Quantum Speed-ups for Boolean Satisfiability and Derivative-Free Optimization

by

Srinivasan Arunachalam

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization - Quantum Information

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Srinivasan Arunachalam

Abstract

In this thesis, we have considered two important problems, *Boolean satisfiability (SAT)* and *derivative free optimization* in the context of large scale quantum computers. In the first part, we survey well known classical techniques for solving satisfiability. We compute the approximate time it would take to solve SAT instances using quantum techniques and compare it with state-of-the-art classical heuristics employed annually in SAT competitions. In the second part of the thesis, we consider a few classically well known algorithms for derivative free optimization which are ubiquitously employed in engineering problems. We propose a quantum speedup to this classical algorithm by using techniques of the *quantum minimum finding* algorithm. In the third part of the thesis, we consider practical applications in the fields of *bio-informatics*, *petroleum refineries* and *civil engineering* which involve solving either satisfiability or derivative free optimization. We investigate if using known quantum techniques to speedup these algorithms directly translate to the benefit of industries which invest in technology to solve these problems. In the last section, we propose a few open problems which we feel are immediate hurdles, either from an algorithmic or architecture perspective to getting a convincing speedup for the practical problems considered.

Acknowledgements

First and foremost, I would like to thank God for without Him any of this would not be possible. All praises and accolades go to Him.

I am grateful to my supervisor, Michele Mosca for having faith in me and accepting me first as an undergraduate research assistant and then a graduate student. His patience and support has been invaluable during my studies in Waterloo. I am also grateful to my thesis readers Thomas Coleman and William Cook for agreeing to be part of my committee and providing helpful feedback.

I do not think much work in this thesis would have been done if not for the help of Shitikanth. He has been a great friend, mentor, go-to guy with any doubts, and has motivated me towards the field of computer science. I would also like to thank Vincent Russo for reviewing and giving me useful comments on this thesis, and overall improving my writing skills. The time we have spent talking of the Queen is unforgettable. I do not think I would have understood and completed the section on Oil field optimization if not for the constant clarifications provided by Obiajulu Joseph Isebor. I also thank Shaowei Cai for helping me understand how parameters used for SAT solvers in SAT competitions are different from how algorithmic complexity in papers work. I would like to thank Kasra Bigdeli, William Cook, David Gosset, Warren Hare, Robin Kothari and Henry Wolkowicz for many email exchanges and insightful discussions. And finally my room mates and friends Akash, Swati, Preetham, Ankita, Mani, Guru, Rana, Ananya for sharing all the fun moments in Waterloo and being there for me when I required them. After hearing me rant, I am sure few of them understand NP-hardness and the classical PCP theorem better than me. I wish everyone of them best of luck in their respective endeavors.

Although mentioned last, I am not even sure if I can even place my parents anywhere for their invaluable support. Their unconditional love and encouragement throughout my life has made me whom I am today.

To my parents

I'll come home soon!

Contents

List of Tables

List of Figures

Preface

The problems I have worked on throughout the extent of my graduate studies vary from theoretically oriented semidefinite programming to complexity theory to practically relevant problems such as quantum random access memory architecture for practical problems. In [AMR13], we consider quantum correlations in two-round prover-verifier interactions illustrating how quantum players could use correlations which exhibit strictly non-classical behavior to their advantage. These correlations manifested in an ability of the player to make use of a form of hedging, where the risk of losing the first game was eliminated by offsetting that risk in a subsequent game. The entire game can be constructed as a semidefinite program to observe the hedging behavior. I contributed to another work in [AGJO$^+$], where we consider the quantum random access architecture proposed in [GLM08]. We analyze the physical implementation of the original Bucket-Brigade architecture and propose a circuit model for the same. A realistic error model was constructed based on the physical implementation in order to understand the need for error-correction to perform Grover's search algorithm faithfully using this architecture. The need for quantum access to read classical memory is essential for many practical problems highlighted in the last chapter.

In this thesis, I have included my work on Boolean satisfiability and derivative free optimization. Extensive research has been done in the areas of practical relevance of quantum cryptography and advanced sensing. However, we have considered a few practical applications of these optimization problems relevant to other industries. We review these important optimization algorithms and analyze how the quantum speed-ups to these optimization algorithms compare with state-of-the-art-classical techniques employed by the industries. Often, with algorithmic improvements proposed in the literature, it is generally assumed, that the speedup should follow in a straightforward manner for industrially relevant problems relying on these algorithms. We attempt to study this question and elucidate the hurdles to the realization of such algorithmic speed-ups when considering important problems. We hope this thesis will serve as a template for analyzing various algorithms in the future to compare their performance on industrially relevant parameters. Towards the end, we have highlighted various problems which would be interesting to consider from the perspective of complexity and practical applications of quantum algorithms.

Chapter 1

Introduction

Quantum computing has been growing for over two decades since the celebrated result of Shor's quantum algorithm to factor numbers in polynomial time which is believed to be a computationally hard problem in the classical setting. There have been many quantum algorithms since, which have demonstrated either a polynomial or exponential separation between the classical and quantum setting such as searching [Gro97], collision finding [AS04], triangle finding [MSS07], element distinctness [Amb07], graph theoretic algorithms [DHHM04], computational geometry [SST02], string matching [RV03] etc. Majority of the speedups in these algorithms involve either quantization of a random walk or Grover's search algorithm which provides a quadratic speedup for searching. Another area where quantum techniques are expected to provide speed-ups to classical algorithms is in the area of optimization. In this thesis, we consider a few important families of optimization problems, which are essential for industries in solving practical applications. We have considered the *Boolean satisfiability* problem as well as *derivative free optimization* and analyzed how quantum algorithmic speedups for these problems compare with state-of-the-art classical algorithms. We hope the studies carried out in this thesis will spur interest in connecting the bridge between theoretically preferred algorithms and practical employed techniques in the industry.

The *satisfiability problem* has been of interest to people in computer science primarily due to the Cook-Levin theorem ([Coo71], [Lev73]), which states that the Boolean satisfiability problem is NP-complete (an abbreviation for non-deterministic polynomial). It also holds the honor of being the first well-defined NP-complete problem. In this regard, any problem which is NP-complete can be reduced to the Boolean satisfiability problem in polynomial time. Assuming P$\neq$NP, it is safe to consider that there exists no polynomial time algorithm for SAT, motivating researchers to devise efficient and scalable algorithms when the size of SAT instances are intractable. Quantum techniques particularly come into the framework when facing such classically intractable problems,

where employing quantum resources and algorithms can solve the problems in realistic time. There have been many interesting results related to the satisfiability problem in the quantum setting as well in the areas of complexity theory[GN13], adiabatic algorithms [Hog03], multi-prover interactive systems [CD10], etc.

Research in this classical NP-complete problem has not stopped just with the theoretical computer science community, but also extended to areas of engineering sciences where faster algorithms for satisfiability would give solutions to practical problems. A few such areas include bio-informatics [LMS06], system modelling and software checking [CBRZ01], combinational equivalence checking [Bra03], circuit design and delay computation [MSS00] [SBSV96], crosstalk noise prediction in integrated circuits [CK99], AI Planning [SKC+93], etc. A faster algorithm for SAT from the theoretical perspective would consequently result in the speedup of solving these important computationally hard problems. In this thesis, we have explicitly analyzed the quantum speedups in the satisfiability problems and compared them to the best known classical heuristics. Although, we do not have quantum computers to test our quantum algorithms, the algorithms we consider have been classically rigorously analyzed, and hence predicting the time it would take for quantum computers to solve similar instances can be approximated well.

Optimization problems arise in various applications when the goal is to find a set of optimum design parameters that maximize or minimize an objective function which measures the merit of the design. Typically, accompanying the objective function and design parameters are a set of constraints which bound the values of these parameters. We have solved minimization or maximization problems before with the recipe of taking the derivatives of the objective function, finding the critical points and checking the second derivative to characterize the critical points. The class of bound-constrained optimization problems we are concerned with in this thesis is referred to as *derivative free optimization*. These problems are characterized by objective functions which are *computational expensive to evaluate* and problems for which *derivatives of the objective function are neither symbolically nor numerically available and only the function values of* $f(x)$ *are available*. Majority of the problems motivating the interest in this technique arise from solving complex engineering problems whose objective functions depend on complex simulations based on equations which define the underlying physical process.

Derivative free optimization is an area with a long history proposed in the 1960's with the Hooke and Jeeves algorithm [HJ61]. Simultaneously along with the developments of algorithms, there have been many software implementations for these algorithms that are used in software packages for optimization such as LGO, MCS, SID-PSM, SNOBFIT, DFO, HOPSACK [RS12]. The renewed interest and increase in papers employing these techniques for optimization in the last decade is due to the proof of global and local convergence of *generalized pattern search* algorithms to the optimal solution in [Tor97], [DLT03] and convergence analysis of *mesh adaptive direct search* algorithms by Dennis and Audet in [ADJ06]. The increasing research

and popularity in this technique of optimization is also due to its applications in diverse engineering problems such as molecular geometry [ANRV04], aeroacoustic [MWDJM04], oil refinery optimization [Ise09], ground water community problem [FRK+08], hydrodynamic design [DV04], helicopter blade design [BDJF+99], installing dampers between adjacent structures [Big12], etc.

Recently, there has been curiosity in the work of D-Wave machine which claims to solve hard optimization problems by annealing type techniques. It may be worth clarifying the connection between this work and the application of the D-Wave machine to solving optimization problems. The D-Wave device does not attempt to perform fault-tolerant quantum computation capable of implementing the broad range of known quantum algorithms. Rather, it is a special purpose device that attempts to speed up the solution of a special class of optimization problems by annealing type methods. There have been many recent articles studying the quantumness of these D-Wave machines [SS13], [SSSV14] and analyzing if the device does provide any speedup over classical computation [RWJ+14]. In this thesis, we consider the full power of a *fault-tolerant scalable quantum computer* when one is built.

One parameter in which the D-Wave machine seems to differ from the traditional fault-tolerant quantum computer is in employing heuristics for optimization problems. While heuristics have many important applications, there are several areas of practical interest in which accuracy of solutions is of prime importance. For example in cryptanalysis, in any protocol the secret key is generally obtained by sampling an astronomical number of random keys to find the key which deciphers the ciphertext. A strong symmetric cipher has the property that, even if a string differs in only one bit from the original decryption key, the string will essentially produce a random text when used for decryption. Thus there is no practical notion of obtaining a good approximation to a decryption key. Another area of importance is in computational biological problems such as protein sequencing, protein folding, cancer computational biology, etc. where precision of the solution is integral. Since the solutions to these problems are important in curing diseases and developing drugs, heuristics and approximation algorithms without guarantees of accuracy of the final solution are undesired. Another area where accuracy is important is in civil engineering. In this thesis we have considered the problem of optimization of inserting dampers between buildings to reduce the effect of pounding during earthquakes. With increasing heights of buildings and increasing investment in constructing sky scrapers, heuristic solutions could result in an sub-optimal solution which is not useful in multi-billion dollar projects. Apart from these problems, there are many more areas of computation, in which heuristics and approximation algorithms are undesired.

The contributions of this thesis can be summarized as follows, we have analyzed the satisfiability problem and compared it with the best known classical heuristics. We observe that in order for the quantum analogue of SAT to compete with the best known classical heuristics and scale faster we need a well defined notion of *directed quantum walks* which has not been defined

and analyzed in the literature. Although *derivative free optimization* has been studied well in the classical setting, the quantum speedup to the algorithm using the *quantum minimum finding algorithm* has not been studied before. We present a few results in this direction of improving the number of queries to the time-consuming black-box evaluations for derivative free optimization using quantum techniques. We have finally considered a few practical problems which rely on the techniques of solving the Boolean satisfiability and derivative free optimization and analyzed the impact of quantum speedup to these problems. For these practical problems, we consider parameters of interest in practice to analyze the trade-off between the ideal clock speeds of quantum computers versus classical computers. We outline challenges and relevant questions that will enable us to understand how to employ these quantum algorithms to solve important optimization problems integral to various industries.

1.1 Organization of Thesis

The thesis is organized as follows. In chapter 2 we provide the necessary preliminaries and background. In section 2.1, we will cover the query model of computation. In section 2.2 we will explain a few algorithms which will be essential in understanding the algorithms in the subsequent chapters. In section 2.2.1 we describe the *quantum minimum finding* algorithm based on the result of [DH96] by Høyer-Dürr and quantum amplitude amplification result in [BHMT00]. In section 2.3 we give a brief overview of random walks and the quantization of these random walks.

In chapter 3, we introduce the satisfiability problem and comment on the complexity of SAT, hardness in SAT instances, and the structure present in instances. In section 3.2, we give an overview of classical algorithms which form the backbone of the current state-of-the-art SAT solvers. In section 3.3, we discuss SAT solvers which are used ubiquitously by industries to solve problems using techniques discussed in section 3.2. In section 3.4, we consider the use of quantum techniques to speedup the algorithms discussed in section 3.3 and give numerical evidence of a quadratic speedup. In section 3.5 we consider the algorithms applied practically to industrially relevant instances and analyze why even with a quantum advantage it wouldn't necessarily imply all the instances considered by industries could be sped up.

In chapter 4 we introduce the theory of derivative free optimization. In section 4.1 we begin by giving a brief overview of optimization techniques which involve the use of gradients before surveying state-of-the-art derivative free algorithms which are used often in practical problems. In section 4.2 we discuss the computational cost of classical derivative free optimization and show how a quantum speedup could improve the algorithm.

In chapter 5 we discuss three practical problems which arise from the idea of satisfiability and derivative free optimization. In section 5.1 we discuss the problem of haplotype inferencing

which is an integral problem in bio-informatics and can be mapped to satisfiability. In section 5.2 we discuss the problem of production optimization and well placement for drilling wells in oil fields, which is generally, solved using derivative free techniques. In section 5.3, we analyze a problem in civil engineering of inserting dampers between adjacent buildings to reduce the chances of pounding during earthquakes. In section 5.4, based on the problems analyzed in the earlier chapters we comment on the ideal clock speeds of quantum computers compared to classical processors.

Finally in chapter 6, we summarize our work and present open questions which remain unanswered.

Chapter 2

Preliminaries

In this chapter, a detailed understanding of the concept of qubits, superposition, measurement is not necessary, though an interested reader is referred to [NC10], [KLM07] for an introduction to quantum computing. We describe the power of a few quantum algorithms, quantum walks and black-box query model which will be employed in the subsequent chapters. These algorithmic primitives have been discussed without going into the technical details in such a way that the speed-ups can be understood without completely understanding the details of their working.

2.1 Query model of computation

Quite often in classical algorithms and complexity theory, one is concerned with the *space* or *time* taken by an algorithm to execute and obtain the solution. We generally assume the worst-case scenarios of these algorithms and formulate the expressions to quantify the time or space these algorithms require. A feature which unites both these models of computation is, they both assume the algorithm has *entire* knowledge of the function governing the problem instance. The complete knowledge of the problem allows the algorithm to manipulate the behavior of the function, using the known parameters to eventually obtain the solution of the problem. However, the setting we are concerned with here is if the knowledge of the function governing the problem instance is hidden from the algorithm and we want to derive some *property* about the function. We assume access to the function in terms of a black-box (or an oracle), a computational model which allows questioning the function and receiving its corresponding output. This model of computation is often referred to as black-box model of computation where each question asked is referred to as a query, which could be a bit string $\{x \in \mathcal{D} \subseteq \{0,1\}^n\}$. When the domain of the function $\mathcal{D}$ is $\{0,1\}^n$, the function is referred to as *total function* and if the domain is a subset of

$\{0,1\}^n$ it is called a *partial function*. The goal in black-box model is to optimize the number of queries made to this function in order to characterize some property of the function.

2.2 Quantum query complexity

In quantum computing, since the notion of *time* is notoriously hard to formulate, there is a need to understand the number of queries in isolation for computations, in order to analyze the separation between quantum and classical models of computation. Many important quantum algorithms have shown a significant improvement to classical algorithms in the *exact query complexity* model. Another setting in which quantum algorithms has given interesting results is the *bounded-error quantum query complexity* model where the algorithm is allowed a small probability of error. For partial functions an exponential separation was shown in [DJ92] between exact quantum query complexity and classical query complexity. This exponential separation also holds when considering exact quantum query complexity versus classical bounded error query complexity. However for total functions the best known separation is a quantum algorithm with a quadratic improvement (through the OR function [Gro97]) over a classical algorithm.

With most quantum algorithms involving speedups in the query model of computation, this model has captured the power of quantum computers well. The first algorithm to illustrate a separation using the power of quantum superposition was given by Deutsch in his landmark paper [Deu85], where he gave a factor of 2 improvement for computing if a function $f : \{0,1\} \to \{0,1\}$ is balanced ($f(0) \neq f(1)$) or constant ($f(0) = f(1)$). The basic idea was to compute $f(0) \oplus f(1)$ using one query using superposition instead of two classical queries. A generalization of this algorithm was presented by Deutsch and Jozsa in [DJ92] which presented the first exponential query separation between the classical and quantum setting. They considered the function $f : \{0,1\}^n \to \{0,1\}$ to determine if the function was balanced ($f(x) = 0$ for half the inputs and 1 for the rest of the inputs) or constant ($f(x)$ is equal for all x). Any exact classical algorithm would require a minimum of $2^{n-1} + 1$ queries to determine either of the two cases, but their algorithm demonstrated that in the quantum setting deciding if $f(x)$ is balanced or constant could be done in a single query. Simon's algorithm [Sim97], [BH97] demonstrated a exponential separation between the exact quantum query complexity and bounded-error classical query complexity for the problem: given a function $f : \{0,1\}^n \to \{0,1\}$ with the promise there exists a string s such that for all x, y, $f(x) = f(y)$ if and only if $x = y \oplus s$, find the string s. The best known classical probabilistic algorithm requires exponential queries, but Simon showed a quantum algorithm that required $O(n)$ queries to find the hidden string.

These algorithms paved way to the seminal Shor's factoring algorithm [Sho97] to factor numbers in polynomial time, which had extensive applications in cryptography and Grover's

algorithm [Gro97] which quadratically improved the search problem. Although Grover's search algorithm provides only a quadratic improvement compared to classical algorithms, its importance arises from the wide range of applications.

