1}{\sqrt{2}}\right)^2$ to be measured 0 or 1.

Hadamard gates can be applied to any number of qubits n as a $2^n \times 2^n$ transformation. The single qubit version explained earlier is the H. Higher dimension Hadamard gates can be calculated by using the tensor product.

Tensor Product. Denoted by $\otimes$ is an operation that, within the context of quantum computation, allows the representation of multiple lower dimension gates to be combined into a single higher dimensional one. Let M and N be single qubit gates in a circuit. We can obtain a two-qubit gate L by using the tensor product.

$$\begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix} \otimes \begin{bmatrix} n_{1,1} & n_{1,2} \\ n_{2,1} & n_{2,2} \end{bmatrix} = \begin{bmatrix} m_{1,1} \begin{bmatrix} n_{1,1} & n_{1,2} \\ n_{2,1} & n_{2,2} \end{bmatrix} & m_{1,2} \begin{bmatrix} n_{1,1} & n_{1,2} \\ n_{2,1} & n_{2,2} \end{bmatrix} \\ m_{2,1} \begin{bmatrix} n_{1,1} & n_{1,2} \\ n_{2,1} & n_{2,2} \end{bmatrix} & m_{2,2} \begin{bmatrix} n_{1,1} & n_{1,2} \\ n_{2,1} & n_{2,2} \end{bmatrix} \end{bmatrix}$$

$$L = \begin{bmatrix} m_{1,1}n_{1,1} & m_{1,1}n_{1,2} & m_{1,2}n_{1,1} & m_{1,2}n_{1,2} \\ m_{1,1}n_{2,1} & m_{1,1}n_{2,2} & m_{1,2}n_{2,1} & m_{1,2}n_{2,2} \\ m_{2,1}n_{1,1} & m_{2,1}n_{1,2} & m_{2,2}n_{1,1} & m_{2,2}n_{1,2} \\ m_{2,1}n_{2,1} & m_{2,1}n_{2,2} & m_{2,2}n_{2,1} & m_{2,2}n_{2,2} \end{bmatrix}$$

In a similar fashion, multiple H gates can be combined into a multi-qubit Hadamard gate.

$$H \otimes H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$H^{\otimes 2} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

Rotation Gate. A qubit representing an electron's spin in 3D space is best visualized on a Bloch sphere as a vector. There are three rotation operators defined $R_x(\alpha), R_y(\alpha), R_z(\alpha)$ for α radian rotations about each axis on a Bloch sphere about x, y, z and are defined as follows,

$$R_x(\alpha) = \begin{bmatrix} \cos\frac{\alpha}{2} & -i\sin\frac{\alpha}{2} \\ -i\sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{bmatrix},$$

$$R_y(\alpha) = \begin{bmatrix} \cos\frac{\alpha}{2} & -\sin\frac{\alpha}{2} \\ \sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{bmatrix},$$

$$R_z(\alpha) = \begin{bmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{bmatrix}.$$

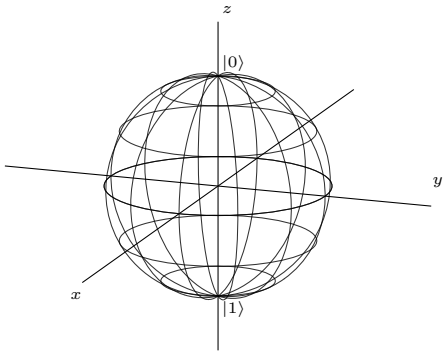


Figure 2.3: Bloch Sphere for qubit representation.

This brief introduction to the quantum computation should be enough to cover the background necessary for this thesis, for more information please refer to [12].

3. COMPUTATION WITH DEUTSCHIAN CTCS

Deutsch's model for computation with CTCs (DCTC model) operates with two sets of bits. The first set, called *Chronology Respecting* (CR), represents the finite amount of classical or quantum information which obeys chronology and acts like regular memory. The second set represents the time traveling bits that goes through closed timelike curve which we will be referring as the CTC bits from now on. The CTC bits in Deutsch's model are represented by a probability distribution as a countermeasure to avoid the Grandfather Paradox. This restriction constricts the possible outcomes of a CTC. It is assumed that the nature sets the CTC bits to a fixed point representing a stationary distribution of the CTC transition.

The model's constraint on the CTC to only accept Markov chains with stationary distributions limits their usability classically. This constraint is less restraining in the quantum world where every quantum operation has a fixed-point (analogous to stationary distribution of a quantum operation) [8].

3.1. Primality Testing

The operation of DCTC based computation is demonstrated below. Algorithm 1 is a DCTC algorithm to calculate whether a given number is prime or not. The algorithm is a simple demonstration of computational power that determines if a number n is prime. We will analyze the algorithm by writing its left stochastic transition matrix C_b first in two separate cases whether n is prime. If n is prime, the machine will never set $b = 1$. With a small probability exponentially scaling with the input length, the machine transitions to $b = 0$. So the machine has no means of setting $b = 1$ but a small chance of setting $b = 0$. With the remaining probability the machine does not alter the distribution of the CTC bit. Mapping this transition behaviour to a Markov chain results in a stationary distribution of $b = 0$ with probability 1.

$$C_b = \begin{bmatrix} 1 & 2^{-n} \\ 0 & 1-2^{-n} \end{bmatrix} \qquad \phi_b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

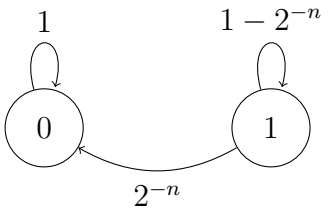


Figure 3.1: Markov Chain representation, CTC transition matrix and the corresponding stationary distribution for CTC in Algorithm 1 if n is prime.

Assuming n is a composite number with k many factors, the machine transitions into a state where $b = 1$ with probability $\frac{k}{n-2}$ because k many uniformly random selections of i will satisfy $n \mod i = 0$ out of $n-2$. For any other i, it leaves b as it is or sets it to 0. This transition behaviour corresponds to the transition matrix and stationary distribution in Figure 3.2.

$$C_b = \begin{bmatrix} 1 - \dfrac{k}{n-2} & 2^{-n} \\ \dfrac{k}{n-2} & 1 - 2^{-n} \end{bmatrix} \qquad \phi_b = \begin{bmatrix} \dfrac{2^{-n}}{\dfrac{k}{n-2} + 2^{-n}} \\ \dfrac{\dfrac{k}{n-2}}{\dfrac{k}{n-2} + 2^{-n}} \end{bmatrix}$$

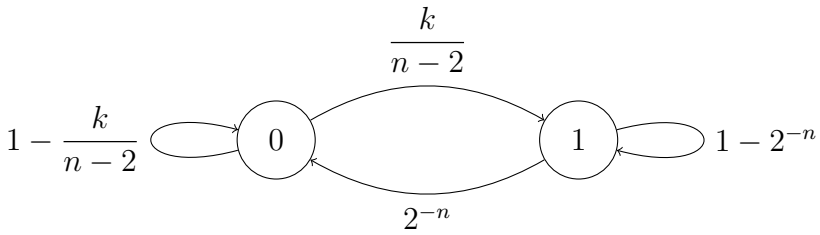


Figure 3.2: Markov Chain representation, CTC transition matrix and the corresponding stationary distribution for CTC in Algorithm 1 if n is composite.

Algorithm 1 Primality Testing

read b from CTC

print b

generate a random integer $i \in \{2, 3, \ldots, n-1\}$

if $n \mod i == 0$ **then** set $b = 1$

with probability 2^{-n}, set $b = 0$

send b via CTC

Figure 3.3: DCTC algorithm for primality testing.

Reading 0 from ϕ_b, ε is the algorithm's probability of incorrectly identifying a composite number as prime.

$$\begin{aligned}
\varepsilon &= \frac{\frac{1}{2^n}}{\frac{k}{n-2} + \frac{1}{2^n}} \\
&= \frac{\frac{1}{2^n}}{\frac{k2^n + n - 2}{(n-2)2^n}} \\
&= \frac{n-2}{k2^n + n - 2} \\
&< \frac{n}{k2^n + n} \\
\varepsilon &< \frac{n}{2^n}
\end{aligned}$$

ε is upper bounded by $\frac{n}{2^{-n}}$ which tends to 0 as n grows.

3.2. Boolean Satisfiability Problem

Boolean Satisfiability (SAT) is the canonical NP-complete problem. For a given boolean formula θ with boolean variables $x_1, x_2, \ldots, x_n$, the objective is to answer whether θ can be 1 for some $X = x_1 x_2 \ldots x_n$.

$$\theta = (x_1 \vee x_2) \wedge (\neg x_3)$$

For the example above, the equation is satisfiable in three ways $X = \{100, 010, 110\}$ so an SAT solver is expected to return true for this instance.

A naive SAT algorithm that uses brute force requires 2^n combinations of X to be explored in the worst case to solve an SAT instance with n variables.

Algorithm 2 SAT

read b from CTC

with probability 2^{-n^2}, set $b = 0$

generate a random string $x \in \{0,1\}^n$

if x *satisfies* θ **then** set $b = 1$

send b via CTC

Figure 3.4: DCTC algorithm for the SAT problem.

In order to represent the CTC computation in Algorithm 2 as a Markov Chain with a transition matrix requires four outcomes to consider depending on the success and probabilities associated with the second and fourth lines of the algorithm. However, because the matrix is column stochastic, we only need to calculate one entry from each column of the matrix and we can calculate the other entry by subtracting it from 1.

In order to calculate the first column of the transition matrix, which considers the case that b is received 0, we only need to consider the success probability of the fourth line which sends $b = 1$ if a generated x is one of the k possible satisfying assignments out of the 2^{-n} possible

$$\frac{k}{2^n},$$

and we obtain the other entry by subtracting it from 1

$$1 - \frac{k}{2^n},$$

Hence the first column of the transition matrix becomes

$$\begin{bmatrix} 1 - \dfrac{k}{2^n} \\ \\ \dfrac{k}{2^n} \end{bmatrix}.$$

The second column of the transition matrix where b is received 1 can be calculated by considering the bit transitioning to 0. This transition occurs only if the probability check on the second line succeeds and the x generated out of the possible 2^n is not one of the k that are satisfying the formula. Multiplying these two probabilities, we obtain the first entry and by subtracting it from 1 we get the corresponding column vector,

$$\begin{bmatrix} \frac{1}{2^{n^2}}\left(\frac{2^n-k}{2^n}\right) \\ 1-\left(\frac{1}{2^{n^2}}\left(\frac{2^n-k}{2^n}\right)\right) \end{bmatrix} = \begin{bmatrix} \frac{2^n-k}{2^{n^2+n}} \\ 1-\left(\frac{2^n-k}{2^{n^2+n}}\right) \end{bmatrix}.$$

Combining the two column vectors we obtain the transition matrix C_b.

$$C_b = \begin{bmatrix} 1-\frac{k}{2^n} & \frac{2^n-k}{2^{n^2+n}} \\ \frac{k}{2^n} & 1-\left(\frac{2^n-k}{2^{n^2+n}}\right) \end{bmatrix}.$$

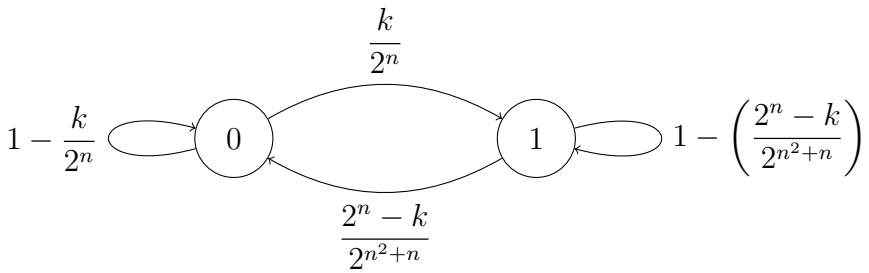


Figure 3.5: Markov Chain representation of the CTC used in Algorithm 2, if θ is satisfiable.

$$\begin{bmatrix} 1-\dfrac{k}{2^n} & \dfrac{2^n-k}{2^{n^2+n}} \\ \dfrac{k}{2^n} & 1-\left(\dfrac{2^n-k}{2^{n^2+n}}\right) \end{bmatrix} \phi_b = \phi_b$$

Let $\phi_b = \begin{bmatrix} x \\ 1-x \end{bmatrix}$.

$$\left(1-\frac{k}{2^n}\right)x + \frac{(1-x)(2^n-k)}{2^{n^2+n}} = x$$

$$\left(\frac{k}{2^n}\right)x + \left(1-\left(\frac{2^n-k}{2^{n^2+n}}\right)\right)(1-x) = 1-x$$

$$\begin{aligned} x &= \left(1-\frac{k}{2^n}\right)x + \frac{(1-x)(2^n-k)}{2^{n^2+n}} \\ x &= \frac{x2^n-kx}{2^n} + \frac{(1-x)(2^n-k)}{2^{n^2+n}} \\ x &= \frac{x2^{n^2+n}-kx2^{n^2}}{2^{n^2+n}} + \frac{(1-x)(2^n-k)}{2^{n^2+n}} \\ x &= \frac{x2^{n^2+n}-kx2^{n^2}+2^n-x2^n-k+kx}{2^{n^2+n}} \\ x2^{n^2+n} &= x2^{n^2+n}-kx2^{n^2}+2^n-x2^n-k+kx \\ 0 &= -kx2^{n^2}+2^n-x2^n-k+kx \\ 0 &= x\left(-k2^{n^2}-2^n+k\right)+2^n-k \\ x &= \frac{2^n-k}{k2^{n^2}+2^n-k} \end{aligned}$$

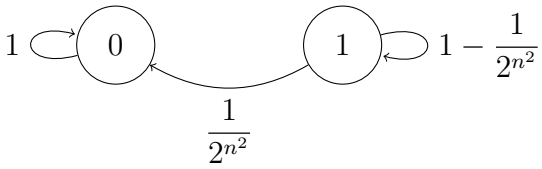
$$\phi_b = \begin{bmatrix} \dfrac{2^n - k}{k2^{n^2} + 2^n - k} \\ 1 - \left(\dfrac{2^n - k}{k2^{n^2} + 2^n - k} \right) \end{bmatrix}$$

The probability of reading 0 from b, the first entry in ϕ_b, corresponds to the error rate of the algorithm ε, if θ is satisfiable.

$$\varepsilon = \frac{2^n - k}{k2^{n^2} + 2^n - k},$$

$\varepsilon \approx 0$ for large n for any k where $0 < k \leq 2^n$.