2.2.1 Quantum Minimum finding algorithm

The problem concerned in this section is to find the minimum element given a database $[x_1, x_2, \ldots, x_n]$. The quantum algorithm for finding the minimum element in a database relies heavily on the descendant algorithm [BBHT96] which is a generalization of Grover's quantum search algorithm. Grover's algorithm for searching in a database works as follows, given a function $f : \{0,1\}^n \to \{0,1\}$ such that it returns 1 for only one of the $N = 2^n$ possibly inputs (such an input shall be referred to as a marked input). Assuming no knowledge about the function $f(x)$, any classical algorithm requires the the function to be queried (to be evaluated) for $O(N)$ inputs to determine the marked input. Grover's algorithm however requires $O(\sqrt{N})$ evaluations of the function to find the marked input. Like other quantum algorithms, this algorithm is probabilistic as well, there is a probability of not reporting the marked input. This probability of failing can however be reduced by repeating the algorithm. This quadratic improvement was also proven to be tight in [BBBV97] and through various lower bound techniques in the past years such as polynomial method, adversary method etc, showing that we cannot find the marked input in lesser than $\Omega(\sqrt{N})$ function evaluations. It was later shown in [BBHT96] even if there were m multiple marked inputs, the algorithm would output one of the marked elements after $O(\sqrt{\frac{N}{m}})$ queries to the black-box. It was in fact shown in their paper that the the number of marked inputs m need not be known ahead of the algorithm to find the marked element.

Theorem 1. *[DH96] Given a database of N elements $[x_1, \ldots, x_N]$, the time taken for the algorithm to find the index of the minimum element in the database is $O(\sqrt{N})$.*

Proof. Note that the exponential sized database need not be present ahead of time for this algorithm to work. The elements $[x_1, x_2, \ldots, x_N]$ could be the output of a black-box for the corresponding inputs $[f^{-1}(x_1), f^{-1}(x_2), \ldots, f^{-1}(x_N)]$. Classically, in this case the number of queries to the black-box to find the minimum is $O(N)$ and this quantum algorithm quadratically improves the number of queries. The quantum algorithm begins by creating a superposition of elements of the database. It is then followed by randomly picking one of the elements of the database and marking those elements in the superposition which are lesser than the picked element. Using the [BBHT96] algorithm, we can search for the marked elements without the knowledge of the number of marked elements in advance. It is then verified if the measured element is lesser than the initially picked element. If it is, then the initially picked element can be replaced by the marked element.

Input: A database $[x_1, \ldots, x_N]$
Output: Index of minimum element in the database.

Step 1: Randomly pick one of the elements x_m.
Step 2: **Repeat** till Run_time $\leq 22.5\sqrt{N} + 1.4 \log_2^2 N$
Step 3: Prepare a superposition

$$\sum_{i=1}^{N} |x_i\rangle |x_m\rangle$$

Step 4: Mark those elements for which $x_i \leq x_m$.
Step 5: Using the [BBHT96] algorithm search for one of the marked elements.
Step 6: Measure the first register (let the outcome be x_k).
Step 7: If $x_k \leq x_m$, set $x_m = x_k$
Step 8: **End** Loop
Step 9: Return x_m

Figure 2.1: Quantum minimum finding algorithm.

Preparation of the superposition takes $\log_2(N)$ time, one step of the [BBHT96] algorithm can be assumed to take one time step, and other steps in the algorithm take constant time. It was shown that the probability of success of this algorithm is *at least* $1/2$ when the algorithm is repeated for $22.5\sqrt{N} + 1.4 \log_2^2 N$ time steps. The probability of success can however be boosted by running the algorithm k times, in which case the success probability of the algorithm would be *at least* $1 - \frac{1}{2^k}$. In the similar spirit there was a canonical algorithm to find the maximum element in the database called the *maximum finding algorithm* [AK99] finding the maximum in an array in $O(\sqrt{N})$ queries. □

2.2.2 Amplitude Amplification

Let $\mathcal{A}$ be an algorithm for computing the function $f : \{0,1\}^n \to \{0,1\}$ which could either be classical or quantum algorithm with *one sided error*. In the case $f(x) = 0$ the algorithm $\mathcal{A}$ never makes a mistake of outputting 1, however in the case $f(x) = 1$, the algorithm $\mathcal{A}$ returns 0 with probability $\varepsilon > 0$. The aim is to firstly construct an algorithm $\mathcal{A}'$ which returns 1 with a probability greater than $2/3$, by recalling $\mathcal{A}$ multiple times and secondly minimize the number of times $\mathcal{A}$ is recalled in the process. Classically it is intuitive to see that to construct $\mathcal{A}'$ we would require $O(\frac{1}{\varepsilon})$ repetitions of $\mathcal{A}$. However in the quantum setting it was shown that,

Theorem 2. *[BHMT00] Let $\mathcal{A}$ be a quantum algorithm with one-sided error and success probability $\varepsilon > 0$. Then, we can construct another quantum algorithm $\mathcal{A}'$ which outputs 1 with a probability at least 2/3 by $O(\frac{1}{\sqrt{\varepsilon}})$ repetitions of algorithm $\mathcal{A}$.*

This idea of quadratically reducing the number of times a one sided classical algorithm is recalled to boost the success probability of another algorithm is referred to as *amplitude amplification*. It is important to notice that in the classical algorithm, if each iteration depends on the previous one (i.e. adaptive in nature), then the theorem [BHMT00] doesn't follow directly and there is no quadratic speedup. Amplitude amplification was shown to be a generalization of Grover's quantum search (which effectively computes the n-bit OR function). Considering the OR function which returns the marked element with non-zero probability ε, amplitude amplification outputs the marked element with *at least* constant probability after querying the function $O(\frac{1}{\sqrt{\varepsilon}}) = O(\sqrt{N})$ times which is effectively the quadratic speedup to searching shown by Grover.

2.3 Random Walks

Random walks are powerful computational tools often encountered in algorithms in computer science and mathematics. The formulation we adopt for random walks are as a succession of steps following a particular rule which is defined by a stochastic matrix. If the stochastic move at any step does not depend on the prior steps of the walk those random walks are referred to as *Markovian chains*. The evolution of the walk over many moves gives us interesting properties of the walk which has been shown to be useful for computations. Quantum walks are natural analogues of random walks where the stochastic matrices describing the probability amplitudes are replaced by unitary transformations, which results in interesting properties due to interference. These walks are very different from classical walks, as they been shown to spread further much faster and not converge to limiting distributions. It is intuitive to see that if after every step of the quantum walk, a projective measurement is made onto the computational basis, the resultant walk is effectively a classical Markovian chain.

Before formulating quantum walks, we describe classical random walks. A classical random walk is characterized by the stochastic matrix M (or commonly referred to as Markov transition matrix) whose entries for example, M_{ij} define the probabilities of moving from vertex i to vertex j on a graph. Given a state of the graph ψ, by left multiplying by the stochastic matrix M, the position of the state ψ after one step of the random walk can be obtained. Often in algorithms requiring n steps of the walk, we require repeated application of walk operator or multiplication of the matrix M^n to the initial state to obtain the state after n steps. We present a classical random walk for solving 2-satisfiability in section 3.2.2 where these ideas become clearer.

Quantum random walks evolved from the similar ideas of a classical random walk. Originally, considered in both the continuous time and space, discrete versions of quantum walks have been the prime focus subsequently. They were found to be easily implementable using quantum circuits and amenable to provide speeds in quantum algorithms. In this section we restrict our-

selves to discrete time quantum walks and the quadratic speedup compared to classical random walks. A naive idea to define a quantum walk is by direct translation of classical random walk, replacing the state ψ by a quantum state $|\psi\rangle$ and replacing the stochastic matrix by a completely positive map. In order to focus on the power of quantum physics, we can restrict our map to be a unitary walk operation, which maintains the system in a coherent state. However, the translation of classical states and maps to quantum states and unitaries don't give us a well-defined quantum walk. Let us consider a simple classical walk where a state can move to either of its neighbors with equal probability. The stochastic matrix governing the walk is

$$M = \begin{pmatrix} \ddots & & \cdots & & \cdots \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ \vdots & \frac{1}{2} & 0 & \frac{1}{2} & \\ & & \frac{1}{2} & 0 & \frac{1}{2} \\ \vdots & & \cdots & & \ddots \\ & & & & \end{pmatrix}$$

If we were to consider the quantum analogue of such a walk with the stochastic matrix replaced by a unitary, at any vertex $|n\rangle$ the walk would move to a superposition of neighboring states $\frac{1}{\sqrt{2}}(|n-1\rangle + |n+1\rangle)$. Similarly at vertex $|n-2\rangle$ the walk would move to $\frac{1}{\sqrt{2}}(|n-1\rangle + |n-3\rangle)$. However, due to the presence of $|n-1\rangle$ term, both these superpositions aren't orthogonal and hence this transformation cannot be unitary. Incorporating other techniques such as phase information, wouldn't preserve the fact that the walk behaves the same at all vertices. Hence, there was a need for a different formulation for quantum walks.

In [AAKV01] Aharonov et al. came up with *coined* quantum walk, where they increased the dimension of the Hilbert space for the quantum walk to $\mathcal{H}_s \otimes \mathcal{H}_c$, where the first space refers to the unitary to be applied on the *state space* conditioned on the second space $\mathcal{H}_c$ called the *coin space*. $\mathcal{H}_c$ can be looked upon as representing the outcome of the classical coin flip or in one of the spin states up/down $\{|\uparrow\rangle, |\downarrow\rangle\}$. This degree of freedom in the coin space $\mathcal{H}_c$ was shown to be required for the physical implementation of quantum walks. Correspondingly they defined operators $\mathcal{C}, \mathcal{S}$ acting on the respective spaces $\mathcal{H}_c, \mathcal{H}_s$ as

$$\begin{aligned} \mathcal{C}|0\rangle &\to \alpha|0\rangle + \beta|1\rangle \\ \mathcal{C}|1\rangle &\to \gamma|0\rangle + \delta|1\rangle \\ \mathcal{S}|n,0\rangle &\to |n-1,0\rangle \\ \mathcal{S}|n,1\rangle &\to |n+1,1\rangle . \end{aligned} \tag{2.1}$$

Note the $\{|\uparrow\rangle, |\downarrow\rangle\}$ can be looked upon as $\{|0\rangle, |1\rangle\}$. Finally, one step of the quantum walk can be represented by the unitary operation $U = \mathcal{S}(\mathbb{I} \otimes \mathcal{C})$, where $\mathcal{C}$ acts first on the coin space with identity acting on the state space, followed by the $\mathcal{S}$ operator on the space $\mathcal{H}_s \otimes \mathcal{H}_c$. If the coefficients $\alpha = \beta = \gamma = -\delta = \frac{1}{\sqrt{2}}$, it is called a Hadamard walk since the unitary $\mathcal{C}$ resembles the Hadamard operation. Another choice of coefficients $\alpha = \delta = \frac{1}{\sqrt{2}}$, $\beta = \gamma = \frac{i}{\sqrt{2}}$ gives the well studied balanced quantum walk.

In order to define the quantum walk, let us define the state $|i, j\rangle$ on the space $\mathbb{C}^N \otimes \mathbb{C}^N$, where the initial state of the walk is $|i\rangle$ and the second register holds the superposition of states after one step of the quantum walk. For each vertex $|i\rangle$ in the graph, we define

$$|\psi_i\rangle = \sum_{j=1}^{N} \sqrt{M_{ij}} \, |i, j\rangle \qquad i = 1, \ldots, N. \tag{2.2}$$

In order to define the unitary defining the quantum walk, let us denote the projection operator onto the span of $\{|\psi_i\rangle : i = 1, \ldots, N\}$ as

$$\Pi := \sum_{i=1}^{N} |\psi_i\rangle \langle\psi_i| . \tag{2.3}$$

After each step of the quantum walk, the walk needs to continue from the second register containing the superposition of neighboring states that resulted from the first step, hence we define the swap operator between the registers $|i, j\rangle$ after each step of the walk as

$$\mathcal{S} := \sum_{i,j=1}^{N} |i, j\rangle \langle j, i| . \tag{2.4}$$

One step of the unitary walk can be finally defined as $U := \mathcal{S}(2\Pi - \mathbb{I})$ and two steps of the walk on the initial state $|\psi\rangle$ would be $\mathcal{S}(2\Pi - \mathbb{I}) \circ \mathcal{S}(2\Pi - \mathbb{I}) |\psi\rangle := (2\Pi^* - \mathbb{I})(2\Pi^* - \mathbb{I}) |\psi\rangle$. In order to completely understand the power of quantum walks, Szegedy in [Sze04] proved a spectral theorem to analyze the spectrum of the eigenvalues of the walk operator U. To understand the result by Szegedy, we define a discriminant matrix $D_{i,j} = \sqrt{M_{i,j} M_{j,i}}$ with eigenvalues between [0,1]. Szegedy effectively proved that if D had a set of eigenvectors $\{|\lambda_i\rangle\}$ with eigenvalues $\{\lambda_i\}$, then the eigenvalues of the discrete-time quantum walk U are ± 1 and $e^{i \cos^{-1}(\lambda)}$. Defining the subspaces $A = span\{|\psi_i\rangle\}$, $B = span\{|\psi_i\rangle\}$, the operator

$$Q = \sum_{j \in V} |\psi_j\rangle \langle j| \tag{2.5}$$

where V refers to the vertices adjacent to j and $|\phi_j\rangle = Q |\lambda_j\rangle$, the theorem can be stated,

Theorem 3. *[Sze04] The eigenvalues of the walk operator U acting on the spaces A, B can be described as follows*

1. *The eigenvalues of U on the subspace $A+B$ are $e^{\pm 2i\lambda_1}, e^{\pm 2i\lambda_2}, \ldots, e^{\pm 2i\lambda_k}$, where $\cos(\lambda_1), \cos(\lambda_2), \ldots, \cos(\lambda_k)$ are eigenvalues of D in the range (0,1). The corresponding (un-normalized) eigenvectors of $U(M)$ can be written as*

$$|\phi_t\rangle - e^{\pm 2i\lambda_t} S\,|\phi_j\rangle \quad t = 1, \ldots, k. \tag{2.6}$$

2. *On $A \cup B$ and $A^{\perp} \cup B^{\perp}$ the quantum walk operator $U(M)$ acts as identity with eigenvalues +1. The linear subspace $A \cup B$ is spanned by the left (and right) singular vectors of $D(P)$ with eigenvalue 1. $U(P)$ has no other eigenvalues on $A + B$.*

3. *On $A \cup B^{\perp}$, $A^{\perp} \cup B$ the operator $U(M)$ acts as the negative of identity with eigenvalues of -1.*

Employing this theorem, a quadratic speedup can be observed for the *search* problem via a quantum walk. Given a graph $G = (V, E)$ with contains *marked* vertices, the classical *hitting time* or the time required to reach the marked vertices with a constant probability can be shown to be $O(\frac{1}{\delta\varepsilon})$ in the classical setting, where δ is the eigenvalue gap between the largest and second-largest eigenvalue, ε is the ratio of the number of marked elements to the total number of elements on the graph. Analyzing the quantum walk, it was shown in [MNRS11] that the quantum algorithm can decide if there is a marked element in time $O(\frac{1}{\sqrt{\delta\varepsilon}})$, which is a quadratic improvement to the classical random walk. The proof for this quadratic speedup is omitted here, an interested reader in quantum walks is referred to excelling surveys in [SM10], [RNB11], [Kem03]

Chapter 3

Satisfiability

The *constraint satisfaction* problem, referred to as CSP, has been ubiquitous in almost every aspect of engineering and science. Most of the well known problems we encounter can be phrased as a CSP such as the n-queens game, the crossword puzzle, Sudoku, Kakuro, graph coloring, etc. The problem involves a homogeneous collection of finite constraints C_i, each involving a finite number of variables x_i belonging to their respective domains D_i. The assignment of values to these variables are related by constraints imposing restrictions on the subset of the variables which they involve. A formal definition for CSP is:

Definition 4. A constraint satisfaction problem is defined by the tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$

1. A set of n variables $\{x_1, x_2, \ldots, x_n\} = \mathcal{X}$
2. A set of domains $\{D_1, D_2, \ldots, D_n\} = \mathcal{D}$ for the respective variables such that $x_1 \in D_1$, $x_2 \in D_2$, $\ldots$,$x_n = D_n$
3. A set of constraints $\{C_1, C_2, \ldots, C_m\} = \mathcal{C}$ where each constraint defines an equation involving *any* number of variables imposing some relation between them.

The aim is to solve or satisfy every constraint in $\mathcal{C}$ with consistent variables assignment. An assignment of domain-bounded variables satisfying each constraint is called a *satisfying assignment*. An example of a simple CSP is $(\{x, y\}, \mathbb{R}^2, \{x \geq 0, y \geq 0\})$, which corresponds to the satisfying assignment of the first quadrant on the $X - Y$ plane.

The *satisfiability problem*, or commonly referred as SAT, is defined as the problem of determining if a Boolean formula has a solution. Formally the problem can be defined as:

Definition 5. Boolean satisfiability problem for k-SAT formula $\mathcal{F}$ is defined by the tuple $(\mathcal{X}, k, \mathcal{C})$

1. A set of variables $\{x_1, \ldots, x_m\} = \mathcal{X}$, where $x_i \in \{0, 1\}$.
2. k denotes the maximum number of variables in each clause C_i.
3. A set of clauses $\{C_1, C_2, \ldots, C_n\} = \mathcal{C}$ where C_i are clauses comprising of disjunction (logical $OR(\vee)$) of binary literals (or variables) $\{x_1, x_2, \ldots, x_m\}$ or their negations. The clauses are combined by conjunction (logical $AND(\wedge)$) of connectives. These clauses give us the framework for the SAT formula $\mathcal{F}$.

We often encounter instances where $|\mathcal{X}| \leq k\,|C|$ with equality (each clause containing distinct variables) giving us an example of uniform k-SAT instance. The aim of the satisfiability problem is to determine if there exists a Boolean assignment of values to each one of the variables x_i such that formula $\mathcal{F}$ is satisfied or evaluates to 1. Considering $\mathcal{F}$ is defined in terms of conjunctions of clauses which are individually a disjunction of literals, representation of $\mathcal{F}$ in this form is called conjunctive normal form (CNF). In terms of logic, if we assign *true* or *false* to each of the literals, we need a satisfying assignment of the variables such that the overall expression $\mathcal{F}$ evaluates to *true*. In this thesis, we would frequently encounter instances of 2-SAT (with clauses containing at most 2 variables), 3-SAT (with clauses containing at most 3 variables), k-SAT (with clauses containing at most k variables). A simple example of 3-SAT is

$$\mathcal{F} = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}). \tag{3.1}$$

Since there are 3 variables involved in each clause it is called a 3-SAT formula. An assignment for $\mathcal{F} = 1$ is $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$ which can be verified trivially.