If $k = 0$ (θ is unsatisfiable), $\varepsilon = 1$ which is the probability of reading 0 from the CTC. However, for an unsatisfiable formula, the actual error rate is $1 - \varepsilon$. Observing this is much simpler by separately analyzing the algorithm for unsatisfiable θ. This analysis is much simpler because the fourth line never executes and there is only one branching possible. We can deduce that the algorithm detects unsatisfiable boolean formulas with certainty because the stationary distribution always yields a 0.



$$CTC = \begin{bmatrix} 1 & \frac{1}{2^{n^2}} \\ 0 & 1 - \frac{1}{2^{n^2}} \end{bmatrix} \qquad \phi = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Figure 3.6: Markov Chain representation, CTC transition matrix and the corresponding stationary distribution for the CTC used in Algorithm 2, if θ is unsatisfiable.

4. COMPUTATION WITH PCTCS

The second model of computation using CTCs is based on quantum postselection as described by Lloyd et al. [5]. In this model, which will be referred to as PCTC, logically contradicting events that are caused by time-travel never occur i.e. the time-traveller can never succeed in killing their grandfather. This condition of the model can be utilized to force the CTC qubit into a paradox for unwanted outcomes thereby granting a computer with a PCTC bit the power of postselection.

4.1. Postselection

Postselection is the ability to discard a set of unwanted outcomes. For a probabilistic algorithm with any non-zero success probability, postselection can be used to improve that success probability by postselecting on the desired outcome, renormalizing the outcome probabilities excluding unwanted ones. Lets assume that we have a probabilistic algorithm with the following outcomes.

$$P(Decided) = 0.25$$
$$P(Undecided) = 0.75$$
$$P(Accept \wedge Decided) = 0.15$$
$$P(Reject \wedge Decided) = 0.10$$

The algorithm decides with probability 0.25 and when it decides, it accepts with probability 0.6 and rejects with probability 0.4. Applying postselection to this algorithm on outcome *Decided*, we assume and apply the condition that the option *Decided* already occurred. In other words, $P(Accept)$ becomes $P(Accept|Decided)$ and $P(Reject)$ becomes $P(Reject|Decided)$.

By applying conditional probability rules, we get

$$P(Accept|Decided) = \frac{P(Accept \wedge Decided)}{P(Decided)} = \frac{0.15}{0.25} = 0.6,$$

$$P(Reject|Decided) = \frac{P(Reject \wedge Decided)}{P(Decided)} = \frac{0.10}{0.25} = 0.4.$$

Without any postselection this algorithm would have accepted only with probability 0.15 and rejected with probability 0.10. The accept and reject probabilities increased four times because the initial $P(Decided) = 0.25$ probability becomes 1 with postselection.

4.2. PCTCs

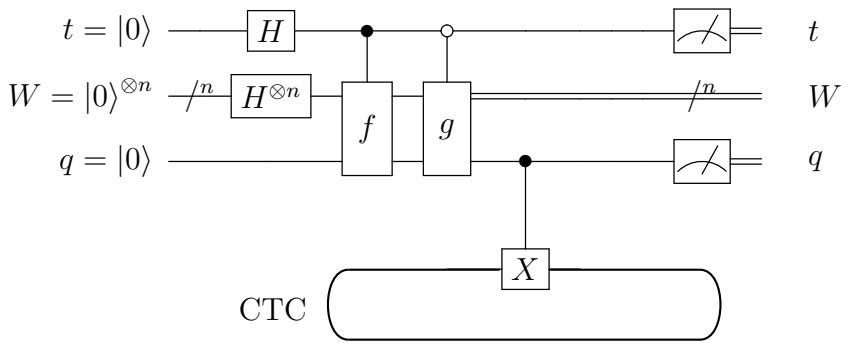


Figure 4.1: Brun and Wilde's PCTC circuit for SAT solving.

Brun and Wilde proposed a PCTC circuit (Figure 4.1) for solving SAT instances [6]. The circuit works with $n + 2$ qubits for an SAT instance of size n and a single PCTC qubit. $|t\rangle$ is a qubit used as flag to control the unitaries f and g. W represents n qubit sized register and q is a qubit used for postselection that interacts with the PCTC.

The circuit works as follows:

(i) Qubits t, q and the register W are initialized to $|0\rangle$.
(ii) Hadamard gates are applied to both t and W.
(iii) Apply a controlled circuit f with t as control to W and $q = |0\rangle$ which maps $W|0\rangle \mapsto W|1\rangle$ if W does not satisfy θ, leaves it unchanged otherwise.
(iv) Apply a controlled circuit g with t as control to W and $q = |0\rangle$ which maps $W|0\rangle \mapsto W|1\rangle$ if W does not hold all 0's and leaves it unchanged otherwise.
(v) Apply a CNOT gate from q to the PCTC qubit.

After circuit execution, if $t = 1$ is measured, the θ is deemed satisfiable with a satisfying set of assignments stored in W. If $t = 0$, it is unsatisfiable with certainty and W holds all zeros.

Step (ii) sets t and W in a superposition so that all of the possible outcomes are equally likely. Circuits f and g in step (iii) and (iv) are applied in mutual exclusion. Step (v) is the postselection step which introduces a paradoxical outcome to the circuit if q is 1 which, by the definition of the PCTC model, cannot happen so at the end of execution $q = 0$ with probability 1.

If the SAT instance θ is unsatisfiable, f can not run and change q because there are no satisfying assignments possible. Because $q = |0\rangle$ option is discarded via post-selection, this implies $t = |0\rangle$ and step (iv) is executed with probability 1. Step (iv), which applies the gate g to W and q. After execution of g, W and q hold $|00\ldots0\rangle|0\rangle$ so we measure all zeros. The result is that the circuit detects unsatisfiable instances with probability 1.
If the SAT instance is satisfiable for m assignments out of 2^n possible: Step (iii) produces m satisfying assignments and the other $2^n - m$ are discarded via postselection. The only possible outcome from step (iv) is all zeros which is the erroneous outcome.

Overall the circuit finds satisfying assignments with probability

$$\frac{m}{m+1},$$

and the error rate of the circuit is

$$\frac{1}{m+1}.$$

Brun and Wilde proposed applying probability amplification by repeating steps (i)-(iv) to decrease error rate. Amplification is achieved by using k copies of t, W and q registers with accompanying Hadamard gates as initiliazers and k many f and g gates. All of the copies share the single PCTC qubit however the X gate on it now only runs if one or more of the k many q registers are 1 which applies the postselection. A non satisfying θ is only deemed so if all of the k many t registers are 0's. This increases the initial success probability to

$$\frac{(m+1)^k - 1}{(m+1)^k},$$

and reduces the error probability to

$$\frac{1}{(m+1)^k},$$

with polynomial increase in circuit size for an exponential decrease in error rate, hence a logarithmic overhead. Without any postselection and amplification, the circuit depicted in Figure 4.1 can successfully identify an SAT instance with only a probability of $\frac{m}{2^n}$, which is equivalent to randomly guessing and checking for a satisfying assignment.

4.3. Alternative to BW Algorithm for SAT

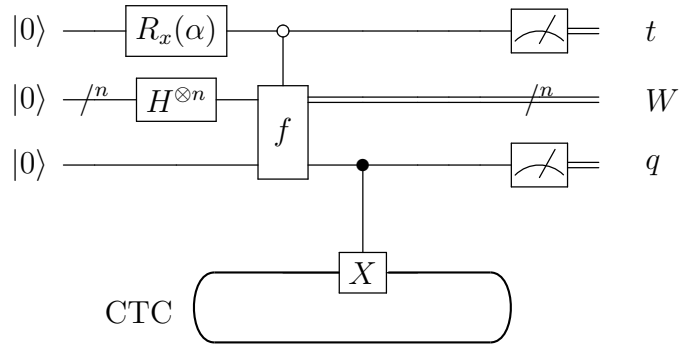


Figure 4.2: Alternative PCTC circuit for SAT.

The circuit depicted in Figure 4.2 has a lower error rate that can be further improved for similar cost with probability amplification. The first difference to the original is the removal of gate g which is required to set W to all zeros for nonsatisfiable SAT instances. Secondly, the Hadamard gate that is applied to the qubit t is replaced with the gate $R_x(\alpha)$ that rotates its qubit α degrees about the x axis and it has the unitary matrix

$$R_x(\alpha) = \begin{bmatrix} \cos\frac{\alpha}{2} & -i\sin\frac{\alpha}{2} \\ -i\sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{bmatrix},$$

$t = |0\rangle$ just before the α rotation.

As an example, applying the gate for $\alpha = \frac{\pi}{15}$ to rotate t:

$$t = R_x\left(\frac{\pi}{15}\right)|0\rangle = \begin{bmatrix} \cos\frac{\alpha}{2} & -i\sin\frac{\alpha}{2} \\ -i\sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\frac{\pi}{30} \\ -i\sin\frac{\pi}{30} \end{bmatrix},$$

$$t \approx 0.99\,|0\rangle - 0.1i\,|1\rangle\,,$$

with t applied to f as a control bit, f runs with probability $|0.99|^2 \approx 0.98$.

In order to inspect the algorithm, consider the three cases based on θ and t.

(i) θ is satisfiable.

- f runs with probability $\left|\cos\frac{\alpha}{2}\right|^2$ $(t = 0)$;
  f outputs a satisfying W without altering its last qubit $|0\rangle$. All the non satisfying W outputs are discarded via postselection.
- f does not run with probability $\left|-i\sin\frac{\alpha}{2}\right|^2$ $(t = 1)$;
  t and q are measured 1 and 0 respectively while W is a uniformly random bit string $\in \{0,1\}^n$

(ii) θ is unsatisfiable.

- f runs with probability $\left|\cos\frac{\alpha}{2}\right|^2$ $(t = 0)$;
  This outcome is discarded because no $W \in \{0,1\}^n$ satisfy θ, causing $q = 1$ with probability 1.
- f does not run with probability $\left|-i\sin\frac{\alpha}{2}\right|^2$ $(t = 1)$;
  t and q are measured 1 and 0 respectively while W is a uniformly random bit string $\in \{0,1\}^n$

At the end of the execution, if t is measured 1, θ is deemed unsatisfiable. If $t = 0$, θ is satisfiable. Unsatisfiable instances are detected without error. Error rate for satisfiable θ is equivalent to

$$\varepsilon = 1 - \left|\cos\frac{\alpha}{2}\right|^2 = \left|-i\sin\frac{\alpha}{2}\right|^2,$$

$$= \left|\sin\frac{\alpha}{2}\right|^2.$$

Assuming $\alpha \geq 0$,

$$\varepsilon = \sin^2\frac{\alpha}{2}.$$

Classical analogue for the $R_x(\alpha)$ is bounded by efficient computability. Assuming the smallest probability that can be generated using a fair coin in polynomial time with input size n and a constant c is bounded by

$$\frac{1}{2^{cn}}.$$

In this context, we use $R_x(\alpha)$ for probability generation so $R_x(\frac{\pi}{2})$ is equivalent to a single fair coin toss for generating small probabilities because

$$\sin^2\frac{\pi}{4} = \frac{1}{2} \quad \text{and} \quad \sin\frac{\pi}{4} = \frac{1}{\sqrt{2}}.$$

For generating probabilities equivalent to the power of cn coins, rotation gate's angle needs to be

$$\alpha = \arcsin\frac{1}{\sqrt{2^{cn}}}.$$

4.3.1. Probability Amplification

Beyond error rate reduction with usage of small α values, the performance of Algorithm 4.2 can be further improved using probability amplification as follows. Inputs of the $R_x(\alpha)$ gates are replicated s many times as depicted in the circuit in Figure 4.3. The gate C is essentially a multi controlled-NOT gate and it flips the value of the t qubit if all of $R_x(\alpha)$ gate outputs are 1, otherwise leaving it 0. More formally, the only transformation C does is:

$$\underbrace{|0\rangle}_{t}\underbrace{|11\ldots1\rangle}_{s} \rightarrow |1\rangle\,|11\ldots1\rangle$$

t is left unchanged for any other configuration.

Selecting an α value equivalent to $\arcsin\left(\frac{1}{\sqrt{m}}\right)$ yields $\frac{1}{m}$ probability for each gate to output 1. $R_x(\alpha)$ used here should be easily achievable because, in classical terms, it only requires $\log m$ fair coin tosses to reach the same probability, which is much lower than the theoretical limit. Using s many inputs and $R_x(\alpha)$ gates, C only flips t with probability $\frac{1}{m^s}$.

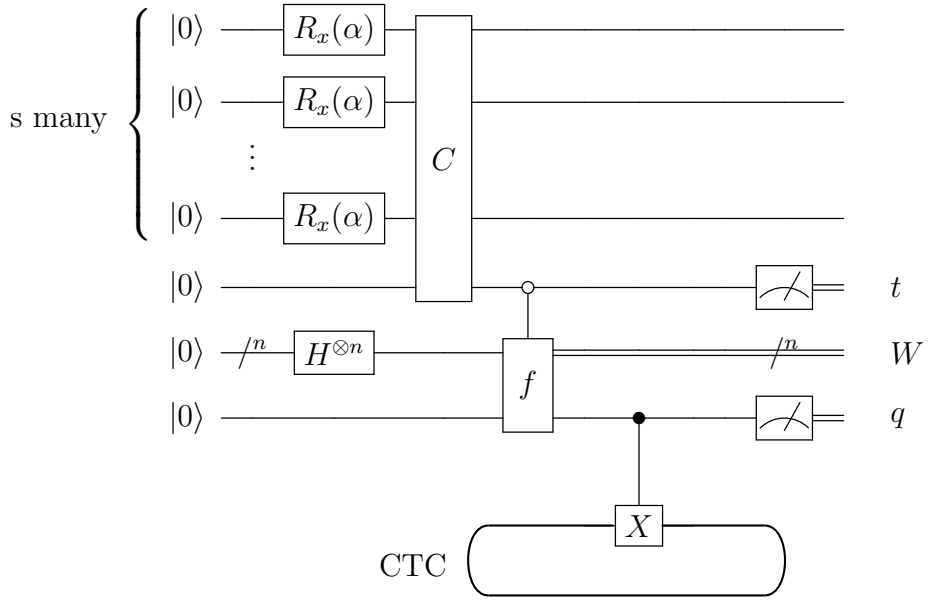


Figure 4.3: Alternative PCTC circuit for SAT with probability amplification.

4.4. Performance Comparison

In order to compare the performances of the two SAT algorithms in Figures 4.1 and 4.3, we assume the following circuit costs.