From the problem definitions of CSP and SAT, it is intuitive to consider SAT as a particular form of Boolean CSP, where the domains of the variables are Boolean and the clauses are connected by logical connectives. An efficient reduction from k-CSP to k-SAT is possible by considering a k-CSP with every constraint containing at most k variables, with the variables taking only Boolean values having the structure as required by k-SAT.

3.1 Complexity of SAT

The first non-trivial instance of SAT intuitively is 2-SAT containing n variables for which Papadimitriou in [Pap91] gave a polynomial time algorithm which solves the problem in $O(n^2)$ steps. The problem becomes intractable considering 3-SAT or for any instance of k-SAT with $k \geq 3$. The Cook-Levin theorem showed that any polynomial-time non-deterministic Turing

machine could be reduced to a polynomial sized Boolean expression, and consequently proved the Boolean satisfiability problem is complete for the complexity class NP. A fairly common technique for proving NP-completeness is by reducing the problem to an instance of SAT through an efficient polynomial time algorithm. Considering P $\neq$ NP (which is widely assumed), it implies in the worst case there exists no polynomial time algorithm to find a satisfying assignment for SAT. Although many instances of SAT for *specific* applications can be solved incredibly fast through various techniques by employing SAT solvers, deterministic or probabilistic polynomial-time algorithms for k-SAT formulas or exponential lower bounds for general k-SAT algorithms remains elusive.

3.1.1 SAT Instances

Although finding a satisfying assignment k-SAT instance with m clauses and n variables is known to be NP-complete, experimental evidence employing various heuristics has shown that the toughness of SAT instances depends intricately on the number of clauses and variables m, n. Intuitively, it can be seen that a SAT instance with many clauses and few variables would likely be unsatisfied and similarly an instance with fewer clauses and many variables would likely be easily satisfiable. From numerical simulation it has been seen that the hardest SAT instances occur near a phase transition region $\alpha_c = m/n$, for which the given SAT instance is considered to be the toughest to solve. For example, considering 3-SAT, the critical constant is $\alpha_c \approx 4.267$ [NLBH+04], where the performance of the SAT solvers are tremendously worsened. This transition between a randomly generated SAT instance being easily satisfiable and probably unsatisfiable is considered as the phase transition occurring at critical ratio $\alpha_c = m/n$ for each k-satisfiable formula and is given by

$$\alpha_c = \frac{1}{\log_2(2^k/(2^k - 1))}. \tag{3.2}$$

Unless mentioned, most of the SAT instances considered in this thesis are in the phase transition region. An interested reader could find many benchmark SAT instances from the SAT repository [SATb] and hard instances can be generated from [Beb]. The universal format in which almost *all* instances for SAT are provided are in .dimacs cnf format with the first 5 lines containing comments before the SAT instance in which clauses are separated by 0's and negation of the literal is denoted by the '-' symbol.

3.1.2 Structured Problems

Often, we come across the *dichotomy* that practical applications involve solving *huge* SAT instances which are generally thought to be *intractable*, whereas algorithmic progress and optimized

implementations have resulted in SAT solvers which have been able to solve these instances in *realistic time*. The answer to the dichotomy lies in the fact that, SAT instances for practical problems are inherently *structured* in construction. Structured instances of SAT are those instances for which the problem can be divided into well defined sub-problems where the intricate dependencies between variables can be exploited in solving the problem. The satisfying assignment for the SAT instance could be incrementally extended by constructing partial solutions to these sub-problems, containing few of the literals and clauses. This is more efficient compared to the unstructured approach of searching, since in this methodology to construct the satisfying assignment, the algorithm avoids search spaces where the partial assignment doesn't satisfy the literals and clauses.

3.2 Classical Algorithms

3.2.1 Brute Force

A naive brute force search algorithm can be used to obtain the satisfying assignment for SAT with a time complexity of $O(2^n)$ as follows

Step 0: **BruteForce** (Input: A k-CNF formula $\mathcal{F}$ with n variables)

Step 1: Assume an initial assignment $\sigma = \{0\}^n$
Step 2: **For** $k = 1 : I_{max}$
Step 3: **If** $\mathcal{F}$ is satisfied by σ
Step 4: Output σ
Step 5: **Else** Find a new assignment σ' such that Hamming distance between σ, σ' is 1.
Step 6: **End**
Step 7: *Output:* **"No Satisfying Assignment found"**

Figure 3.1: Brute force algorithm.

Intuitively, the total number of possibilities for the simple exhaustive search grows with the number of variables exponentially. In the worse case, an exponential number of steps $I_{max} = 2^n$ need to be carried out before finding a satisfying assignment. In the example for 3-SAT in the previous section, the satisfying assignment was obtained by considering every possible Boolean assignment of $\{x_1, x_2, x_3, x_4\}$ in order to find the solution. Hence the total number of possible assignments was 2^4. For a problem with 100 variables there would be a mammoth 10^{30} possible assignments.

3.2.2 Polynomial-time algorithm for 2-SAT

The polynomial time random walk algorithm on the space of strings (assignments) proposed by Papadimitriou in [Pap91] is described below

Step 0: **2-SAT** (Input: A k-CNF formula $\mathcal{F}$ with n variables)

Step 1: Guess an initial assignment σ at random
Step 2: **For** $k = 1 : I_{max}$
Step 3: **If** $\mathcal{F}$ is satisfied by σ
Step 4: Output σ
Step 5: **Else** Choose an unsatisfied clause at random.
Step 6: Flip the value of the variable associated with a random literal in that clause.
Step 7: **End**
Step 8: *Output:* **"No Satisfying Assignment found"**

Figure 3.2: Random walk for 2-SAT.

In each step of the walk, the algorithms moves from one assignment σ_1 to σ_2 if these strings have a Hamming distance of 1 and the differing bit belongs to a clause that is unsatisfied for the assignment σ_1. Since the algorithm has equal probability ($= 1/2$) of moving towards and away from the satisfying assignment, in the worst case scenario it resembles a random walk on a line. It is known that an unbiased random walk on a line requires an expected number of $\Theta(N^2)$ steps to travel a distance of N. Hence the random walk for 2-SAT finds the satisfying assignment in $I_{max} = O(n^2)$ steps. If a satisfying assignment is not found before that, it can be concluded there doesn't exist a satisfying assignment for the SAT instance. The number of required steps can be obtained with a recursion relation between two successive iterations of the algorithm to find the satisfying assignment.

The polynomial nature of this algorithm is a reflection of the fact that in every step of the random walk, there is a probability 1/2 with which the assignment is moving towards the satisfying assignment and 1/2 with which its moving away from the assignment. Similarly for k-SAT with $k \geq 3$, upon a bit flip to a random variable in a clause in each iteration, the probability of moving towards the satisfying assignment is $1/k$ and *this* leads the random walk to be biased away from the assignment under consideration at any step. Theory of random walks states that, a walk that is biased (with a higher probability) away from the current solution will take exponential number of steps to reach the satisfying assignment, which consequently results in the exponential running time random walk algorithms for k-SAT. Proposing faster algorithms involves closely understanding the structure of the SAT instances to come up with more efficient algorithms.

3.2.3 DPLL Algorithm

The first non trivial algorithm in literature for solving SAT dates back to 1962, called the **D**avis-**P**utnam-**L**ogemann-**L**oveland (DPLL) algorithm [DLL62] was based on complete exploration of the search space to find the satisfying assignment. Complete exploration was what leads to a marginal slowdown of this algorithm compared to other algorithms mentioned later in this section. The algorithm divides each formula into clauses and searches in the manner of a tree with the top of the tree being the SAT instance and the leaves in each level being the clauses. In some sense, the structure of this algorithm looks similar to the depth-first search of a tree branching out from the SAT instance. There are various heuristic techniques, which distinguish it from naive depth first search, that guide the nature of the search and in effect optimize the algorithm. The fundamental techniques employed are:

1. *Non-chronological backtracking*: Backtracking is a technique where the satisfying assignment for the problem is built by incrementally building up solutions from an initially assumed random assignment. On reaching a level of the tree where the assignment doesn't satisfy the instance at that particular level, the algorithm backtracks to that level of the tree where the solution satisfies all assignments and carries on with an alternative branch. Non-chronological refers to jumping back up multiple levels instead of going back one level each time and this has proven to be more effective than chronological backtracking.

2. *Unit propagation*: Unit clauses are unsatisfied clauses in the formula that have all but one of the literals assigned. Hence the only possibility for both the clause and formula to evaluate to true is if the last unassigned literal in the unit clause is assigned 1. Unit propagation deletes variables within a clause and gets closer to discovering if there is an empty clause in the instance (a conjunction over null variables which effectively evaluates to false), in which case the formula is not satisfiable.

3. *Clause Learning*: The speedup in DPLL and subsequent algorithms such as Conflict Driven Clause Learning (CDCL) solvers [MSLM09] was based on a powerful technique called clause learning. During backtrack when the algorithm hits upon a conflict it learns a new clause which explains the conflict and hence prevents the repetition of the same conflict again. This new added clause to the database precludes the solver from searching the regions of space where the solution cannot exist.

4. *Pure Literal*: The basic idea underlying this heuristic is that if a literal occurs as the same literal in all the clauses, then it might as well be assigned the value that makes that literal true. Th assignment of this variable is guaranteed not to conflict with any other assignments since it does not introduce a false literal in any clause. The pure literal rule deletes clauses

bringing us closer to the empty formula, in which case the formula is trivially satisfiable with the current assignment.

5. *Resolution Rule*: If a formula $\mathcal{F}$ contains the clauses $(x \vee a) \wedge (\overline{x} \vee b)$ then it reduces to solving $(a \vee b)$. This rule forms the basis of many other techniques for complete solvers, as the algorithm proceeds in each step and concludes when it is an empty formula or all clauses are empty.

Step 0: **DPLL** (Input: A k-CNF formula $\mathcal{F}$, Clause set C, Partial valuation σ)

Step 1: **If** all clauses in C are true, return *satisfiable* and the assignment σ.
Step 2: **Else**, if σ causes few empty clauses, backtrack and carry out DPLL$(\mathcal{F}, C, \sigma')$.
Step 3: **If** C contains unit clause C_i with unassigned variable x_i
Step 4: DPLL$(\mathcal{F}, C, \sigma \cup \{x_i = 1\})$.
Step 5: **If** C contains pure literal x_j
Step 6: DPLL$(\mathcal{F}, C, \sigma \cup \{x_j = 1\})$.
Step 7: Choose an undefined literal x_l occurring in C_i.
Step 8: Carry out resolution step, DPLL$(\sigma \wedge x_l) \vee$ DPLL$(\sigma \wedge \overline{x_l})$

Figure 3.3: DPLL function.

The DPLL algorithm begins with the CNF formula and recursively calls the DPLL function. The first step of the algorithm given a CNF formula, is the decision step, where it chooses an unassigned variable and assigns it a value. Consequently, using the heuristics and optimization techniques the solver finds all the unit clauses created from the last variables assignment. This iterative procedure continues until the current assignment leads to an empty clause (an unsatisfied clause with all the variables assigned values), in which case the solver backtracks and tries the other variable assignment in order to find a satisfying assignment. Through the iterations, the algorithm learns conflicts and avoids these search spaces in subsequent iterations. Avoiding the search spaces where partial assignments of the variables doesn't satisfy the algorithm is referred to as search-space pruning. The algorithm terminates either when all clauses are satisfied and the instance is true, or in the worst case it declares there is no satisfying assignment if the literal assignments have been checked for all possible combinations. The ability to declare the SAT instance as unsatisfiable is the primary advantage of a complete DPLL solver compared to randomized solvers described in the following sections. The worst case complexity of the problem is intuitively exponential where it searches every literal assignment, or is $O(c^n)$ for some constant $c > 1$ employing various heuristics. Over the past years many significant optimization techniques and improvements have been proposed and the DPLL algorithm has been very successful in designing classic solvers such as zChaff, Grasp, Minisat, SATO, Stalmarck, etc. winning many awards in the annual SAT competitions.

3.2.4 Walksat

In combination with various heuristic modifications, Walksat was considered highly successful in practical SAT solving and continues to be used for many practical problems. The algorithm, earlier referred to as GSAT was proposed in 1993 [SKC+93] by Selman, Kautz and Cohen. The algorithm follows the ideas of local search, starts by picking a random assignment and checking if the current assignment satisfies the instance. If it does, the algorithm outputs the satisfying assignment. Otherwise, the algorithm randomly picks an unsatisfied clause C_i, with probability p it picks a literal x_m from that clause and flips the value assigned to x_m or with probability 1-p greedily picks the variable from the clause which would have least effect on other clauses to get unsatisfied by the variable assignment in the CNF. The algorithm checks again if the new assignment satisfies the clause C_i. This process continues until a preset number of steps or time-out. The algorithm restarts with a new random assignment if no satisfying assignment is found after a preset number of steps in the walk. Walksat is a modification of the earlier algorithm GSAT proposed by the same authors, the fundamental difference being the heuristic probabilistic flipping of the chosen literal with probability $1 - p$. One of the key drawbacks of Walksat is its incompleteness. After the preset number of iterations of the random walk if the Walksat algorithm doesn't find a satisfying assignment, it doesn't ascertain that there doesn't exist a satisfying assignment for the SAT instance.

Step 0: **WalkSat** (Input: A k-CNF formula $\mathcal{F}$ with n variables)

Step 1: **For** $i = 1 : I_{max}$
Step 2: Generate a random assignment $\sigma \in \{0,1\}^n$
Step 3: **For** $j = 1 : F_{max}$
Step 4: **If** σ satisfies $\mathcal{F}$, **STOP** and output σ.
Step 5: **Else**, Choose a random unsatisfied clause
Step 6: Select a variable from the unsatisfied clause and flip it
Step 7: With probability p, choose a random variable.
Step 8: With probability $1 - p$, *greedily* choose random variable.
Step 9: **End**
Step 10: **End**

Figure 3.4: Walksat Algorithm.

Using this local search technique there have been many algorithms which have been designed [LWZ07] varying the noise parameter (i.e. probability p) in the protocol. It has been shown that the Walksat algorithm performs best when the probability p is set as 0.47. One of the key obstacles in Walksat is that the performance is worsened on structured SAT instances, in which case the DPLL algorithm is preferred. The performance of Walksat on structured instances is worse because it requires *at least* $O(N^2)$ flips to propagate dependencies among variables during

its random walk, which the *unit propagation* step in DPLL algorithm in 3.2.3 handles in linear time. It is intuitive to see that the Walksat algorithm does not respect any structural dependence of the instance and goes about by walking on a hyper-graph till it finds a satisfying assignment without any search-space pruning.

3.2.5 Schöning's Algorithm

Schöning's randomized algorithm [Sch99] was one of the first algorithms which gave a rigorous analysis of time scaling for finding a satisfying assignment to solve k-SAT. Fundamentally, it was a multi-start unbiased random walk algorithm similar to Walksat with finitely short number of steps, which are repeated exponential number of times to find the satisfying assignment. This random walk algorithm was completely derandomized to give a deterministic algorithm in [MS11]. Schöning's algorithm follows a simple search paradigm starting with a randomly chosen assignment σ.

Step 0: **Schöning's Algorithm** (Input: A k-CNF formula $\mathcal{F}$ with n variables)

Step 1: **For** $i = 1 : I_{max}$
Step 2: Generate a random assignment $\sigma \in \{0,1\}^n$
Step 3: **For** $j = 1 : 3n$
Step 4: **If** σ satisfies $\mathcal{F}$, **STOP** and output σ.
Step 5: **Else**, Choose a random unsatisfied clause
Step 6: Choose a random literal from the unsatisfied clause and flip it
Step 7: **End**
Step 8: **End**

Figure 3.5: Schöning's Algorithm.

In [Sch99], Schöning showed that if random walk in steps [3-7] carries on for a finite number of steps $F_{max} = 3n$ then the probability that the walk finds a satisfying assignment for k-SAT after $3n$ steps is $O(2 - 2/k)^{-n}$. Iterating the random walk for $I_{max} = O(2 - 2/k)^n$ times up to polynomial factors gives a constant success probability of finding a satisfying assignment for the instance. Considering 3-SAT for example, this algorithm scales as $O(4/3)^n$ time, which was improved later by Rolf [Rol03] by combining Schöning's algorithm with a randomized solver to give a slightly better upper bound of $O(1.32793^n)$. Schöning's algorithm can also be extended naturally to the k-CSP problem, by considering the scenario where there are n variables which can take at most d values and the maximum constraint size is of the order l. In the same paper it was proven that the similar algorithm to solve *any* CSP has a complexity of $O(d - d/l + \varepsilon)^n$ where ε is a predefined permissible error bound.

3.2.5.1 Optimization of Algorithm

Drawing inspiration from practical realizations of Walksat and experimental evidence, we hypothesize that although the original random walk algorithm restarts after $3n$ steps, we could propose a new random walk, which could perform better by restarting after a larger number of steps instead of $3n$. This compromises the concise proof provided by Schöning where he proves that an exponential number of restarts guides the random walk to a solution with a high probability. Experimentally verifying the algorithm, we indeed see that by setting an arbitrarily large number of steps as a restart parameter, the algorithm does find a satisfying assignment *much faster*. The following table elaborates the exact time it takes for small instances of SAT to find a satisfying assignment altering the number of steps in the random walk. This technique of not restarting after $3n$ steps does not have any performance guarantee, however assuming the instance definitely has a satisfying assignment this new algorithm performs better.

	I_{max}	F_{max}	Time Taken/ Iteration	Total Time
Sat Instance (3 SAT)				
Instance 1: n=20, m=91	1715	60	0.026 s	45.4 s
Instance 2: n=20, m=91	59	220	0.46 s	27.1 s
Instance 3: n=20, m=91	26	500	0.24 s	6.3 s
Instance 4: n=20, m=91	11	1000	0.12 s	1.3 s
Instance 5: n=50, m=218	-	150	t/o	t/o
Instance 6: n=50, m=218	533	1000	0.36 s	192 s
Instance 7: n=50, m=218	6	30000	3.2 s s	18.7 s
Instance 8: n=50, m=218	1	800000	13.2 s	13.2 s
Instance 9: n=75, m=325	-	225	t/o	t/o
Instance 10: n=75, m=325	337	5000	0.92 s	309.2 s
Instance 11: n=75, m=325	12	30000	3.77	45.3 s
Instance 12: n=75, m=325	1	900000	35.5 s	35.5 s

Table 3.1: Table illustrating the effect of increasing walk steps in section 3.5 where n denotes the number of variables and m denotes the number of clauses.