- Single qubit gate cost : 1
- Multi(n) qubit gate cost : n
- Controlled gate cost : gate's cost + 1

BW's algorithm, Algorithm 4.1 with probability amplification that uses s many repetitions has total cost λ_{BW},

- s many 1 cost H gates
- s many n cost $H^{\otimes n}$ gates
- s many $n+2$ cost f gates
- s many $n+2$ cost g gates
- One s cost quantum analogue of an AND gate
- One s cost quantum analogue of an OR gate
- One CNOT gate with cost 2

$$\lambda_{BW} = s + sn + 2s(n+2) + 2s + 2 = 3sn + 7s + 2.$$

Alternative version, Algorithm 4.3 (in short, A) with probability amplification has total cost λ_A,

- s many 1 cost $R_x(\alpha)$
- n many 1 cost H gates
- One $n+2$ cost f gate
- One $s+2$ cost C gate
- One CNOT gate with cost 2

$$\lambda_A = s + n + (n+2) + (s+2) + 2 = 2n + 2s + 6.$$

Recall that Algorithm 4.1 has error rate $\varepsilon_{BW} = \frac{1}{m+1}$ and the alternative version has $\varepsilon_A = \sin^2 \frac{\alpha}{2}$ without amplification. Error rates with s repetitions and using $\alpha = \arcsin \frac{1}{\sqrt{m}}$ are,

$$\varepsilon_{BW} = \frac{1}{(m+1)^s},$$

$$\varepsilon_A = \frac{1}{m^s}.$$

Dividing the reciprocal of error rates by the added cost of amplification yield the total error reduction/total cost functions $\mathcal{M}$,

$$\mathcal{M}_{BW} = \frac{\frac{1}{\varepsilon_{BW}}}{\lambda_{BW}} = \frac{(m+1)^s}{3sn + 7s + 2},$$

$$\mathcal{M}_A = \frac{\frac{1}{\varepsilon_A}}{\lambda_A} = \frac{m^s}{2n + 2s + 6}.$$

Dividing the metrics for comparing growth rates as n increases yield,

$$\frac{\mathcal{M}_A}{\mathcal{M}_{BW}} = \frac{m^s(3sn + 7s + 2)}{(m+1)^s(2n + 2s + 6)},$$

$$\lim_{n\to\infty} \frac{m^s(3sn + 7s + 2)}{(m+1)^s(2n + 2s + 6)} = \frac{3s}{2}\left(\frac{m}{m+1}\right)^s.$$

For large $m \approx n$ and a small constant s,

$$\lim_{n\to\infty} \frac{\mathcal{M}_A}{\mathcal{M}_{BW}} > 1,$$

$$\mathcal{M}_A > \mathcal{M}_{BW}.$$

which means $\mathcal{M}_A$ requires less circuit size for the same reduction in error rate, or that it yields less error for the same circuit size. Hence, the Algorithm 4.3 is more cost efficient.

4.4.1. Performance Evaluation

Performance metrics, $\mathcal{M}_A$ and $\mathcal{M}_{BW}$, are compared for three different m values and a constant $s = 10$. For the following plots, $\mathcal{M}$ (y-axis) is the performance metric and n (x-axis) is the number of variables in the SAT instance.

The first case considers SAT instances with relatively low amount of satisfying assignments, $m = \log n$ with $s = 10$ repetitions for amplification. Figure 4.4 shows the plots of the two performance metrics for this case. It can be observed that $\mathcal{M}_A$ grows much faster than $\mathcal{M}_{BW}$ until $n \approx 2000$ after which the growth rates are observed to be roughly the same.

$$\mathcal{M}_A = \frac{(\log n)^{10}}{2n + 26},$$

$$\mathcal{M}_{BW} = \frac{(\log n + 1)^{10}}{30n + 72}.$$

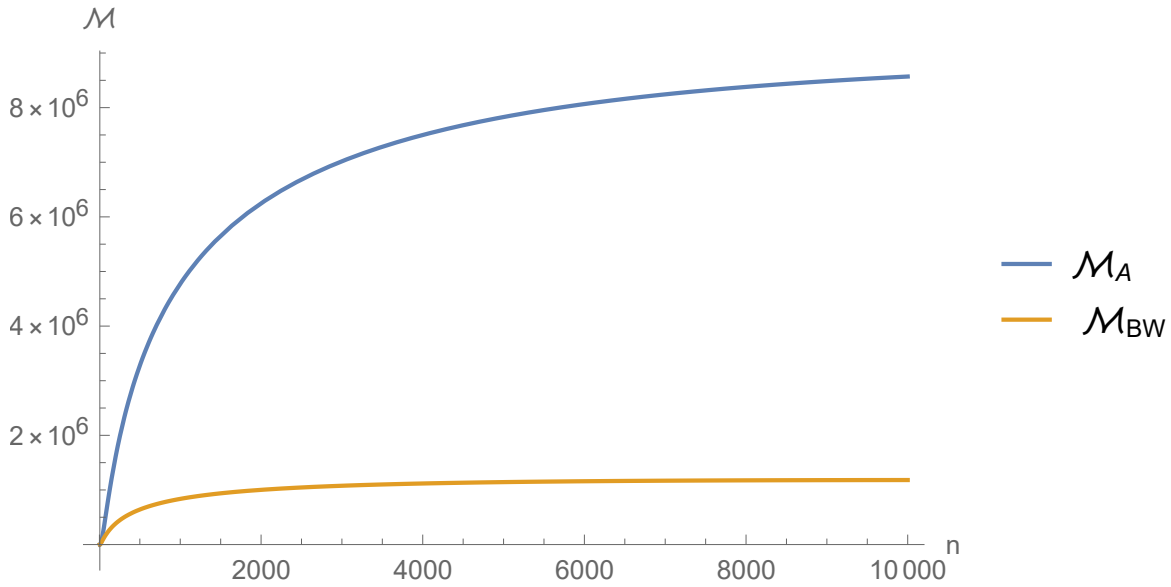


Figure 4.4: Error reduction per added circuit cost comparison for Algorithms 4.1 and 4.3. $m = \log n$, $s = 10$.

Second case considers SAT instances with $m = n$ and $s = 10$ repetitions for amplification. Figure 4.5 shows the plots of the two performance metrics for this case.

$$\mathcal{M}_A = \frac{n^{10}}{2n + 26},$$

$$\mathcal{M}_{BW} = \frac{(n+1)^{10}}{30n + 72}.$$

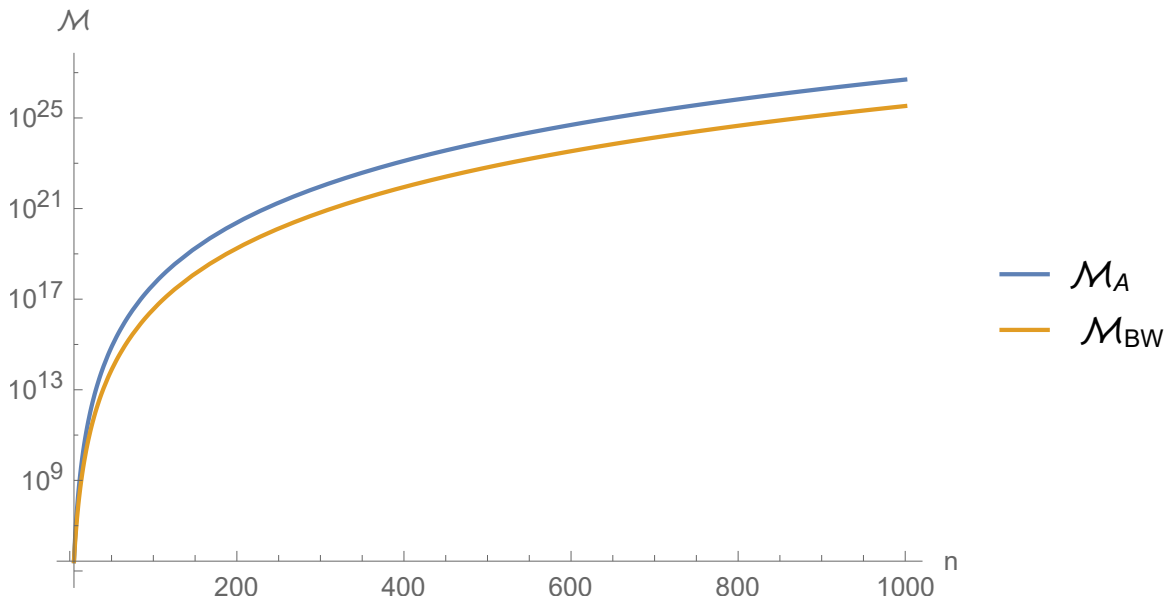


Figure 4.5: Error reduction per added circuit cost comparison for Algorithms 4.1 and 4.3. $\mathcal{M}$ axis with logarithmic scaling. $m = n$, $s = 10$.

Lastly, an SAT instance with half of all of its possible assignments satisfying its equation is considered in Figure 4.6.

$$\mathcal{M}_A = \frac{\left(2^{(n-1)}\right)^{10}}{2n+26},$$

$$\mathcal{M}_{BW} = \frac{\left(2^{(n-1)}+1\right)^{10}}{30n+72}.$$

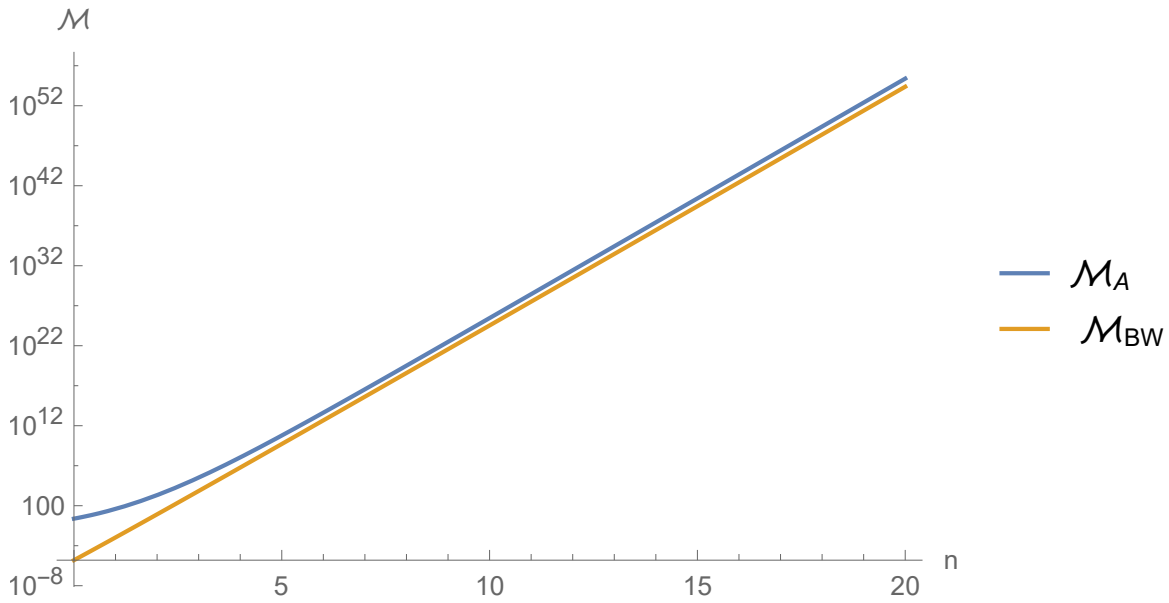


Figure 4.6: Error reduction per added circuit cost comparison for Algorithms 4.1 and 4.3. $\mathcal{M}$ axis with logarithmic scaling. $m = 2^{n-1}$, $s = 10$.

5. CHAIN-CTC MODEL

Chain-CTC model uses short DCTCs of fixed length to send a piece of information back in time. This model has the modularity advantage as well as being possibly more practical and "easier" to construct [9]. The model uses a single DCTC of length h repeatedly to send a piece of information given to it at any time t to time $t - h$. Reusing the DCTC available, the computer sends the information to a time between $t - h$ and $t - 2h$ depending on the time spent between receiving and sending it.

5.1. Primality Testing with Chain-CTCs

We examine how the Chain-CTC model works on the chained version of Algorithm 1 for primality testing in Algorithm 3 given below. For a given number n with k many factors, if at the and of its execution $b = 0$ then the number given is deemed prime, if $b = 1$ the number is deemed composite.

Algorithm 3

read c_1

print c_1

read c_2

$c_1 \leftarrow c_2$

send c_1

read c_3

$c_2 \leftarrow c_3$

send c_2

$b \leftarrow c_3$

generate a random integer $i \in \{2, 3, \ldots, n-1\}$

if $n \mod i == 0$ **then** set $b = 1$

with probability 2^{-n}, set $b = 0$

send $c_3 = b$

Figure 5.1: Primality testing with Chain-CTCs.

It is relatively easier to analyze Chain-CTCs starting with the last one, because previous ones often are dependent on the last. The transition matrix of c_3 denoted by C_3 is equal to the transition matrix of the DCTC in algorithm 1.
If n is prime,

$$C_3 = \begin{bmatrix} 1 & 2^{-n} \\ 0 & 1 - 2^{-n} \end{bmatrix}, \qquad \phi_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

C_1 and C_2 apply a transition that sets the value of their bits to the value of c_3. This transition matrix corresponds to a matrix that is formed of columns that are equivalent to ϕ_3.

$$C_1 = C_2 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \phi_1 = \phi_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

If n is composite,

$$C_3 = \begin{bmatrix} 1 - \frac{k}{n} & 2^{-n} \\ \frac{k}{n} & 1 - 2^{-n} \end{bmatrix},$$

$$\phi_3 = \begin{bmatrix} \frac{2^{-n}}{\frac{k}{n} + 2^{-n}} \\ \frac{\frac{k}{n}}{\frac{k}{n} + 2^{-n}} \end{bmatrix},$$

$$C_1 = C_2 = \begin{bmatrix} \frac{2^{-n}}{\frac{k}{n} + 2^{-n}} & \frac{2^{-n}}{\frac{k}{n} + 2^{-n}} \\ \frac{\frac{k}{n}}{\frac{k}{n} + 2^{-n}} & \frac{\frac{k}{n}}{\frac{k}{n} + 2^{-n}} \end{bmatrix}, \qquad \phi_1 = \phi_2 = \begin{bmatrix} \frac{2^{-n}}{\frac{k}{n} + 2^{-n}} \\ \frac{\frac{k}{n}}{\frac{k}{n} + 2^{-n}} \end{bmatrix}.$$

The algorithm identifies prime numbers with certainty. If a number is composite it is wrongfully identified as a prime with probability

$$\frac{2^{-n}}{\frac{k}{n} + 2^{-n}},$$

which tends to 0 exponentially as n increases.

5.2. Chain-CTCs with multiple branchings

The transitions in Chain-CTCs that have more than one computational branching that occur between CTC's endpoints can be modelled just like DCTC's because the underlying model is the same. The additional "tunneling" CTC's where there is only a single computational branch are easy to model as well because the columns in their transition matrices are equivalent to the stationary distribution of the CTC where the actual computation occurs.

Using Chain-CTCs with algorithms that have multiple branchings on different times is more interesting. Chaining multiple DCTCs get increasingly complicated as number of branchings increase as demonstrated below.