The number of iterations I_{max} and time taken have been averaged over 100 independent runs of the algorithm to compile this final table. The algorithm was manually stopped after 1000 seconds of running and if a satisfying assignment wasn't found by then, we have labeled that as time-out (t/o). It should be noted that the time taken is significantly less in the scenario where there is a higher number of allowed walk steps F_{max}. For instances where $n \geq 50$, with $F_{max} = 3n$ it was difficult to find a satisfying assignment, hence we had to assume larger values of F_{max} to compile the table. This difference in the total time taken would become even more significant when we consider larger instances. This table was compiled by simply implementing Schöning's

algorithm in Matlab without any heuristics.

3.2.6 PPSZ Algorithm

In 1998, **P**aturi, **P**udlák, **S**aks, and **Z**ane came up with an improved PPSZ algorithm [PPSZ98] for k-SAT which was also called ResolveSAT. It was a randomized variant of the DPLL algorithm. Before describing the algorithm, we introduce few notations to understand the algorithm better. Let the given CNF formula be denoted by $\mathcal{F}$ and the partial assignment of the variables be σ. The *restriction* imposed on the set of variables in the assignment not constrained by σ is denoted by $\mathcal{F}' = F|\sigma$ (formula obtained by making σ true in $\mathcal{F}$). This can be obtained by treating each clause in such a way that, if the clause is set to 1 by σ then delete the clause. Otherwise replace the clause with another clause by deleting any literals of the clause that are set to 1 by σ. Another definition which should be noted is the concept of *bounded resolution* similar to the *resolution* defined in section 3.2.3. Two clauses C, C' are said to conflict on a particular variable x if C contains x and C' contains $\overline{x}$ or vice-versa, and they are called resolvable pair. If they conflict on *only one* exact variable they are denoted as 1-bounded. For such a pair, the resolvent, denoted by $R(C, C')$ is defined as $C_x \vee C'_{\overline{x}}$ where $C_x, C'_{\overline{x}}$ are defined by removing the respective conflict variables. Consequently if $\mathcal{F}$ is satisfiable containing the resolvent pair (C, C') then the new formula $F \wedge R(C, C')$ has the same satisfying assignment as the original $\mathcal{F}$. A pair of clauses are called s-bounded, when their resolvent pair $R(C, C')$ has less than s conflicts. The algorithm can be described as follows:

Step 0: **PPSZ** (Input: A k-CNF formula $\mathcal{F}$, integer d, assignment σ)

Step 1: F:=do k^d- bounded resolution on F.
Step 2: Random permutation π of variables(F)
Step 3: **For** each $x \in$ variables(F) ordered by π
Step 4: **If** F contains a unit clause x or $\overline{x}$ respectively
Step 5: Set the assignment of σ in order to satisfy the unit clause
Step 6: Choose F:=$F|_x$ or F:=F_x depending on $\sigma(x) = 1$ or $\sigma(x) = 0$
Step 7: **End**
Step 8: Return σ

Figure 3.6: PPSZ Algorithm.

The running time of this algorithm is $O(2^{n-n\pi^2/6k})$ for k-SAT which was later de-randomized by Rolf in [Rol05] with a running time of $O(2^{\left(1-\frac{\mu_k}{k-1}\right)n+\varepsilon})$. In comparison to Schöning's algorithm, the PPSZ algorithm is better in terms of running time for k-SAT for $k \geq 4$. The PPSZ solves 3-SAT in $O(1.3071^n)$ steps. The caveats for PPSZ algorithm are however its inefficiency when there are multiple solutions for the SAT instance, and consequently it is considered one of the

faster algorithms for Unique k-SAT which as the name suggests involves solving a SAT instance with a unique satisfying assignment.

3.2.7 An improvement to Schöning+PPSZ

Iwama and Tawaki in [IT04] proposed an algorithm which was a combination of Schöning's algorithm and PPSZ with a better time bound of $O(1.324^n)$ for 3-SAT. The basic idea was the observation that PPSZ algorithm gets worse when the number of solutions increases in which case Scöning's algorithm is more efficient. Their algorithm involved coupling the ResolveSat which was a randomized DPLL with bounded resolution with the local search algorithm by Schöning in parallel. Since both algorithms cannot achieve their worst-case complexities simultaneously, the resultant parallel algorithm was expected to perform better than the constituent algorithms. However, the time scaling of their algorithm was better for 3-SAT, 4-SAT and for $k \geq 5$ it didn't outperform other algorithms. The caveat in PPSZ of not being able to find multiple satisfying assignments was overcome in this algorithm.

Step 0: **IT Algorithm** (Input: A formula $\mathcal{F}$ which is a k-CNF with n variables)

Step 1: Repeat k times
Step 2: General a random assignment σ
Step 3: Carry out steps [3-7] of Schöning's algorithm
Step 4: **If** a satisfying assignment is found, return σ
Step 5: Carry out steps [3-7] of PPSZ algorithm.
Step 6: **If** a satisfying assignment is found, return σ
Step 7: Output **No Solution found**

Figure 3.7: Iwama-Tawaki Algorithm.

The probability with which the above algorithm finds a satisfying assignment in steps [2-6] for 3-SAT was $O(1.324^{-n})$ which was obtained by combined analysis of the respective algorithms. Hence the polynomial algorithm is repeated exponential number of times $I_{max} \approx O(1.324^n)$ in order to find the satisfying assignment with constant success probability.

3.3 SAT solvers

The annual SAT solving competitions [sata] is a platform where people globally compete to solve benchmark hard problems using their designed SAT solvers. Considering SAT is one of the most ubiquitously known NP-complete problem to which most of the problems can be polynomially reduced to, an efficient algorithm to solve this problem is a competitive challenge and

would be a significant milestone for researchers. Most of these participating solvers are designed for various purposes and differ fundamentally in how they are designed, how they work, their efficiency and their guarantees. Some of these are designed to work best on problem instances drawn randomly from a particular distributions, while some are designed to cater to particular industrial problems, some on hard problems near critical regions, and some algorithms are based on heuristics and hence do not prove non-existence of a satisfying assignment. Due to all these factors, any particular SAT solver wouldn't necessarily be efficient on all SAT instances. Some famous solvers winning awards in SAT competitions chronologically are GRASP [SS97], CHAFF [MMZ+01] in 2002, Minisat [ES04] in 2005, SATzilla2009$_R$ [XHHLB09] in 2007, Glucose [AS09] in 2011, and Lingeling aqw [Bie13] in 2013. It should be noted that these solvers are not exhaustive, even as we read, there are computer scientists improving existing algorithms to build a more efficient solver.

A natural question that arises is, *which solver is the best?* And *which solvers do the industries use?* The first question doesn't have a single answer for the reason that each SAT solver is built for either a particular application with their respective pros and cons. As for the second question, it is generally unknown since industries do not disclose the solvers they use to solve a problem, but we believe they would use the state-of-the-art solvers which employ Walksat or DPLL for solving SAT depending on the problem instances. The incomplete stochastic local search nature of Walksat, has proven to be highly effective compared to deterministic solvers, on problems which involve solving random k-SAT instances. There have been many comparative studies with DPLL-solvers in [GS03],[FF04] where they show that for randomly generated instances of SAT with no structure, occurring in the planning problem, bounded model checking, hardware and software verification, graph colouring and circuit synthesis problems, the best DPLL based solver *minisat* didn't perform as well as Walksat. Minisat was one of the better solvers used by people since it was an open source code. It was considered a norm to beat this code to show a particular optimization technique was promising. Fundamentally, Mini(imal)sat is a complete solver, an implementation of DPLL algorithm based on backtracking and conflict driven learning and hence performs better on special structured instances. In general, as mentioned in section 3.1.2 the hardness of SAT instances lies in the structures hidden in them rather than the size of the instances, in which case DPLL-based solver *minisat* has an advantage over the random walk Walksat. Consequently industrial SAT instances which heavily depend on structure are efficiently solved by DPLL-type algorithms and thus would be the preferred solver by industries.

An interesting idea which was investigated in [FF04], was to couple the advantages of both the DPLL algorithm and the local search algorithm to concoct a hybrid algorithm which performs better than either. Ferris and Froehlich however showed that there was *no* improvement. They considered if the incomplete but fast nature of Walksat could be used as a heuristic for the com-

plete but wasteful (in the context of searching the entire space) DPLL algorithm. They highlight that for smaller clause/variable ratio their hybrid algorithm performs in par with Walksat but at the critical phase transition region performs very much slower than either of DPLL or Walksat. In [WS06], Wei and Selman consider improving the random walk in Walksat by introducing constraints to capture long range dependencies (which as mentioned in section 3.2.4 causes Walksat to perform worse on structured instances). By introducing a "negative" bias in their random walk (away from the marked element), they show that their new algorithm performs better on structured instances than the original Walksat algorithm in [SKC+93]. However, this speedup to Walksat only holds for instances with *relatively few* long range dependencies which is not the case for many industrial instances in software and hardware checking, in which case their new Walksat algorithm didn't compare well to DPLL algorithm. In fact for 2-SAT instances their proposed Walksat algorithm found the satisfying assignment in $\Theta(N^2)$ steps which was solved by DPLL algorithm in linear time.

3.4 Quantum improvements

3.4.1 General Overview

It is interesting to note that many algorithms for satisfiability in sections 3.2.4, 3.2.5, 3.2.6, 3.2.7 can be expressed as exponential number of iterations (I_{max}) of a polynomial time algorithm to find the satisfying assignment. An expression of I_{max} was explicitly calculated in [Sch99], where it was shown that the exponential number of iterations of the random walk boosts the algorithm to achieve a constant probability of success. As highlighted in section 2.2.2, by amplitude amplification the exponential number of iterations to achieve constant success probability in the classical algorithm can be quadratically improved. In all the considered algorithms, the random walk is non-adaptive or no iteration depends on previous iterations of the walk which allows for a quadratic speedup. Amplitude amplification algorithm converts any classical algorithm with running time t, with success probability p having overall running time $O(\frac{t}{p})$ to a quantum algorithm which achieves the same constant probability by iterating the algorithm quadratically fewer times with running time $O(\frac{t}{\sqrt{p}})$. The observation to employ amplitude amplification to the random walk algorithm for k-SAT was first noticed by Ambainis in [Amb04] and subsequently in [DW05] they highlighted quantum improvements to various random walk algorithms. The following table summarizes the theoretical quadratic speedup of algorithms for various SAT algorithms.

	Classical Complexity	Quantum Complexity
Algorithm		
Schöning's Algorithm [Sch99]	$O(2-2/k)^n$	$O(2-2/k)^{n/2}$
3-SAT [Her11],[Rol05]	$O(1.30704^n)$	$O(1.1433^n)$
k-SAT with no condition on clause length [DW05]	$O\left(2^{n\left(1-\frac{1}{\ln\frac{m}{n}+O(\ln\ln m)}\right)}\right)$	$O\left(2^{n/2\left(1-\frac{1}{\ln\frac{m}{n}+O(\ln\ln m)}\right)}\right)$
PPSZ Algorithm [PPZ97]	$O(2^{n-n\pi^2/6k})$	$O(2^{n/2-n\pi^2/12k})$

Table 3.2: Classical vs. Quantum query complexity for SAT algorithms.

To illustrate the computational algorithmic speedup, we have compiled the table 3.3 where we have implemented Schöning's algorithm in [Sch99] using Matlab without the use of any heuristic to present few results of the classical time taken to find the satisfying assignment. Using this data, we have calculated the effect of amplitude amplification on the algorithm. Theoretically if there were I_{max} iterations with F_{max} flips respectively, the overall time taken classically is dependent on $I_{max}F_{max}$.

	Number of iterations	Time Taken/ iteration (2GHz)	**Overall time** (2GHz)
Solving 3-SAT			
n=20 m=91	1.6283×10^3	0.0113 s	18.35 s
n=30 m=134	3.7976×10^4	0.0157 s	10 m
n=45 m=200*	2.4275×10^3	3.665 s	2.5 h
n=53 m=237*	4.7233×10^5	0.024 s	3.2 h
n=65 m=297*	1.2234×10^6	0.0121 s	4.1 h
n=75 m=325*	9.1197×10^8	2.4080×10^{-5} s	6.1h

Table 3.3: Summary of classical time (with clock speed=2GHz) required for solving the 3-SAT instances. Starred instances imply the number of random walk steps $F_{max} \geq 3n$.

By amplitude amplification the overall time taken quantumly would depend on $\alpha\sqrt{I_{max}}F_{max}$, where α is the ratio between the classical versus quantum clock speed or effectively the slowdown in quantum hardware compared to classical architecture. We have considered the possibilities of a quantum computer being slower than a classical computer by a factor of $\alpha = 10^3$ and the case where the quantum computer would have the same clock speed as a classical computer. Essentially, since we are computing the quantum time versus the classical time with the same number of iterations I_{max}, it is intuitive to see that when $I_{max} \geq \alpha^2$, we get a quantum advantage using amplitude amplification without rebalancing the parameters (F_{max}, I_{max}). In the following table, we quantify the improvement by quadratically improving the number of iterations in the random walk algorithm for each instance in table 3.3.

	Number of iterations	Time Taken/ iteration (2MHz)	**Overall time** (2MHz)	Time Taken/ iteration (2GHz)	**Overall time** (2GHz)
Solving 3-SAT					
n=20 m=91	32	11.3s	362 s	0.0113 s	0.36 s
n=30 m=134	154	15.7 s	40 m	0.0157 s	2.46 s
n=45 m=200	39	3665 s	40 h	3.665 s	2.3 m
n=53 m=237	542	24 s	3.61 h	0.024 s	13 s
n=65 m=297	0.868×10^3	12 s	2.9 h	0.0121 s	10.5 s
n=75 m=325	2.3744×10^4	2.4080×10^{-2} s	12.4 m	2.4080×10^{-5} s	0.57 s

Table 3.4: Summary of quantum time (with clock speed=2MHz and 2GHz) for solving the SAT instances.

Experimentally, from the above table we do not gain a significant advantage (with a 10^3 times slower quantum computer) for smaller instances of SAT with few variables since the number of iterations taken to obtain the satisfying assignment is small. This is however immediately offset once we consider instances of SAT where it requires a very high number of iterations to obtain a satisfying assignment. Due to time compromises we couldn't run higher instances of SAT but it is natural to see that for k-SAT with possibly $n = 500$, $m = 2200$ we could obtain speedups comparing days to minutes (even with a 10^3 times slower quantum computer) demonstrating the power of using amplitude amplification to classical algorithms.

3.4.2 Quantum analogues of SAT solvers

In the previous section, we looked at the algorithmic speedup of few algorithms along with few numerical implementations which seemed to complement our intuition. In this section we consider the most popular heuristic techniques employed by industry, i.e. Walksat and DPLL and consider if we could get a quantum improvement. In Walksat, due to the heuristic variable selection technique in the random walk step, there doesn't exist a rigorous analysis in terms of the time scaling of the algorithm. However, it is intuitive to see from the pseudo codes of Schöning's algorithm and Walksat that fundamentally Walksat, similar to Schöning's algorithm, is a random walk algorithm for finite (F_{max}) steps and this random walk is run an exponential I_{max} number of times to achieve constant success probability. Moreover, regardless of the heuristic step in the random walk, none of the iterations of Walksat are dependent on the previous iterations and hence is non-adaptive. Similar to Schöning's algorithm, by employing the techniques of amplitude amplification we could achieve the same results as I_{max} iterations of Walksat using $\sqrt{I_{max}}$ iterations of a simple quantum-Walksat.

Unlike other random walk algorithms DPLL is deterministic in nature where the steps made by the algorithm proceed in a deterministic manner and are dependent on the steps taken in prior

iterations. Consequently, as mentioned in section 2.2.2 it can be seen that amplitude amplification technique does not directly apply to DPLL algorithm. However, instead of amplitude amplification there do exist techniques from [CGW00] where they consider a tree-like structured search algorithm and propose utilizing the structure of such search problems to speedup the searching quantumly in each step. This could be looked at as a quantum analogue of DPLL algorithm. We have gone through this idea in detail in section 3.5.3.

We have shown that irrespective of the technique employed by industries to solve satisfiability, there exist techniques such as amplitude amplification or nested Grover's search that bring about a quantum improvement to incomplete random walk solvers or complete depth-first search algorithms. Quantum SAT solving heuristics (apart from those designed for annealing type algorithms with special purpose devices) have not been extensively developed, since we do not have a quantum computer to test the solver. However, any quantum improvement to algorithms used for SAT *should* ideally be compared with famously used classical SAT solvers employing DPLL or Walksat algorithms.

3.4.3 Quantum Heuristics

A potential question that could posed here is could we use quantum adiabatic algorithms to speedup classical heuristics like simulated annealing and do any better? This scenario was considered initially by Farhi et al. [FGG+01] where they conceptualized a quantum adiabatic algorithm, which was fundamentally the quantum analogue of simulated annealing in the classical framework. The concept involved in quantum adiabatic algorithm is that a Hamiltonian, H_g is initially prepared for which the ground state is known and can be easily prepared. It is allowed to slowly transition to another final Hamiltonian H_f which would encode solutions to the desired problem. Considering the SAT instance, the final Hamiltonian H_f would encode the solution to an instance of SAT. The quantum adiabatic theorem states that if a quantum computer starts in a ground state with high probability, it will remain in the ground state at the end of the evolution, hence would describe the solution to the problem.

In particular the inefficiency in this problem arises in the time scaling of the algorithm. It is not known clearly how slowly the slow evolution of the Hamiltonian should be and consequently how the running time of the algorithm scales with the complexity of the problem size. There have been many results of annealing run on asymptotically large problems analyzing the running time of such algorithms. In [VDMV01], they argue that the adiabatic approach to solving problems is similar to local search techniques and proved exponential lower bounds for few minimization problems. In [Hog03], [HY11] adiabatic approach to solving k-SAT was considered where they showed that employing heuristic techniques like adiabatic algorithm improves over unstructured search but *doesn't* give a better running time when compared to the classic Walksat algorithm.

The running time of these adiabatic algorithm still remains exponential and in fact runs slower for particular instances.

3.5 Practical Flipside

3.5.1 Amplitude Amplification

In section 3.4 we highlighted amplitude amplification could be instrumental in speeding up classical algorithms and solvers which are considered state-of-the-art by industries. In this section though, we highlight some obstacles for quantum techniques to provide an actual advantage on industrial SAT instances. Considering the Walksat algorithm used in [CSL13] SAT competition 2012, the algorithmic parameters (I_{max},F_{max}, p) were set as $(2, 2 \times 10^9, 0.48)$. In any eventuality, the solver was manually aborted after 5000 seconds of running time. The number of iterations, I_{max} which was expected to be an exponential number of iterations to find a satisfying assignment is set to 2 and in practice the random walk is bounded above by a large number of steps.