Algorithm 4

read ctc_1
$c_1 \leftarrow FlipCoin$
read ctc_2
$ctc_1 \leftarrow ctc_2$
send ctc_1
$c_2 \leftarrow FlipCoin$
read ctc_3
$ctc_2 \leftarrow ctc_3$
send ctc_2
$c_3 \leftarrow FlipCoin$
if $c_1 == c_2 == c_3 == 1$ **then**
| send $ctc_3 = 1$
else
| send $ctc_3 = 0$

Figure 5.2: Chain-CTC algorithm with multiple branchings.

In order to construct C_3 we need to analyze computation branches that correspond to different coin flip outcomes. Let C_m^{ijk} be the transition matrix that corresponds to the transition that is applied to CTC C_m if the current branch coin flip outcomes are $c_1 = i, c_2 = j, c_3 = k$. We start analysis by considering C_3 first. If at least one of c_1 or c_2 is 0, transitions are all the same because they are not dependent on the new c_3' value.

$$C_3^{00k} = C_3^{01k} = C_3^{10k} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix},$$

with corresponding stationary distributions

$$\phi_3^{00k} = \phi_3^{01k} = \phi_3^{10k} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

If $c_1 = c_2 = 1$, received value of the CTC bit is solely dependent on c_3',

$$C_3^{11k} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

$$\phi_3^{11k} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

All of the C_3^{ijk} transitions above with k as a random variable, occur with equal probability $\frac{1}{4}$. Combining their stationary distributions yield ϕ_3,

$$\phi_3 = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{7}{8} \\ \frac{1}{8} \end{bmatrix}.$$

C_2 is analyzed similarly however in this case c_2 is also subject to change in addition to c_3. If $c_1 = 0$, we receive 0 from ctc_3 with probability 1 regardless of c_2' or c_3',

$$C_2^{0jk} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \phi_2^{0jk} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

C_2 only sends 1 if both $j = k = 1$ which occurs with probability $\frac{1}{4}$ so we have,

$$C_2^{1jk} = \begin{bmatrix} \frac{3}{4} & \frac{3}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \qquad \phi_2^{1jk} = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix}.$$

Combining these two stationary distributions of equiprobability yield,

$$\phi_2 = \frac{1}{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{7}{8} \\ \frac{1}{8} \end{bmatrix}.$$

In order to construct C_1, first we consider the case $c_1 = 0$ where $ctc_1 = 0$. Hence we have

$$C_1^{0jk} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \phi_1^{0jk} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

If $c_1 = 1$, we receive $ctc_1 = 1$ with probability $\frac{1}{4}$

$$C_1^{1jk} = \begin{bmatrix} \frac{3}{4} & \frac{3}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \qquad \phi_1^{1jk} = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix},$$

$$\phi_1 = \frac{1}{2}\phi_1^{0jk} + \frac{1}{2}\phi_1^{1jk} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \frac{3}{4} & \frac{3}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix},$$

$$\phi_1 = \begin{bmatrix} \frac{7}{8} \\ \frac{1}{8} \end{bmatrix}.$$

Overall we have obtained,

$$\phi_1 = \phi_2 = \phi_3 = \begin{bmatrix} \frac{7}{8} \\ \frac{1}{8} \end{bmatrix}.$$

Algorithm 4's three CTCs have different branchings depending on the coin flips among them however they all have the same stationary distribution.

6. RANDOMIZED COMPUTATIONS WITH CTCS

Algorithm 5

read r

$c \leftarrow$ Flip Coin

if $r == 0$ **then**
| send 1

if $r == 1$ **then**
| send c

print r

Figure 6.1: Simple DCTC algorithm with a coin flip inside its CTC.

The CTC evolution in Algorithm 5 can be represented with the transition matrix

$$\begin{bmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{bmatrix},$$

which has the stationary distribution

$$\phi_r = \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \end{bmatrix}.$$

Hence the algorithm prints 0 with probability $\frac{1}{3}$ and 1 with probability $\frac{2}{3}$.

Algorithm 6

$c \leftarrow$ Flip Coin

read b

if $b == 0$ **then**
| send 1

if $b == 1$ **then**
| send c

print b

Figure 6.2: Simple DCTC algorithm with a coin flip outside its CTC.

Algorithm 6 is almost the same as the previous one except the coin flip step is moved out of the CTC loop.

Starting with the case $c = 0$, the transitions in the CTC becomes deterministic

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which has the stationary distribution

$$\phi_b^0 = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix},$$

hence the algorithm prints 0 and 1 with probability $\frac{1}{2}$ if $c == 0$.

If the coin flip results in $c = 1$, the transition matrix becomes

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix},$$

with its corresponding stationary distribution

$$\phi_b^1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and the algorithm prints 1 with probability 1.

Combining the two initial fair coin flips results we get $\phi_b = \frac{1}{2}\phi_b^0 + \frac{1}{2}\phi_b^1 = \begin{bmatrix} \frac{1}{4} \\ \frac{3}{4} \end{bmatrix}$.

$$\frac{1}{2} \cdot \frac{1}{2} + 0 = \frac{1}{4} \quad \text{chance to print 0,}$$

$$\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot 1 = \frac{3}{4} \quad \text{chance to print 1.}$$

Hence, the Algorithm 5 prints 0 with probability $\frac{1}{3}$ whereas Algorithm 6 has probability $\frac{1}{4}$ to print 0 using fair coins.

If the coin is biased with $P(0) = \frac{1}{3}$, Algorithm 5 have probabilities of printing 0 and 1, $\frac{1}{4}$ and $\frac{3}{4}$ respectively whereas its $\frac{1}{6}$ and $\frac{5}{6}$ for the Algorithm 6, interestingly. The difference in their results get even larger as a more biased coin is used. This behaviour is generalized to a biased coin with $P(0) = \frac{1}{n}$ as

$$\phi_r = \begin{bmatrix} \frac{1}{n+1} \\ \frac{n}{n+1} \end{bmatrix}, \qquad \phi_b = \begin{bmatrix} \frac{1}{2n} \\ \frac{2n-1}{2n} \end{bmatrix}.$$

Despite doing seemingly the same operation and transitions, here we have two algorithms that produce two different outcomes. This shows that even if quantum gates were implemented perfectly with no errors, designing algorithms to work as intended with CTCs might require extra attention. Regarding this atypical behaviour, Aaronson and Watrous discussed how two quantum operations that are arbitrarily close can have drastically different fixed points [8].