Naturally, by amplitude amplification of the number of iterations, the time scaling is not going to improve due to the chosen parameters for this problem. This choice of parameters for Walksat is not an exclusive case though, in *majority* of the problems where Walksat is implemented, the maximum number of restarts (or I_{max}) for practical problems that are considered are 100 ([ZRL03], [SKC+93], [ZY13], [Coh11], [Men09],[WLZ08], [BH99], [Ach07], [Fuk04]) with an arbitrarily high number of random walk steps. Note the similarity between this aspect and section 3.2.5.1 where we optimized the number of random walk steps for Schöning's algorithm and realized that the time scaling of the entire algorithm is better with a larger number of random walk steps with fewer number of restarts.

From the perspective of theoretical guarantees, setting F_{max} (or the number of flips) as exponential would *not* guarantee finding a solution, however from the perspective of time complexity it appears to perform better in practice which is a prime focus in SAT competitions or industries. In [KSS10], they analyze the tradeoff between the probability of success of Walksat with varying noise parameters. Without consideration of the number of iterations, their results show that with noise parameter $p = 0.56$, setting $F_{max} \approx 10^{10}$ reduces the probability of error of Walksat to approximately 10^{-4}. Industries are primarily concerned with finding a solution rather than any success guarantees and over the time people have realized that by incrementing F_{max} and reducing random restarts I_{max}, the SAT solver finds a satisfying assignment faster. This notion was demonstrated to be faster in [CSL13], where they demonstrated that rather than randomly restarting with a new assignment after a small number of steps of the walk, carrying on with the random walk for a large number of steps proved to be better. The following table

compiles few results from previous papers where the number of iterations I_{max} is non trivial to get a picture of how amplitude amplification *doesn't* give us a significant advantage. Note that in all the instances mentioned below the solutions were always found.

	Literals	Clauses	I_{max}	F_{max}	**Overall classical time**(2GHz)	**Overall quantum time** (200MHz)	**Overall quantum time**(2GHz)
SAT solver							
Walksat [SKC+93]	600	2550	100	241651	35 s	28 s	2.8 s
Walksat[SKC+93]	2000	8480	100	23×10^6	3225 s	2530 s	253 s
BGWalksat[ZRL03]	32109	150027	100	2.2×10^7	358.5 s	290 s	29 s
BGWalksat[ZRL03]	39598	19477	100	23×10^7	334 s	262 s	26.2 s
Beijing Competition [Coh11]	2100	113729	100	15×10^5	1449 s	1140 s	114 s
Beijing Competition [Coh11]	2250	123329	100	15×10^5	3137 s	2470 s	247 s
Walksat43 [Fuk04]	459	4598	100	3×10^5	7.62 s	6 s	0.6 s
Walksat43 [Fuk04]	19500	103887	100	3×10^5	1456 s	1140 s	100 s

Table 3.5: Numerical evidence of Amplitude amplification in sample instances.

We also present results from [Spe93], where Spears analyzes the GSAT algorithm (Walksat algorithm described 3.4 with $p = 0$) proposed in [SKC+93] and provides exact running time analysis for various hard SAT instances. [1]

	Flips	F_{max}	I_{max}	Classical time (2GHz)	Quantum time / iterations (200MHz)	**Overall time** (200MHz)	Quantum time/ iteration (2GHz)	**Overall time** (2GHz)
Solving 3-SAT								
n=100, m=425	21250	500	425	0.1 m	$2.8235\times10^{-3}s$	23 s	2.8235×10^{-4} s	2.3 s
n=200, m=825	497000	2000	2485	2.8 m	3.3803×10^{-3} s	4.4 m	3.3803×10^{-4} s	0.44 m
n=300, m=1275	1390800	6000	2318	12 m	5.1769×10^{-3} s	19.3 m	5.1769×10^{-4} s	1.9 m
n=400, m=1700	3527200	8000	4409	34 m	5.7836×10^{-3} s	39.2 m	5.7836×10^{-4} s	3.9 m
n=500, m=2125	9958000	10000	9958	96m	5.7843×10^{-3} s	78.4 m	5.7843×10^{-4} s	7.8 m

Table 3.6: Summary of quantum time (with clock speed=200MHz and 2GHz) to solve hard SAT instances.

From the above tables, it can be inferred that, in the quantum paradigm, irrespective of the number of flips, optimistically by quadratically improving the number of iterations, we get an improvement of maximum 10 times compared to the classical setting. This speedup would be offset if we considered a quantum computer to be slower than 10 times than a classical computer, which is highly likely.

An interesting idea which could warrant some quantum improvement lies in *rebalancing* the parameters for the Walksat algorithm. In the above instance, we have simply considered the

[1] Since the number of iterations for the hard instances considered is not mentioned in [SKC+93], I_{max} is set as $10^*(flips/F_{max})$ iterations.

Walksat algorithm considered in SAT competitions or practical applications and applied amplitude amplification to I_{max} keeping F_{max} a constant. From various successful implementations it is well known that having an exponential number of flips and finite number of iterations outperforms the original Walksat algorithm which has exponential number of iterations of finite number of walk steps to attain constant success probability. The relationship between I_{max} and F_{max} seems intricate in reality and there has not been much study in understanding how the parameters would play a role in the overall success probability of the algorithm. Although it seems unlikely that increasing the number of iterations would interest the industries, employing quantum amplitude amplification to an increased number of iterations with a fewer walk steps, might compare better than the classical Walksat algorithm considered in reality. Due to limited code access and time constrains, we have considered this idea for Schöning's algorithm implemented on Matlab.

Instance	I_{max}	F_{max}	Total classical time (10 GHz)	Time required for Random walk	Total quantum time (10 GHz)
20/91	1739	60	22 s	0.0127 s	0.42 s
	463	100	6.91 s	0.0147 s	0.25 s
	253	140	5.36 s	0.0212 s	0.27 s
	164	180	4.63 s	0.0282 s	0.28 s
	58	300	2.72 s	0.0469 s	0.24 s
	24	500	1.854 s	0.0771 s	0.29 s
	12	720	1.154 s	0.09 s	0.24 s
	10	1000	1.22 s	0.12 s	0.29 s
50/218	414	2000	131 s	0.3164 s	5.0 s
	122	3500	85 s	0.69 s	5.9 s
	75	5000	62 s	0.8267 s	5.57 s
	19	10000	23.6 s	1.296 s	4.3 s
	3	50000	15.69 s	5.23 s	7.14 s
	2	100000	14.3 s	7.15 s	7.61 s
	1	800000	13.2 s	13.2 s	10.36 s
75/326	600	4000	447 s	0.745 s	14.3 s
	102	8000	187 s	1.833 s	14.5 s
	22	15000	60 s	2.73 s	9.89 s
	8	50000	38 s	7.6 s	16.8 s
	2	100000	39 s	19.5 s	21.6 s
	1	500000	36 s	36 s	28.2 s

Table 3.7: Rebalancing parameters in the implementation of Schöning's algorithm

In the table above, the quantum time for each instance was computed after applying amplitude amplification to the number of iterations as discussed in section 3.4. It can be seen that there is a trade-off between the number of iterations and the number of flips while analyzing the quantum time required for all the instances. In each instance, there exists a set of parameters (I_{max}, F_{max}) which improves upon the best classical time and amplitude amplification applied directly to Schöning's algorithm without increasing the number of walk steps. For example, for the instance n=50, m=218, classically one would choose I_{max}=1 and F_{max}=800000, and

clearly there is no quantum speed-up in this case. However, the rebalanced parameter choice of I_{max}=18 and F_{max}=10000, gives the optimal quantum time (assuming the same clockspeeds) as well as improves upon the best classical time. In the same spirit, it would be interesting to consider rebalancing such parameters on actual SAT-solvers which are employed by industries for instances of practical relevance. We have listed this as one of the interesting directions for future research.

3.5.2 Quantum random walk

Another perspective that could be considered here is, could we *quantumly improve the random walk* itself? Could we *use the result of [Sze04] for random walks in SAT* to improve the number of flips and consequently time to find a satisfying assignment? The first question was considered by Hoyer in [Hoy08] where he analyzes the effect of quantum version of Walksat on SAT instances. It was seen earlier in section 3.2.4 that the random walk doesn't respect the structure in the instance, since it walks randomly on a hypergraph searching for a marked element. This was explicitly demonstrated for 2-SAT instances where the quantum walk did *not* respect the structure of the 2-SAT problem to yield any algorithmic advantage. The scenario does not improve when considering arbitrary k-SAT, where the quantum version of Walksat is not expected to outperform the classical random walk. Walksat algorithm is considered powerful *only* when the SAT instances are unstructured, which is often not the case in industrial applications involving instances with intricate internal structure. Hence a quantum speedup to the random walk will necessarily not have considerable impact when considering industrial problems with structured instances.

However, even in the unstructured instances, the quantum speedup to the random walk isn't a straightforward generalization of the result of Szegedy's spectral theorem [Sze04]. The complication primarily arises because the random walks in Walksat are on *directed graphs* because of the presence of sinks in the hypergraph, which represents the unknown satisfying assignment of the SAT instance. For directed graphs, quantizing a random walk is non-trivial since *not all* SAT instances need to be represented by a Eulerian graph [2]. The quantum walk operator for a Eulerian graph can be represented by a unitary map, which is a square matrix to reassign the n incoming amplitudes to n outgoing amplitudes. In undirected discrete-time walks, depending on the stochastic matrix there is an obvious option (to return back to the edge in the previous step) to not break the symmetry in the walk, however this option doesn't exist in directed quantum walks. In the eventuality of the graphs not being Eulerian (which is common in the hypergraphs of a general k-SAT instance) alternatives were proposed such as linear isometries or non unitary operators for the walk operator and interspersing the walk with measurements to define a "partial quantum walk" in [Mon05].

[2] A graph is Eulerian if the number of edges incident and reflected for every vertex on a graph is the same.

Linear non-unitary walk operators were employed in [Hoy08] for algorithms to solve SAT and it was seen to not result in any significant improvement. Considering 2-SAT explicitly, it was shown that the average probability of any directed quantum walk converging to a solution was equivalent to the probability of a classical random walk with a Markov chain being simulated. Montanaro in [Mon05] analyzed directed quantum walks, where it was shown that discrete-time quantum walk on a directed graph can be defined *if and only if* the graph is *reversible* [3]. The proof of this reversibility criterion relies on using the cycles through the graph instead of the coin space in undirected walks. This is however *not* useful in the hypergraph for k-SAT where the objective is to understand the nature of the graph globally. Interspersing the walk with measurements didn't lead to a full quantum speedup because measurements maintain coherence among the nodes in a cycle and not amongst the cycles themselves, hence resembling a *partial quantum walk in the reversible regions of the graph.*

Employing non-linear isometries and interspersing measurements in random walks are unfortunately not applicable and do not lead to any speedup for the random walk algorithms of k-SAT. The primary reason is the irreversibility in the directed hypergraph for k-SAT instances. By removing the directedness property from the graphs, it results in an undirected quantum walk for which a quadratic speedup in hitting time compared to classical walk has been discussed in section 2.3. The above arguments lead us to the following *conjectures*, for unstructured SAT instances solved using Walksat by industries, amplitude amplification does not propose any significant advantage as discussed in section 3.5.1. Improving the random walk can immensely improve the running time of SAT instances, however since the instances are represented by directed graphs, until we have a well defined notion of a quantum speedup for *directed walks* that we are able to exploit, we cannot improve the running time for satisfiability.

3.5.3 Complete SAT solver

Most of the work in literature involved with SAT is on randomized/probabilistic procedures to solve SAT. In this section we shall highlight an algorithm which can be used to solve SAT completely. This algorithm was first highlighted by Cerf et al. [CGW00] where they considered the problem of structured quantum search and provide a quantum speedup which performs better than quantum unstructured search and classical structured search. It is intuitive to believe that classical structured algorithm is faster in searching for a marked element than an unstructured algorithm since it avoids searching in the spaces where the partial assignments aren't satisfied. The first mention of a quantum speedup to classical structured searching was provided in [CGW00], which was called the *nested quantum search algorithm* since it involved a quantum speedup to the levels of the tree in the structured search algorithm.

[3] A graph is reversible if given any two vertices in the graph (i, j), there is path from i to j and vice versa.

Consider a search tree similar to depth first search, which contains the SAT instance at the top and slowly branches out to every literal at the base of the tree. The idea in [CGW00] is to prepare a superposition of partial solutions at level i, where the assigned variables are called *primary variables*. With the knowledge of the partial assignments (of primary variables) at level i, search exhaustively for the *secondary variables* that satisfy the remaining literals below the level i. Using the structure in the problem and partial assignments of primary variables, searching the entire space can be avoided. The choice of i is important because if i is near the base of the tree it would imply creating a superposition of partial solutions (constituting primary variables) of almost the entire space. Although creating such a huge superposition is undesirable, such a superposition of primary variable assignments is a better predictor for searching the secondary variables. Choosing i near the root of the tree wouldn't be useful as well since a majority of the partial assignments near the root would satisfy their respective instances not allowing the algorithm to discriminate between solutions and non-solutions.

Considering the problem of assigning b values to n variables, naively there are b^n possible assignments through which we would need to sample to obtain the correct assignment. Using quantum search this complexity can be improved by sampling $O(b^{n/2})$ possibilities to find the required assignment. It was shown in [CGW00], by considering structured problems (represented by depth-first search) the same problem could be solved in $O(b^{\alpha n})$ queries where α is a constant dependent on the level of nesting i in the algorithm.

Skeleton Algorithm ([CGW00])

- Construct a superposition of primary solutions at level i with equal amplitude similar to unstructured search.
- Perform a quantum search on each of the descendants of the partial solutions constructed in the previous step. The outcome of this step involves a superposition of assignments where few assignments have their amplitudes amplified.
- Using the above two steps as a oracle, repeat this oracle a finite number of times until the amplitude of the satisfying assignment is maximum before measurement.

Figure 3.8: Structured Grover search algorithm.

It is also interesting to note that the above algorithm improves only the search steps in the classical structured searching algorithm and does not consider other factors involving the clauses and the literals of the SAT instance. It was shown that optimal nesting level for 3-SAT is 0.682 and hence the complexity of 3-SAT with n variables was $O(1.2668^n)$ which was faster than the $O(1.33^n)$ complexity of random-walk algorithm by Schöning [Sch99]. However, it is worse than amplitude amplification applied to Schöning's algorithm which has the complexity of $O(1.155^n)$. A probable reason for this slower algorithm is because CNF expressions are sensitive to the values of *all* the variables they contain and the assignments they are assigned. An important advantage

employing this nested Grover search is its completeness, since if an instance is unsatisfiable then the algorithm would return that the instance doesn't have a satisfying assignment with high probability.

In order to fairly compare this quantum nested search algorithm for SAT with DPLL-based SAT solvers we need to employ heuristics that are applied in standard DPLL algorithms as mentioned in section 3.2.3. In the first step and second step, while preparing a superposition for the primary variables and searching exhaustively for secondary variables, we could employ techniques such as backtracking, clause learning, pure literals, unit propagation, to make the algorithm more efficient and faster. Note that there is no need to quantize each of the heuristic techniques in order for them to be incorporated into the algorithm since effectively we are considering a classical algorithm and only the search steps are quantumly improved. Hence, using the power of quantum speedup coupled with the classical heuristic techniques employed in DPLL solvers, we could design a faster quantum SAT solver.

The drawback however is the complexity of DPLL algorithms are very complex to analyze and there doesn't exist any rigorous bounds on the running time of the algorithm, which could be attributed to various heuristics employed in the algorithm. From an algorithmic perspective in [ABM04], [AHI05] they showed lower bounds for recursion tree-like resolution searches proving exponential lower bounds for *special* cases of DPLL algorithm. Proving a general exponential lower bound for DPLL algorithm would be equivalent to the claim P $\neq$ NP. Participants of SAT competitions, design heuristic improvements to existing algorithms and *verify* their algorithm on the hard instances to check if the running time is indeed faster. However, we do not have large scale quantum computers, and thus are not able to experimentally quantify the time scaling of the quantum version of the algorithm with heuristic improvements. The algorithm in theory *should* perform faster than the classical DPLL solver but we cannot provide exact complexities or run times to prove it.

3.6 Summary

In this chapter we gave an introduction to hardness of SAT in section 4.2 by going over the complexity of SAT and defining *structure* in instances which is integral for solving hard SAT instances. We went through few classical algorithms in section 3.2 which are often implemented in famous SAT solvers discussed in section 3.3. We showed in section 3.5.1 that quantum amplitude amplification doesn't necessarily guarantee speedups in many problems which are considered often in practical problems or SAT competitions. Speeding up the quantum walk in random walk algorithms would have been interesting since *majority* of the incomplete classical algorithms depend on random walks, however a well defined notion of quantum speedup in *directed walk* seems

a hard problem. Finally in section 3.4 we commented on nested Grover's search algorithm which could be thought of as a quantum analogue of DPLL algorithm. Although this algorithm was faster than classical random walk algorithms, it didn't compete well with amplitude amplification applied to these random walk algorithms and hence we proposed applying heuristic techniques in order to fairly compare this quantum analogue with actual DPLL algorithm. Since this is a complete solver without any algorithmic analysis, implementing the algorithm is the only way to compute the time scaling of this algorithm. This quantification seems a hard problem without access to a large scale quantum computer.

Chapter 4

Derivative Free Optimization

This chapter deals with optimization of deterministic functions with the assumptions that the derivatives of the functions are not available and the output of functional values are provided by a "black-box" whose inner working is completely unknown. The framework of the optimization problem considered is a constrained *minimization* or *maximization* with the following structure:

$$\begin{aligned} &min\ f(x) \\ &subject\ to\ x \in \Delta \end{aligned} \tag{4.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be optimized with the vector x defining the n design parameters bounded by $\Delta = \{x \in \mathbb{R}^n | Lb \leq x \leq Ub\}$ where Lb, Ub are the lower and upper bounds for the design parameters. The assumptions made about the derivatives and the complex function evaluations are what makes this optimization interesting, since otherwise to find the optimum of the optimization problem, the gradients are computed which characterizes the critical points of the function. However, in most of these engineering problems, the objective function depends on the output of a numerical simulation of a physical process. These simulations are expensive to evaluate because they involve numerically solving large systems of partial differential equations governing the physical system. Hence, there is a need to define gradient-free methods which do not require the explicit calculations of gradients of the objective function $f(x)$ and instead optimize using only the values obtained by evaluating the objective function. Since the algorithm proceeds by computations involving function evaluations, the overall running time of such algorithms is dominated by the time taken per function evaluation by simulation. In this chapter, we are primarily concerned with optimizing the number of function calls made to the black-box/simulator using quantum techniques in order to reduce the overall running time of the algorithm.

4.1 Optimization techniques

The algorithms often employed to solve the optimization problem can be classified broadly into either *gradient based techniques* which require the use of derivatives of the objective function and *derivative free optimization* which do not require the use of derivatives. The preferred method for optimizing the problem depends on the engineering application. In section 4.1.1 we have mentioned a few gradient based techniques and in section 4.1.2 we present few prominent derivative free algorithms and finally state the limitations and advantages of derivative free methods compared to gradient based algorithms in section 4.1.9.

4.1.1 Gradient based techniques

Gradient based techniques are algorithms that use the derivatives of the objective function to solve the optimization problem. If the gradients of the objective function with respect to all the variables are known, these techniques are known to be more efficient and converge faster compared to gradient-free techniques. Few well known gradient based optimization techniques often applied to problems include gradient descent, conjugate gradient, quasi-Newton method and sequential quadratic programming (SQP) [BT95] which is used to handle non-linear constraints in these problems. Often in engineering problems which employ these techniques, alternatives are used to compute the gradients, such as adjoint-based methods or numerical finite differences techniques, to reduce the computational effort for computing gradients. The central idea behind numerical finite differences is the following equation

$$\frac{\partial f}{\partial x_i} = \lim_{\delta_i \to 0} \frac{f(x_i + \delta x_i) - f(x_i - \delta x_i)}{2\delta x_i}, \qquad i = 1, \ldots, n \tag{4.2}$$

used to compute the partial derivative of a function with respect to all variables, where n is the number of variables and δx_i is the perturbation size. The drawback of the numerical finite differences technique is that it requires more (approximately twice) function evaluations to compute the gradients of the objective function with respective to all variables. Since, function evaluation is considered expensive, even if derivatives are available in many engineering problems, this technique is generally not employed when the number of variables in the optimization problem are high. An efficient alternative is the adjoint-based technique, which involves computing the Jacobian matrix and Lagrange multipliers for the objective function. Details of this technique have been omitted here; an interested reader is referred to [Ise09], [Sar06] for this method of computing gradients. The drawback in computing gradients is that it requires information about the black-box, such as the gradients of the objective function with respect to other constraint variables. Hence this technique of computing gradients with respect to control variables for optimization is useful with complete knowledge of the objective function. This is often not the case

in engineering problems where simulations (or black-boxes) often rely on legacy or proprietary codes which are not easily accessible.

4.1.2 Derivative free techniques

Derivative free techniques, unlike the gradient-based techniques, do not require the explicit computations of gradients and use just the function evaluations from the black-box. There have been many algorithms which have been described in the literature for derivative free techniques since its conceptualization in 1960. These algorithms can be sub-categorized into deterministic techniques such as the Nelder-Mead method, the generalized pattern searching method and heuristic algorithms such as genetic algorithms, tabu search, particle swarm optimization, etc. In this section, we give a brief introduction to a few of the prominent techniques. There have been several papers [RS12],[MW09] comparing derivative-free algorithms for optimization problems that arise in engineering applications, the most prominent one being [FRK+08] where six derivative free algorithms have been compared for the groundwater community problem. An interested reader is also referred to the first book [CSV09] dedicated to the subject of derivative free optimization by Conn, Scheinberg and Vicente.

4.1.3 Hooke-Jeeves Algorithm

The Hooke-Jeeves direct search algorithm [HJ61] was the first direct search algorithm proposed in 1961 for optimization problems by employing a simple search strategy to find the optimum. The algorithm initially chooses an initial point and evaluates the objective function. The search phase is based on a two moves, *exploratory moves* and *pattern moves*. The algorithm begins with the exploratory moves, searching in all directions with a predefined step length to check if the objective function takes a lower value. If such a lower evaluation occurs, the algorithm moves to that point. If no decrease is found, then the step size is reduced and the exploratory moves step is repeated. Pattern moves occur if a decrease occurred during the exploratory moves step, in which case the algorithm goes in the same direction as the previous decrease step with the same step size to check if the new point has a functional value lesser than the current iterate. If not, then the algorithm reverts back to the exploratory moves step. The algorithm stops after a predefined number of steps or if the predefined tolerance level for the step size is achieved.

The Hooke-Jeeves algorithm is inherently sequential in nature where the current search direction and step size depends entirely on the previous iterate. If the exploratory move succeeds, the pattern moves step is performed and if pattern move fails the exploratory moves step is carried out and correspondingly the step size changes. It can also be noticed that the algorithm

doesn't involve gradient computations and depends only on function evaluations from the black-box to converge to the optimum point. This technique of robust searching for the optimum is advantageous and has proven to be significantly fast. However, the disadvantage of this algorithm is that it doesn't parallelize easily and hence this algorithm is preferred if the entire optimization algorithm is performed on a single core and a cluster of systems is not at our disposal.

4.1.4 Nelder-Mead Algorithm

The Nelder-Mead algorithm [NM65] was proposed in 1965 and is still widely used in various libraries of functions for software. The algorithm begins by defining a set of initial points which form a simplex (a polytope covering all the chosen points). In each successive iteration the algorithm evaluates the function at all the points in the simplex and determines the worst corner point. Once this point is discovered, the algorithm replaces this worst point by another vertex creating a new simplex through a set of predefined operations in such a way that the new simplex would optimistically give optimized functional values. The operations that are employed to find this new point are reflection about the centroid, expansion, inside contraction, outside contraction or the shrink step (wherein all the points of the simplex are replaced by a new polytope). All these steps are shown in figure 4.1. To prevent stagnation in the algorithm, Kelley in [Kel99] proposed to enforce a *sufficient decrease* condition determined by an approximation of the gradient. If the algorithm stagnates, it restarts from a completely different simplex. A drawback of the Nelder-Mead algorithm is the intractability as search dimension n increases. In fact it was shown [SS99] that for moderate search spaces with dimensions $n \geq 10$ the algorithm becomes inefficient and hence is not preferred when the search space is big.

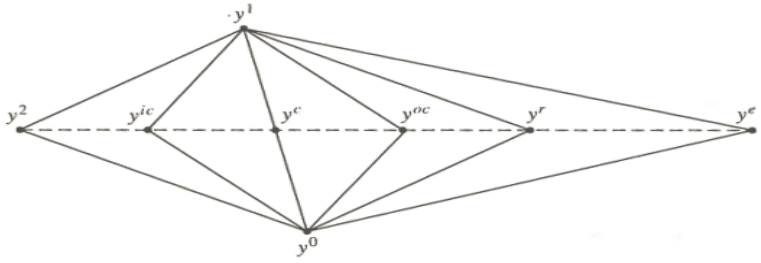


Figure 4.1: [CSV09] Illustration of a simplex drawn around the points y_0, y_1, y_2 with y^{ic}, y^{oc}, y^r, y^e representing the inside contraction, outside contraction, reflection and expansion points of the simplex about the centroid y^c.

The above diagram represents the possible set of operations which could be repeated in every step of the Nelder-Mead algorithm to determine the new point of the simplex once the worst point is found. Keeping the figure in mind, we describe the Nelder-Mead algorithm below.

1. **Initialization**:
   Define: Simplex of $n+1$ vertices $X_0 = \{x_0^0, x_0^1, x_0^2, \dots x_0^n\}$.
   Define: $0 < \gamma^s < 1$, $-1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e$
   where δ_r is reflection parameter, δ_e is expansion parameter, δ_{oc} is outside contraction parameter, δ_{ic} is inside contraction parameter.

For k=0,1, . . . ,n

2. **Evaluation Step:** Evaluate f on all the points in X and arrange the points in X such that
$$f(x^0) \leq f(x^1) \leq f(x^2) \leq \ldots \leq f(x^n)$$
3. **Reflection Step:** Reflect x^n about the centroid(x^c) of the remaining n points in X. Evaluate the f at x^{ref}
$$x^{ref} = x^c + \delta^r(x^c - x^n)$$
   If $f(x^0) \leq f(x^{ref}) \leq f(x^{n-1})$ replace x^n by x^{ref} and generate the new simplex X_{k+1}
4. **Expansion Step:**
   - If $f(x^{ref}) < f(x^0)$ evaluate $f(x^{exp})$
   $$x^{exp} = x^c + \delta^e(x^c - x^n)$$
   If $f(x^{exp}) \leq f(x^{ref})$ replace x^n by x^{exp} and generate the new simplex X_{k+1}.
   - **Else** Replace x^n by x^{ref} and generate X_{k+1}.
5. **Contraction Step:**
   If $f(x^{ref}) \geq f(x^{n-1})$, perform either of the following contractions:
   (a) *Outside Contraction:* **If** $f(x^{ref}) < f(x^n)$, evaluate $f(x^{oc})$
   $$x^{oc} = x^c + \delta^{oc}(x^c - x^n)$$
   If $f(x^{oc}) \leq f(x^{ref})$, replace x^n by x^{oc} and generate X_{k+1}. **Else** Perform Shrink Step
   (b) *Inside Contraction:* **If** $f(x^{ref}) \geq f(x^n)$, evaluate $f(x^{ic})$
   $$x^{ic} = x^c + \delta^{ic}(x^c - x^n)$$
   If $f(x^{ic}) < f(x^n)$, replace x^n by x^{ic} and generate X_{k+1}. **Else** Perform Shrink Step
6. **Shrink Step** Evaluate f at the points x^{new}
$$x_i^{new} = x^0 + \gamma^s(x^i - x^0), \qquad i = 1, \dots n$$
   and replace the points in the original simplex X_k by these new points x_i^{new} to generate X_{k+1}

End Loop
Output: Evaluate the points on simplex X_n and output the point, where the function is minimized.

Figure 4.2: Nelder-Mead algorithm

4.1.5 Generalized Pattern Searching algorithm

Torczan, in 1997, defined and analyzed the *Generalized pattern search* [Tor97] algorithm for derivative free optimization problems. This pattern search was the first algorithm to be proposed with a rigorous proof of convergence to the optimal solution. The global convergence of these pattern search algorithms for constrained as well as unconstrained optimization was established in [Tor97], and the local convergence properties of these pattern search algorithms was shown in [DLT03]. The convergence analysis of these proofs are heavily reliant on the theory of positive bases and has been omitted in this thesis, an interested reader is referred to the respective papers.

The primary components of the algorithm can be divided into an optional *search step* and a required *poll step*. The *search step*, involves evaluating finite number of points on a mesh M_k and searching for a point at which the function $f(x)$ obtains its minimum. The search step could be thought of as a step to guide the algorithm in a direction where the optimum could be found, effectively increasing the efficiency of the algorithm. Since, the convergence of the algorithm is completely independent of this step, the user has the freedom in speeding up this step to improve the time taken for function evaluations. The *poll step* is invoked in the case that the search step doesn't find a point in the mesh M_k with functional value lesser than the present iterate. Before explaining the poll step of pattern search algorithm, we need an understanding of positive spanning basis.

4.1.5.1 Positive Spanning Basis

The notion of a positive spanning set, introduced in [Dav54] by C.Davis, is an integral part of derivative free optimization. The motivation behind using this basis set is, given a non-zero vector $\vec{v} \in \mathbb{R}^n$, there is at least one vector $\vec{d}$ in the positive spanning basis which forms an acute angle with $\vec{v}$. In optimization this translates to the following, if $\vec{v}$ is the negative gradient of a continuously differentiable function i.e., $\vec{v} = -\nabla f(x)$, any vector $\vec{d}$ that forms an acute angle with $-\nabla f(x)$ is a direction of descent. This is interesting since by evaluating a finite number of directions it can be determined if a point is a local mesh optimum, instead of considering all possible directions.

Definition 6. A positive spanning set [CSV09] is a set of vectors whose linear combination spans $\mathbb{R}^n$ with non-negative integer coefficients. Positive span of a set of vectors $[v_1, \ldots, v_r]$ in $\mathbb{R}^n$ is the convex cone formed by all the positive linear combinations of $[v_1, \ldots, v_r]$. Hence the span of such a positive basis could be written as

$$\{v \in \mathbb{R}^n : v = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_r v_r | \alpha_i \geq 0\}. \tag{4.3}$$

Definition 7. [CSV09] A descent direction of a continuously differentiable function f at a given point $x \in \text{dom}(f)$ is a direction $\vec{d}$ where there exists $\overline{\alpha}$ such that $f(x+\varepsilon\vec{d}) < f(x)$ for all $\varepsilon \in [0, \overline{\alpha})$.

Considering the case of 2 variables that is, $\mathbb{R}^2$ the following figure shows the possible combinations for a positive spanning bases.

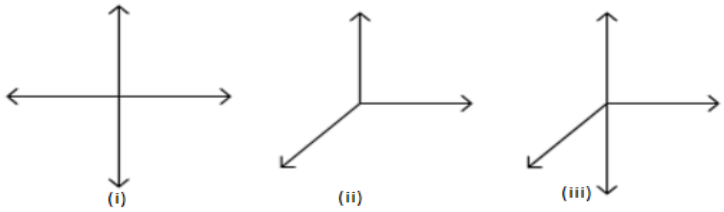


Figure 4.3: [CSV09] Illustration of positive bases for $\mathbb{R}^2$: The leftmost figure (i) represents the maximal positive basis constructed from the basis vectors (b_i) and their negative counterparts ($-b_i$). The middle figure (ii) represents the minimal positive basis which can be constructed by the basis vectors and their sum. The rightmost figure (iii) could be obtained by adding another vector to the positive basis resulting in a positive spanning set which is not a basis.

We will come back to this figure and show how this is useful in optimization later. A few simple properties of positive spanning basis are established in the following theorem, which will be important in the poll step of the GPS algorithm.

Theorem 8. *Given that $[v_1, ..v_r]$ spans $\mathbb{R}^n$ positively, with $v_i \neq 0$ for all $i \in \{1, 2, \dots r\}$, it follows that*

1. *$[v_2, ..v_r]$ spans $\mathbb{R}^n$ (or it contains a subset of $r-1$ vectors spanning $\mathbb{R}^n$).*
2. *There exists scalar quantities $\alpha_1, \alpha_2, \dots \alpha_r$ with $\alpha_i > 0$, $i \in \{1, 2, \dots, r\}$ such that $\sum_i \alpha_i v_i = 0$.*
3. *Given a non zero vector $x \in \mathbb{R}^n$, there always exists an index $i \in \{1, 2, \dots, r\}$ such that $x^T v > 0$.*

Proof. 1. Since it is given $[v_1, \dots, v_r]$ spans $\mathbb{R}^n$ positively, it follows that a vector $-v_1$ can be written in terms of the positive basis with positive coefficients as $-v_1 = \sum_i \beta_i v_i$. Subtracting $\beta_1 v_1$ from both, $-(1+\beta_1)v_1 = \sum_{i \neq 1} \beta_i v_i$. Since $\beta_1 \geq 0$, dividing the entire expression by $-(1+\beta_1)$, we obtain

$$v_1 = -\sum_{i \neq 1} \frac{\beta_i v_i}{1+\beta_1}. \tag{4.4}$$

This shows that v_1 is in the span of all the other vectors $[v_2, \ldots, v_r]$ and hence this subset spans $\mathbb{R}^n$. It also follows that for every $i \in \{1, 2, \ldots r\}$, the vector $-v_i$ can be written in terms of the remaining $r-1$ vectors in the positive basis and hence there always exists a subset of $r-1$ vectors spanning $\mathbb{R}^n$.

2. In order to prove this statement, we employ the result 1, or given $[v_1, ..v_r]$ spans $\mathbb{R}^n$ positively, for all i we can construct $-v_i$ as a span of the other $r-1$ vectors. It follows that

$$v_k + \sum_{i \neq k} \frac{\beta_i v_i}{1+\beta_k} = 0, \qquad k = 1, \ldots, r. \tag{4.5}$$

By adding all these equations,

$$\left(1 + \sum_{i \neq 1} \frac{\beta_i}{1+\beta_i}\right) v_1 + \left(1 + \sum_{i \neq 2} \frac{\beta_i}{1+\beta_i}\right) v_2 + \ldots + \left(1 + \sum_{i \neq r} \frac{\beta_i}{1+\beta_i}\right) v_r = 0. \tag{4.6}$$

Since $\beta_i \geq 0$, setting $\alpha_j = 1 + \sum_{i \neq j} \frac{\beta_i}{1+\beta_i}$, statement 2 follows.

3. Let us consider a non-zero vector $x \in \mathbb{R}^n$, since $[v_1, \ldots, v_r]$ spans $\mathbb{R}^n$ positively,

$$\begin{aligned} x &= \alpha_1 v_1 + \alpha_2 v_2 + \ldots \alpha_n v_n \\ x^\perp x &= (\alpha_1 v_1 + \alpha_2 v_2 + \ldots \alpha_n v_n)^\perp x \\ &= \alpha_1 v_1^\perp x + \alpha_2 v_2^\perp x + \ldots \alpha_n v_n^\perp x \end{aligned} \tag{4.7}$$

where $\alpha_i \geq 0$ for all i. Since $x^\perp x > 0$, it implies that there exists i such that $\alpha_i v_i^\perp x > 0$. Hence at least one of the scalars $x^\perp v$ has to be positive.

□

Property (iii) from Theorem 8 is at the heart of convergence of direct search methods in derivative free optimization. Given a continuous smooth function which is differential at a point x ($\nabla f(x) \neq 0$), there always exists a vector $\vec{d}$ which is part of a positive spanning set such that

$$-\nabla f(x)^\perp \vec{d} > 0. \tag{4.8}$$

Put simply, a positive spanning set contains at least one direction of descent. Any vector is guaranteed to have a positive projection (at a non-stationary point in the domain of the function) along at least one of the vectors in the set of positive bases. As long as the current iterate is not a local optimum point, one of the directions of future polling from the set of points D_k will be a direction of descent for the objective function. Figure 4.4 explicitly shows how there always exists a direction of descent for any vector $\vec{v}$, given a positive basis in $\mathbb{R}^2$ as shown in figure 4.3.