6.1. Chain-CTC Examples

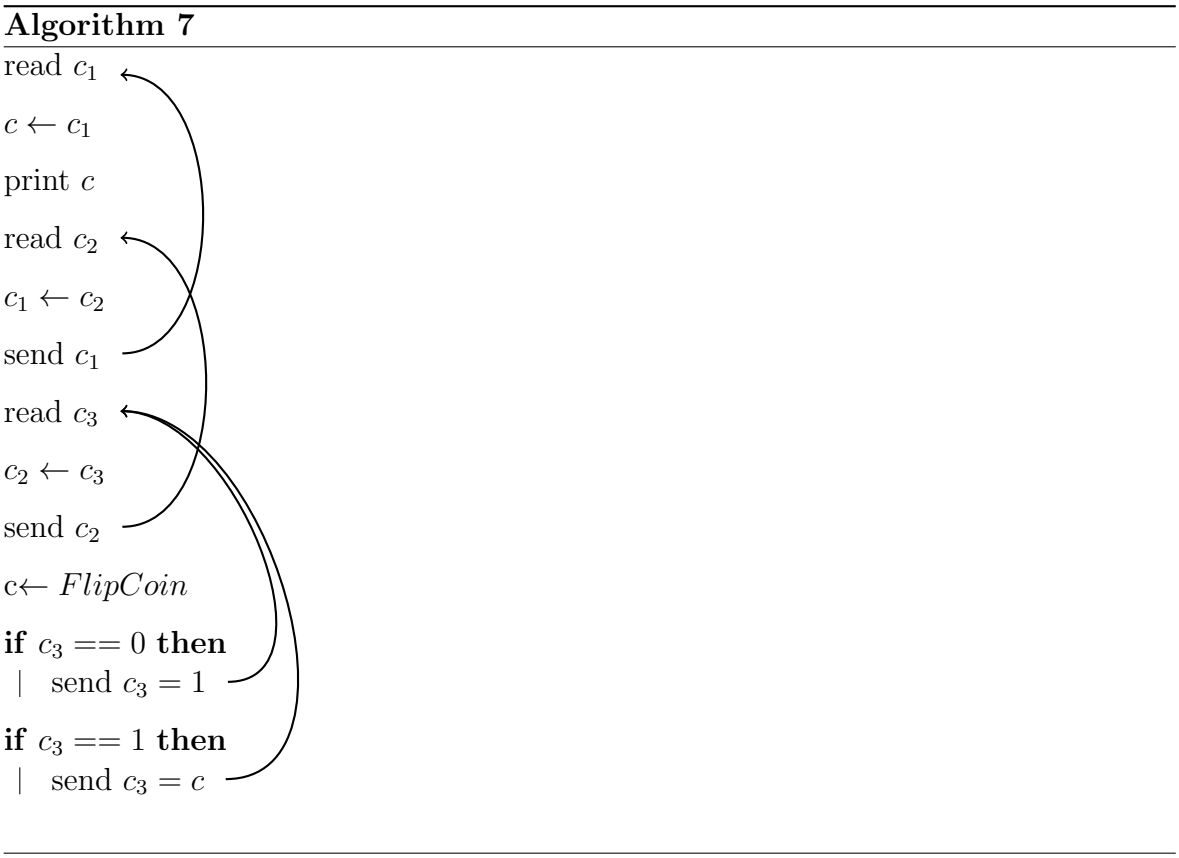


Figure 6.3: Chained version of Algorithm 5.

The analysis of Algorithm 7's CTC's transition matrix is fairly straightforward. As C_3 does a simple transition same as the only CTC in Algorithm 5.

$$C_3 = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{bmatrix}, \quad \phi_3 = \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \end{bmatrix}.$$

Because we have $C_1 = C_2$ as tunneling CTCs, they simply copy the distribution that comes from C_3 to their CTC bits hence we have

$$C_1 = C_2 = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}, \qquad \phi_1 = \phi_2 = \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \end{bmatrix}.$$

Algorithm 8

$c_0 \leftarrow FlipCoin$
read c_1
$c_1 \leftarrow FlipCoin$
read c_2
$c_1 \leftarrow c_2$
send c_1
$c_2 \leftarrow FlipCoin$
read c_3
$c_2 \leftarrow c_3$
send c_2
$c_3 \leftarrow FlipCoin$
if $c_0 == c_1 == c_2 == c_3 == 1$ **then**
| send $c_3 = 1$
else
| send $c_3 = 0$

Figure 6.4: Chained CTC algorithm example.

Algorithm 8 is analyzed by considering c_0's outcomes separately.

(i) if $c_0 == 0$,

c_3 is received 0 and sent 0 with probability 1, regardless of other coins' results.

$$C_3^{0jkl} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad \phi_1^{0jkl} = \phi_2^{0jkl} = \phi_3^{0jkl} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

(ii) if $c_0 == 1$,

The algorithm branches just like Algorithm 7 which has stationary distributions corresponding to

$$\phi_1^{1jkl} = \phi_2^{1jkl} = \phi_3^{1jkl} = \begin{bmatrix} \frac{7}{8} \\ \frac{1}{8} \end{bmatrix}.$$

Combining the stationary distributions for both c_0 outcomes of equiprobability yields

$$\phi_1 = \phi_2 = \phi_3 = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \frac{7}{8} \\ \frac{1}{8} \end{bmatrix} = \begin{bmatrix} \frac{15}{16} \\ \frac{1}{16} \end{bmatrix}.$$

This result, which is in line with expectations of four classical coin flips, suggests the odd behaviour in Algorithms 5 and 6 are most likely rare and require specific transitions and branchings concurrently to be observed.

7. CONCLUSION

In this thesis, we have studied the powerful models of computation in the presence of Closed Timelike Curves. We have demonstrated their computation power by showing how they can solve a Boolean Satisfiability problem efficiently. We examined how DCTC computation can result in atypical results under some circumstances with randomization. We have proposed an alternative version to Brun and Wilde's postselection based Boolean Satisfiability algorithm. We compared our proposed algorithm against BW's for various SAT instance types which shows that our proposed version is more cost efficient per same error reduction. Lastly, we have investigated the Chain-CTC model which streamlines the utilization of DCTCs and analyzed how they work with various branchings of randomized algorithms.

REFERENCES

1. Deutsch, D., "Quantum mechanics near closed timelike lines", *Physical Review D*, Vol. 44, No. 10, p. 3197, 1991.

2. Brun, T. A., "Computers with closed timelike curves can solve hard problems efficiently", *Foundations of Physics Letters*, Vol. 16, No. 3, pp. 245–253, 2003.

3. Bacon, D., "Quantum computational complexity in the presence of closed timelike curves", *Physical Review A*, Vol. 70, No. 3, p. 032309, 2004.

4. Aaronson, S., M. Bavarian and G. Gueltrini, "Computability Theory of Closed Timelike Curves", *arXiv preprint arXiv:1609.05507*, 2016.

5. Lloyd, S., L. Maccone, R. Garcia-Patron, V. Giovannetti, Y. Shikano, S. Pirandola, L. A. Rozema, A. Darabi, Y. Soudagar, L. K. Shalm and A. M. Steinberg, "Closed timelike curves via postselection: theory and experimental test of consistency", *Physical review letters*, Vol. 106, No. 4, p. 040403, 2011.

6. Brun, T. and M. M. Wilde, "Perfect state distinguishability and computational speedups with postselected closed timelike curves", *Foundations of Physics*, Vol. 42, No. 3, pp. 341–361, 2012.

7. Allen, J., "Treating time travel quantum mechanically", *Phys. Rev.*, Vol. A90, No. 4, p. 042107, 2014.

8. Aaronson, S. and J. Watrous, "Closed timelike curves make quantum and classical computing equivalent", *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Vol. 465, pp. 631–647, The Royal Society, 2009.

9. Say, A. C. C. and A. Yakaryılmaz, "Computation with multiple CTCs of fixed

length and width", *Natural Computing*, Vol. 11, No. 4, pp. 579–594, 2012.

10. O'Donnell, R. and A. C. C. Say, "One time-traveling bit is as good as logarithmically many", *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 29, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

11. O'Donnell, R. and A. C. C. Say, "The Weakness of CTC Qubits and the Power of Approximate Counting", *ACM Trans. Comput. Theory*, Vol. 10, No. 2, pp. 5:1–5:22, May 2018, `http://doi.acm.org/10.1145/3196832`.

12. Nielsen, M. A. and I. Chuang, *Quantum Computation and Quantum Information*, AAPT, 2002.