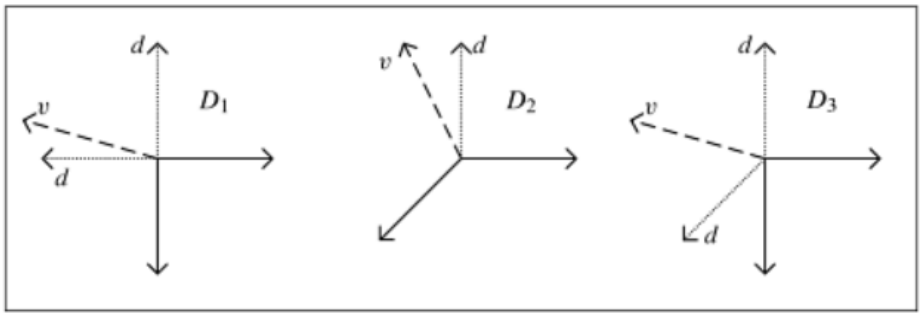


Figure 4.4: [CSV09] Given a positive spanning set in figure 4.3 for $\mathbb{R}^2$ and a vector $\vec{v} = -\nabla f(x)$, there always exists at least one element in the set $\vec{d}$ such that $\langle \vec{v}, \vec{d} \rangle > 0$.

In [Dav54], it was proven that if the linearly independent vectors $[v_1, \ldots, v_r]$ span $\mathbb{R}^n$ positively, then the number of elements in a positive basis cannot be more than $2n$ and the minimal number of elements for a positive basis must be $n+1$. The maximal positive basis is obtained by determining the basis for $\mathbb{R}^n$ along with the negative counterparts of these respective basis vectors and the minimal positive basis can be constructed by the basis vectors of $\mathbb{R}^n$ along with the vector formed by the negative sum of all the basis vectors. Quantifying the number of elements in the positive bases is important since it places a bound on the finite number of directions that need to checked before deciding if a point is a local mesh optimizer.

Returning to the algorithm, we recall that if the search step fails to find a descent among the points in the mesh M_k, the poll step is evoked. In the poll step, the algorithm first creates a positive spanning basis at the present iterate. It evaluates the function on the set of directions defined by the positive spanning basis in order to evaluate if there exists a point with lower functional value on the poll set. If no such point exists, the present iterate under consideration is the local mesh optimizer. If the tolerance criterion is however not met, then the step size is decreased and the search and poll step repeat. We shall now present the basic framework of the *generalized pattern searching* algorithm,

1. **Initialization**:
   Define $x_0 \in \mathbb{R}^n$ (such that $f(x_0)$ is finite), tolerance value ε, parameters $\tau \geq 1$ and $0 \leq \gamma < 1$
   Define M_0 to be the mesh on $\mathbb{R}^n$ and let the step size be $\alpha_0 > 0$.
   Define $\mathcal{D}$ to be a matrix whose columns are the positive bases or spanning set in $\mathbb{R}^n$, in general we could define the mesh

$$M_k = \{x_k + \alpha_k \mathcal{D} z | z \in \mathbb{Z}^{n_d}\}$$

   where n_d is the number of columns in the matrix $\mathcal{D}$ and α_k is the mesh size parameter.

For k=$0, 1, 2, \ldots, k_{max}$

2. **Search Step**:

   Search on a trial set of points on the mesh M_k and if an improved point is found i.e. $f(x_i) < f(x_k)$, then the *SEARCH* is successful. Goto *Update Parameters*. If the *Search* fails, goto *Poll Step*.

3. **Poll Step**:

   If the Search step fails, choose a set of positive spanning directions and form the poll set M_{poll} as the set of points adjacent to x_k in the corresponding directions. This can be formalized by defining a positive spanning matrix $\mathcal{D}_k$ whose columns consist of basis vectors of the positive spanning set. The mesh can be written as

   $$M_{poll} = \{x_k + \alpha_k d | d \in \mathcal{D}_k\} \subset M_k.$$

   Evaluating $f(x_j)$ on M_{poll} if an improved point $x_{poll} \in M_{poll}$ is found, i.e. $f(x_{poll}) < f(x_k)$, then *Poll Step* is successful, goto *Update Parameters*. If for all points on the mesh $x_{poll} \in M_{poll}$, we cannot find a point which evaluates lesser than the present iterate, $f(x_{poll}) > f(x_k)$, then the *Poll Step* is unsuccessful and goto *Update Parameters* .

4. **Update Parameters**:

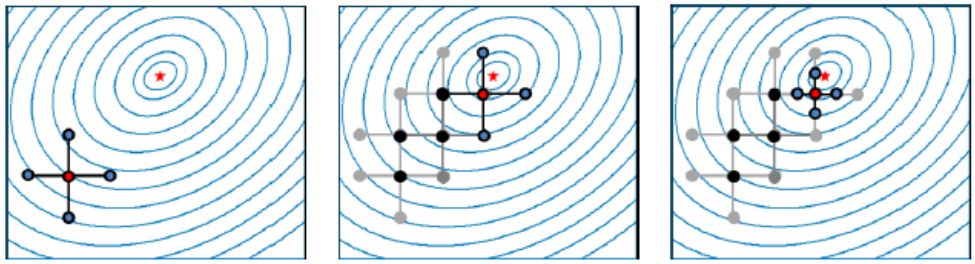
   - If the search step is successful, then increment k, update $x_{k+1} \leftarrow x_i$, $\alpha_{k+1} = \tau\alpha_k$ and Perform **Search Step** again.
   - If the search step is unsuccessful and poll step performed is successful, then increment k, $\alpha_{k+1} = \tau\alpha_k$, and perform **Search Step** again.
   - If the Poll Step is not successful then x_k is the local mesh optimizer. If the tolerance condition is satisfied, $STOP$ and output x_k, if not increment k refine the mesh, $\alpha_{k+1} = \gamma\alpha_k$, and restart **Search Step** from this point.

End Loop

Figure 4.5: Generalized pattern searching algorithm.

In figure 4.6, the polling step in the GPS algorithm is demonstrated, giving us a picture of how the algorithm progresses with each iteration. The algorithm begins by searching on a mesh until it reaches a point where the objective value cannot be lowered further. The polling step begins thereafter by evaluating the objective function at points in the poll directions in each iteration. Initially the step size for polling is large, but as the algorithm proceeds the step size contracts in order to reach the local optima and the algorithm stops when the step size is below the pre-defined tolerance level.

The poll step in the GPS algorithm 4.5 is an example of complete polling. *Complete polling* refers to evaluating all the points on the poll bases before deciding the point to which the present iterate should move to, whereas the other variant is *opportunistic polling* where the present iterate moves to the first point on bases where the function takes a value lesser than the present



(a) Initial polling stencil (b) Successive polling iterations (c) Contraction of stencil

Figure 4.6: [Ise13] Blue lines represent the contours of the objective function and the red star denotes the local optima, with the crosses representing how the algorithm moves in each step. The poll points representing the positive basis are the corners of the cross centered by the red circle which is the poll center.

iterate. Complete polling has been shown to be more reliable and robust, and is often preferred in algorithms which have an access to a cluster of processors since the points can be individually evaluated on parallel machines.

4.1.6 Mesh Adaptive Direct Search

Mesh adaptive direct search (MADS) proposed in [ADJ06] is a modified version of the generalized pattern searching algorithm. In GPS, the algorithm begins with an initial point, evaluates the function on finite number of points on the mesh around the chosen point, to check if the value of the function decreases. If the search step fails, the algorithm evaluates in the direction of the positive spanning basis. If a lower evaluation is encountered the algorithm moves to the new point, evaluates the poll basis at this point and carries on. If a point with lower functional value is not found, then the search radius is reduced and algorithm begins with the search step. If the tolerance level is satisfied then the algorithm stops. However in MADS, there are two parameters, the *poll size* parameter which restricts the points from which the poll basis is selected and the *mesh size* parameter which defines the grid within the region defined by the poll size parameter. Both these parameters are equal for the GPS algorithm and represented by the mesh size parameter α_k. Hence the number of positive spanning sets that can be formed by subsets of the positive spanning matrix $\mathcal{D}$ is constant in all iterations of GPS. MADS incorporates dynamic reordering in the poll step to consider a variable set of poll directions where the poll points of successive iterations are dependent on the poll directions of the previous successful iterations. The advantage is that the union of all poll directions over all iterations is dense in $\mathbb{R}^n$, meaning that potentially every direction is explored eventually at the end of the algorithm. This dynamic reordering provides a more robust convergence to this pattern searching algorithm compared to GPS algorithm.

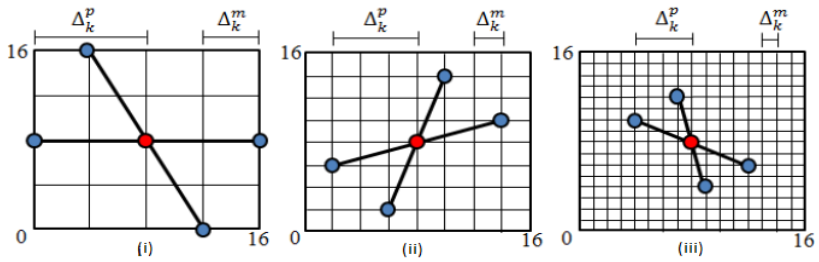


Figure 4.7: [Ise13] The poll points representing the positive basis are the edges of the cross around the red circle which is the poll center. Fig (i) is the MADS at iteration k with poll stencil size Δ_k^p=8 and mesh size Δ_k^m=4, successively as the iterations carry on the mesh size is halved and poll stencil size is reduced by $\sqrt{2}$ in figure (ii) at iteration $k+2$ has poll stencil size Δ_k^p=5.67 and mesh size Δ_k^m=2, and in figure (iii) at iteration $k+2$ the poll stencil size is Δ_k^p=4 and mesh size Δ_k^m=1.

In figure 4.6, since the orientation of the stencil is fixed it is called the generalized pattern searching algorithm. However the variation in MADS as shown in figure 4.7 is that the orientation of the successive poll step depends on the previous iteration and hence can adaptively vary from step to step. Hence the polling is effectively done in a dense set of directions as shown in figure 4.8 and consequently the iterates have access to more possible polling directions.

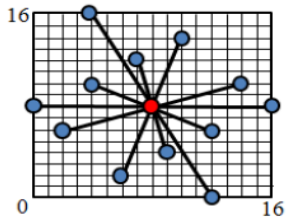


Figure 4.8: [Ise13] Union of poll directions in iteration k, $k+1$, $k+2$.

In [ADJ02], Audet and Dennis showed that the convergence of this algorithm requires only the assumption that f is continuously differentiable with a Lipschitz [1] derivative [2] near the limit point. They also showed that with an additional assumption of f being twice strictly differentiable near the limit point, MADS converges to a local minimizer with probability 1.

The choice between GPS and MADS for employing derivative free optimization is dependent on the engineering problem considered. In production optimization for oil refineries considered in [Ise09] the GPS and MADS had similar performance, and hence GPS was considered. However in

[1] Lipschitz function is defined as a function $f(x)$ with $|f(x) - f(y)| \leq C\,|x - y|$ for all x, y and a constant C.

[2] If the function f is defined on domain $\mathcal{D} \in \mathbb{R}$ and f' is continuous on $\mathcal{D}$, there exists a neighbourhood $\mathcal{V} = [\alpha - \varepsilon, \alpha + \varepsilon]$ of the limit point α such that f' is Lipschitz on $\mathcal{V}$.

another problem of generalized field development optimization in [Ise13] MADS was considered for the pattern searching algorithm. We have also considered a problem in civil engineering of installing dampers between adjacent buildings [Big12] where the MADS algorithm is preferred compared to the GPS. In [ADJ06], Dennis and Audet consider a number of problems wherein generalized pattern search algorithm stagnates and MADS converges to an optimal solution. Also in [AA06], they give examples of problems where it was shown that the GPS algorithm stalls at saddle points whereas MADS escapes and converges to the local minima, which can be primarily attributed to having a dense set of polling directions as illustrated in figure 4.8.

4.1.7 Surrogate Management Frame

Engineering problems in general consider the derivatives of the function to be computationally expensive to calculate. In many practical applications, evaluating the function depends on the output of a numerical simulation and is expensive since it is dependent on many optimization variables that must be evaluated before evaluating the function. Employing surrogates is a computational enhancing technique wherein this expensive objective formula is replaced by surrogates. Typically surrogates have lower accuracy or have lesser quality than the objective function but are cheaper to evaluate or consume fewer computing resources. These surrogates are evaluated instead of the original objective function and in effect the most time consuming step in the algorithm, i.e. the formula evaluation, is simplified. Often the surrogate management framework can be divided into two categories, one wherein the entire physical model itself is simplified to evaluate f or secondly, approximating f which can be obtained by evaluating f at a finite number of points and interpolating or smoothing the function values obtained. Often the second technique is employed in computational expensive problems. The surrogate management framework in other aspects is similar to the GPS algorithm except that the function evaluation black-box is replaced by surrogates for cheaper evaluation.

4.1.8 Heuristics

Heuristics such as genetic algorithms do not require the explicit computations of gradients and are good candidates for stochastic derivative free algorithms for optimization. In genetic algorithms, the algorithm generates a set of possible candidates called a population and evaluates the objective function on these individuals. Based on the evaluations, the algorithm ranks the elements and performs operations such as selection, mutation, crossover to the population. Selection is process of discarding the worst possible solutions and choosing the best for the next iteration, thereby allowing the next iteration to contain candidate solutions which are more likely to be the optimum of the optimization problem. Mutation is the process of randomly choosing one bit of

the candidate and probabilistically flipping that bit, giving the algorithm an exploratory nature where it evaluates elements randomly to check if a better solution can be obtained. Crossover is the process of combining two solutions that were ranked high in the selection step, allowing subsequent iterations to contain candidates which are more likely to be optimal solutions than previous iterations.

It should be noted that GA performs well in finding approximate solutions to all types of engineering problems since the algorithm doesn't make any assumption about any characteristics of the underlying objective function. The algorithm also does not require the evaluation of the gradients for the optimization problem, since it proceeds by simply evaluating the function through the black-box. The stopping criterion of genetic algorithms is generally predefined by the user as the number of generations in the algorithm in which the objective function doesn't have any improvement. The drawback of genetic algorithms is however, given a limited number of predefined generations there is no guarantee the solution to the problem can be found. There is also no absolute assurance that heuristics will find the global minimum when the number of variables is high and the function is complex.

4.1.9 Advantages and Limitations

The common advantage in all derivative free techniques is the fact that the gradients of the objective function are not required for the optimization of the problem, which is essential for problems where the derivatives of the objective function are expensive to calculate. Another characteristic of GPS is the fact that the algorithm parallelizes naturally since the function evaluations on the set of search and poll points on the mesh M_k can be evaluated on parallel processors in a cluster. Hence the number of black-box evaluations in each processor would naturally reduce to $O(n/k)$ with k processors. In fact in the production optimization and well optimization problem discussed in section 5.2, the MADS algorithm is completely parallelized with the number of processors allocated for each iteration equaling twice the number of variables in the problem or the maximum number of poll points to be evaluated in each step. Another interesting advantage of derivative free techniques is that they can escape local minima since they consider a large step size in the start and reduce as the algorithm progresses whereas gradient based methods would get trapped in such local minima. Employing larger step size in the algorithm allows pattern search algorithms to move to search spaces which are more conducive for refined searching. Thereafter, we could search in the poll directions to find the minima in a restricted region by reducing the step size and evaluating the function on the poll points.

However, on the flipside when considering these algorithms applied to bound constrained optimization, derivative free techniques require an order of magnitude more function evaluations

than gradient based techniques, which is the most time consuming step in the algorithm. Another drawback with derivative free techniques is the cost of implementing these algorithms scale with the number of variables involved in the optimization problem. Derivative free algorithms for problems which involve over 150 variables are practically impossible to implement without any computational enhancing techniques. This inefficiency was significantly improved however through multi-core computing and surrogate management framework as discussed in section 4.1.7 which are often used to reduce the computational cost.

4.2 Quantum Speedup in computational cost

4.2.1 General Overview

The running time analysis of derivative free optimization algorithms is hard to formulate, because it is a tool which can be adapted to different applications. Effectively the overall running time of the algorithm is dominated by the time taken for each function evaluation which is dependent on the problem considered. In most of the applications, evaluation of the function is considered as a black-box to which queries are made for each functional evaluation in the algorithm. Hence, we formulate the number of the queries made to the black-box instead of the time complexity of the algorithm. Considering a problem with n variables in the grid $\mathbb{R}^n$, typically in each iteration, the first *search step* involves global minimization by evaluating approximately $O(n)$ points on the mesh M_k. The second *poll step* evaluates the function in the poll directions, which in the worst case involves evaluations at $2n$ points (there are at most $2n$ basis vectors in the positive spanning set as discussed in section 4.1.5.1) of the mesh M_{poll}.

The number of iterations k_{max} in algorithm 4.5 depends on the application considered. In areas of refinery optimization it is parametrized by the time the algorithm is run and tolerance is predefined during the instantiation of the algorithm, whereas in areas of molecular geometry the stopping criteria is determined by the energy tolerance of the final structure of the molecule and there are few problems which run the algorithm till the step size in the algorithm is below the tolerance. Overall with k_{max} iterations, the total number of black-box queries for the pattern searching algorithm is $O(nk_{max})$.

4.2.2 Quantum Speedup

In MADS/GPS algorithms, each iteration of the algorithm requires $O(n)$ black-box queries to find the poll direction where the function is minimized. Considering that each black-box evaluation takes an hour, as the problem gets complicated with more number of variables n, the

number of queries increases. If either the time taken per black-box evaluation or the number of black-box evaluations can be reduced, it would improve the overall running time of the entire algorithm. Reducing the time taken per evaluation requires knowledge of the objective function, but since the objective function is considered a black-box we consider the optimization of the number of black-box evaluations. The algorithm in each search step and poll step evaluates a finite number of points on the mesh to find the point at which the function is locally minimized. We employ the *quantum minimum finding* algorithm described in section 2.2.1 in both these steps to improve the number of queries made to the black-box. In the search step, we create a superposition of search points and obtain the minima on the mesh M_k in $O(\sqrt{n})$ black-box evaluations. In the complete polling step, the poll directions are computed and superposition of poll points is created. The black-box is queried $O(\sqrt{n})$ times to find the point at which the functional value is minimum on the mesh M_{poll}.

For iteration k

1. **Search Step**:
   - Prepare superposition of n search points from mesh M_k as $\sum_{i,k} |x_i^s\rangle$
   - One query of the black-box would result in $\sum_{i,k} \left|x_{i,k}^s\right\rangle \left|f(x_{i,k}^s)\right\rangle$, using *quantum minimum finding* subroutine evaluate the black-box $O(\sqrt{n})$ times.
   - Measure the first register to obtain the search point where the minimum can be found.
   - If yes goto *Update Parameters*.
   - If the *Search* fails, goto *Poll Step*.
2. **Poll Step**:
   If the Search step fails, choose a set of positive spanning directions and form the poll set M_{poll}.
   - Prepare superposition of poll points (note the number of poll points can be between $n+1$ to $2n$) from positive spanning directions as $\sum_{i,k} |x_i^m\rangle$.
   - One query of the black-box would result in $\sum_{i,k} \left|x_{i,k}^m\right\rangle \left|f(x_{i,k}^m)\right\rangle$, using *quantum minimum finding* subroutine evaluate the black-box $O(\sqrt{n})$ times.
   - Measure the first register to obtain the poll point $\left|y_{i,k}^m\right\rangle$ where the minimum can be found.
   - If $f(y_{i,k}^m) < f(x_k), \forall x_k \in M_{poll}$ goto *Update Parameters*.
   - If *Poll Step* is unsuccessful, goto Update Parameters.

Figure 4.9: Quantum speedup in Search and Poll step in GPS.

In algorithm 4.9, the black-box is queried quadratically fewer number of times compared to the classical GPS algorithm by employing quantum subroutines in the search and poll step. It is important to note that this speedup only reduces the number of queries made to the black-box and doesn't have any influence on the iterate x_k after each iteration. This is integral since the

convergence analysis of all pattern search algorithms depend primarily on the poll step of the algorithms. The probability of error during the measurement of the final superposition in the search and poll step in the above algorithm can be reduced by parallel repetition of the algorithm as mentioned in section 2.2.1. Effectively, without affecting the convergence of the pattern search algorithm, using quantum techniques the number of queries to the black-box can be quadratically reduced, which is integral for engineering applications where evaluating the black-box is expensive and could take hours per evaluation.

4.3 Practical Flipside

Effectively using quantum search pattern search algorithms for derivative free optimization can be performed in $O(k_{max}\sqrt{n})$ black-box evaluations. Although, this is not a quadratic speedup to the entire protocol, this is significant considering the complexity of the derivative free algorithm depends on the time taken in the search and poll step, since the computational cost of the algorithm is dominated by the cost of evaluating the function. Hence evaluating the black-box quadratically lesser number of times would vastly improve the time scaling of the overall optimization problem.

On the flipside, this improvement of the time scaling wouldn't generalize to all problems which employ pattern search algorithm for DFO due to *complete parallelization*. Effectively, as it will be seen in section 5.2.5, for problems such as generalized field development optimization, for a grid size of $\mathbb{R}^{60}$ (with 60 variables) there are approximately 120 processors in the cluster dedicated to the search and poll step for each iteration. Hence by quadratically improving the number of queries to the black-box, *neither* the number of processors that could have solved the optimization problem can be reduced *nor* the overall time required by the algorithm to find the optimum. This scenario has been explicitly considered and analyzed in section 5.2.5 for the generalized field development problem. In this optimization problem, parallel quantum searching results in a marginal speedup when the grid size is huge ($\approx \mathbb{R}^{125}$) where the number of processors dedicated for the problem is limited. This flipside however doesn't apply to all algorithms, for example considering the problem of installing dampers in between adjacent buildings, it can be seen that the entire optimization is carried out on a relatively small cluster in which case a quadratic speedup to the number of queries, improves the time required for the overall running time. Hence in such problems the quadratic reduction in the number of queries provides a speedup. The exact numeric showing this improvement has been presented in section 5.3.4.1.

4.4 Summary

In this chapter, we have gone through various important classical derivative free algorithms which are employed in problems where the function evaluation as well as the derivative information is not available or computational hard to compute. In section 4.2 we proposed a quadratic speedup to the search step and the poll step in the algorithms, which reduced the number of queries made to the black-box in both steps. In section 4.3 we argued why the quadratic improvement to the number of queries doesn't directly translate to the reduction in time for problems which are solved using clusters and parallel processors. In the following chapter, we consider two practical applications of generalized field development optimization in section 5.2 and placement of dampers in between adjacent buildings in section 5.3. In the first problem the quadratic advantage is not observed primarily due to costly function evaluations and parallelization of the algorithm, whereas in the second problem a quadratic speedup can be significantly observed since the black-box evaluations are relatively cheaper requiring higher number of evaluations.

Chapter 5

Applications

5.1 Haplotype Inferencing

Genomics, the study of genes of organisms, has been an area of interest for many decades, motivated by the quest to understand the influence of genetic imperfections on diseases. The current research in this field is in developing the International HapMap project [GBH+03] which involves collection of as many human genes as possible, in order to help scientists get access to this database and analyze genes associated with any disease. With faster and robust architecture in the past few years, there is renewed interest and rapid development in the field of genomics. Before describing *haplotype inferencing* problem we shall go over few terms which will be used.

5.1.1 Biological definitions

DNA(Deoxyribonucleic acids) makes up the cellular structure of large complex molecules, consisting of two long double strands of nucleotides connected by weak bonds. *Nucleotides* are subunits of DNA sequences consisting of four different types of nucleobases Adenine(A), Cytosine(C), Guanine(G) and Thymine(T), where bonds in DNA are formed by base pairs of nucleotides (A and T or G and C). *Single nucleotide polymorphism (SNP)* is a DNA sequence variation occurring due to one of the nucleotides (A,T,G,C) being altered between different organisms. Variants in the DNA sequence caused by SNP determine how humans react to diseases and treatment. A *gene* is a locatable region of genomic sequence which accounts for the inheritance of characteristics from one's parents. The variations in the genes are called *alleles*, giving rise to different characteristics in individuals. At a genomic level, in diploid organisms, there are two copies of each chromosome which are received from each parent. The genetic constitution of each of the chromosomes is

called a *haplotype* and the conflated(mixed) data present in both the chromosomes is called the *genotype*.

5.1.2 Problem and complexity

Researchers are interested in haplotype data since it has been shown to be a better predictor of diseases as they have more information about gene alleles compared to genotype data. Understanding this sequence would help in identifying and curing diseases which could have arisen from any one of the parents. However, determining this haplotype data experimentally is time-consuming and expensive while genotype data is much easier to get, hence genotypes rather than haplotypes are usually obtained. The haplotype inferencing problem is: *For a set of genotypes, find a smallest set of haplotypes, conditioned on each genotype being explained by a pair of haplotypes*. Given the heterozygous sites in a set of genotypes the objective is to determine which copy of a pair of chromosomes each gene allele belongs to. Answering the haplotype inferencing problem for required sizes of genotypes would have significant impact on the biomedical groups in USA, Canada, Africa, Europe [GBH+03] as well as organizations which are looking for drug cures for common diseases such as diabetes, cancer, cardiovascular disease, inflammatory diseases etc.

The computational version of this problem has been proven to be APX-hard (or considered hard to approximate) [LPR04] and consequently NP-hard as well. For a range of parameters of preliminary interest such as 50 genotypes with 50 sites, a solution can be computed efficiently with the help of integer programming. The computational intractability is noticed for larger parameters such as 50 genotypes with 100 sites where a near-parsimony solution becomes inefficient, but can be computed given a long time. However, Gusfield in [Gus04] showed that for 100 genotypes with 150 sites, the integer program approach is not feasible and Clark's algorithm which is often employed these days in the haplotype inference problem is inaccurate and requires many repetitions to find the satisfying set of haplotypes. Since the information of the haplotyes is critical in drug testing and are indicators of the diseases, it is integral to employ techniques which give high accuracy results. There were other polynomial size (in the size of the input) integer program formulations for haplotype inferencing problem using pure parsimony in [BH04] which improved on the results of Gusfield.

5.1.3 Reduction to SAT

In this section, the mathematical model in [LMS06] for haplotype inferencing is constructed before reducing it to an instance of SAT. The notation required for this model is presented first

before providing the clauses and variables in the SAT model, that need to be satisfied in order to solve the haplotype inferencing model.

5.1.3.1 Notation

Assuming we are given a set of n genotypes, $\mathcal{G}$, each of length m (the length corresponds to the number of single nucleotide polymorphism(SNP) sites). For convenience, each genotype could be referenced by $g_i \in \mathcal{G}$, with $i \in [1, \ldots, n]$ and the j^{th} site of the i^{th} genotype can be addressed as g_{ij}. Each genotype will be assigned one of the ternary variables $\{0, 1, 2\}$ for reasons mentioned later. The set of r haplotypes, $\mathcal{H}$, can be defined each of length m. As before the haplotype could be referenced by $h_i \in \mathcal{H}$ with $i \in [1, \ldots, r]$ and the sites in haplotype h_i can be referenced as h_{ij} with $j \in [1, \ldots, m]$. Each haplotype is assigned one of the binary variables $\{0, 1\}$. When referring to the haplotype or genotype as h_i, g_i, it refers to a string of m characters each representing the state of the respective sites in the haplotypes or genotypes.

From the previous section, it is known that two haplotypes from the parents combine to give a genotype present in the offspring. Since every individual has two *copies* of each chromosome, the haplotypes are assigned $\{0, 1\}^m$. Considering these combine to form a genotypes, the values assigned to genotype variables are $\{0, 1, 2\}^m$. Furthermore, when both the combining haplotypes have the assigned variables 0, the genotype has the resultant assignment 0, when both the combining haplotypes have been assigned 1, the genotype has resultant assignment 1. The ambiguity arises when the genotype has been assigned 2, which could arise from either of the haplotypes being assigned 1 and the other haplotype being assigned 0. With this terminology, we could state the Haplotype Inference by Pure Parsimony (HIPP) problem: to minimize the overall number r of haplotypes to solve the haplotype inferencing problem (i.e. each of the n genotypes being explained by a pair of haplotypes).

5.1.4 SAT Model

For a specific value of i specifying the genotype $g_i \in \mathcal{G}$, the aim is to minimize r (the number of candidate haplotypes) such that each g_i is explained by a pair of haplotypes. The model uses selector variables s^{a_1}, s^{a_2} which are strings of length r, each bit of the individual string assigned to each haplotype in $\mathcal{H}$. For example, if the genotype g_i is explained by two haplotypes h_{1i}, h_{2j}, then the corresponding selector variables $s^{a_1}_{h_{1i}}, s^{a_2}_{h_{2j}}$ are assigned the value 1 respectively and if they do not satisfy genotype g_i they evaluate to 0. Given a genotype, it should be noted that, only one bit from the selector variable strings s^{a_1}, s^{a_2} should be assigned to 1, which implies that only one haplotype each from sets a_1, a_2 satisfies the given genotype. The corresponding clause

for this condition can be framed as

$$\Big(\sum_{k=1}^{r} s_{ki}^{a_1} = 1\Big) \wedge \Big(\sum_{k=1}^{r} s_{ki}^{a_2} = 1\Big) \qquad i = 1, \ldots, n. \tag{5.1}$$

In order to specify each condition for $g_i \in \{0, 1, 2\}$, three cases have been considered and the corresponding SAT conditions have been sketched.

1. When g_i **is assigned to 0**: For any pair of haplotype to justify any particular genotypes their respective selector variables have to be assigned 1. It follows that if $g_i = 0$ and h_{1i}, h_{2i} justifies g_i, then the corresponding haplotype variables *must* definitely be 0 according to our definition. Thus, if a site $g_{ij} = 0$, it implies

$$(\overline{h_{kj}} \vee \overline{s_{h_{ki}}^{a_1}}) \wedge (\overline{h_{kj}} \vee \overline{s_{h_{ki}}^{a_2}}), \qquad k = 1, \ldots, r. \tag{5.2}$$

2. When g_i **is assigned 1**: Following definition, the haplotype variables *must* definitely be assigned 1 and the corresponding selector variables have to be selected to satisfy the genotype. Hence, if a site $g_{ij} = 1$ it implies

$$(h_{kj} \vee \overline{s_{h_{1i}}^{a_1}}) \wedge (h_{kj} \vee \overline{s_{h_{1i}}^{a_2}}), \qquad k = 1, \ldots, r. \tag{5.3}$$

3. When g_i **is assigned 2**: In this scenario there are two possibilities, either of the haplotypes can evaluate to 1, with the added constraint that both the haplotypes justifying a particular genotype must have different values. The corresponding selector variables have to be set to 1 as well in order to specify the haplotype selected. For this case, another two variables $g_{ij}^a, g_{ij}^b \in \{0, 1\}$ are introduced such that $g_{ij}^a \neq g_{ij}^b$. In terms of a clause it can be written as

$$(g_{ij}^a \vee g_{ij}^b) \wedge (\overline{g_{ij}^a} \vee \overline{g_{ij}^b}) \tag{5.4}$$

and the corresponding satisfiability equation representing this condition is

$$(h_{kj} \vee \overline{g_{ij}^a} \vee \overline{s_{ki}^a}) \wedge (\overline{h_{kj}} \vee g_{ij}^a \vee \overline{s_{ki}^a}) \wedge (h_{kj} \vee \overline{g_{ij}^b} \vee \overline{s_{ki}^b}) \wedge (\overline{h_{kj}} \vee g_{ij}^b \vee \overline{s_{ki}^b}) \qquad k = 1, \ldots, r. \tag{5.5}$$

The equations 5.1,5.2,5.3,5.5 concludes the SAT model to solve the haplotype inferencing problem. Given n genotypes, r haplotypes and m sites in each genotype and haplotype, assuming r_{opt} haplotypes are required to justify n genotypes, the number of clauses in the SAT model is $O(r_{opt}nm)$ and the number of variables are $O(r_{opt}m + r_{opt}n + nm)$. The worst case scenario corresponds to each genotype being explained by 2 *distinct* haplotypes or $|\mathcal{H}| = 2n$, when the number of clauses and variables in the SAT model reduces to $O(n^2m)$ and $O(nm + n^2)$ similar to the integer programming model of [Gus04].

5.1.5 Quantum speedup of classical techniques

In this section, we present results from [LMS06] where they consider problem instances generated using Hudson's program. The DPLL-based solver *minisat* is used to solve the SAT instance. An interested reader can also generate instances using the HapMap project or [SATb]. The approach is to generate the SAT instance using section 5.1.4 for a given set of genotypes and present it to the SAT solver to solve the instance to obtain the satisfying assignment (i.e. the minimum set of haplotypes which satisfy the genotypes). In [LMS06], the proposed SHIP (Sat-based Haplotype Inference by Pure Parsimony) is compared to the best known solvers and it was shown that the SHIP solvers were the fastest compared to all other techniques as well as solved almost all instances within 1000 seconds.

	Sites	Genotypes	Hybrid [BH06]	Harper[WX03]	SHIPs[LMS06]
Benchmarks					
Uniform	30	50	3/15	15/15	15/15
Uniform	75	30	2/10	8/10	10/10
Non-Uniform	50	30	1/15	12/15	15/15
Non-Uniform	100	30	0/15	4/15	15/15
Hapmap	30:75	7:68	12/24	13/24	23/24
Total	30:75	7:68	18/79	52/79	78/79

Table 5.1: Comparison of the performance of SHIP, Harper, Hybrid solvers for HIPP problem on 79 SAT instances.

The above table clearly shows that SHIPs is the most accurate and robust solver to solve the haplotype inferencing problem. We would have ideally liked to compare our quantum complete SAT solver described in section 3.5.3 with *minisat* which is a DPLL solver, however as mentioned earlier the complexity of quantum DPLL-based algorithms are hard to formulate. Unless a quantum computer exists to test the efficiency of the complete solver, it is not clear how to quantify the quantum improvement to the algorithm in obtaining the solutions for the SAT instances. Intuitively, the proposed quantum complete SAT solver should solve the exact same instances the classical *minisat* solver solves for haplotype inferencing, since we are effectively only speeding up the process of searching in the depth first search.

In [LMS06], apart from the above instances, they also consider instances containing more genotypes and prove that for even the toughest instances of SAT, their satisfiability methodology of solving the haplotype inferencing problem is efficient. However, on the flipside, the SAT solvers ran for over 10,000 seconds. The instances considered in the paper are not the most complex instances, since genotypes can have up to 500 sites or more, for which the SAT instances can be intractable even for the most efficient SAT solvers. One of the challenges mentioned in the paper is to come up with efficient techniques to improve existing SAT solvers for these intractable

instances and to understand the symmetry involved in the haplotype inferencing problem to design better algorithms for this problem. In this thesis, although we are not able to quantify the speedup provided using quantum techniques, we consider a possible answer to their first open question as our quantum version of complete DPLL solver in section 3.5.3 which *should* in theory perform better than the existing DPLL based *minisat* solver.

5.2 Generalized Field development optimization

Optimization methodologies applied to the field of reservoir engineering have been studied for many years. The global demand for petroleum and oil has led to increasing explorations and constructions which have led to greater enthusiasm for research in this subject. Optimization techniques have been developed for several types of oil field development problems such as determining optimal *location* of new wells (well placement optimization), the optimal *operations* of the existing wells (well control optimization), total *number* of wells to be drilled, the *type* of wells to be drilled (injector or producer), the *drilling sequence* of these wells, etc. In this section, we give an introduction to production optimization, well location and control optimization involved in reservoir management. We finally analyze two cases from [IDEC13] involving the well control and well location optimization and present results on how quantum analogues of derivative free techniques can be applied to this problem.

5.2.1 Production optimization

Production optimization involves the maximization of the undiscounted net present value(NPV) or cumulative oil produced from a well on a day-to-day basis or over a period of time by finding the optimal allocation of well connections, values for the variables such as well rates, bottom hole pressure(BHP)[1]. The objective function of the optimization problem $J(u)$ is calculated by computing the cost of oil and water injection and subtracting it from the cost of manufactured oil from the wells. Often with the increased time of simulation of the reservoir, the number of operating variables increases and hence the objective value becomes complex. Added to the number of variables, the dynamics of this reservoir simulation is non-linear with respect to the control variables, and hence the objective function is expensive to evaluate. In reality, the objective function is computed from a black-box (often referred to as *reservoir simulator*), which we discuss in section 5.2.3. The constrained production optimization problem can be framed as the following maximization program:

[1]Bottom-hole pressure (BHP) is the pressure at the bottom of a well.

$$\begin{aligned} &\text{maximize:} \\ &\qquad J(u) = r_o Q_o(u) - c_{wp} Q_{wp}(u) - c_{wi} Q_{wi}(u) \\ &\text{subject to:} \\ &\qquad c_i(u) \leq 0 \quad i = 1, \ldots, m \\ &\qquad L_B \leq Au \leq U_B \end{aligned} \tag{5.6}$$

where r_o is the price of oil (\$/STB [2]), c_{wp} and c_{wi} are the cost of handling produced water and the cost of water injection [3] (\$/STB), and Q_o, Q_{wp} and Q_{wi} are the cumulative oil production, water production and water injection (STB) obtained from the reservoir simulator. In this constrained optimization problem, $J(u)$ is the net present value to be maximized over the control variables u with the non-linear constraints c_i (such as injection and production constraints) subjected to the constraints $L_B \leq Au \leq U_B$. One possibility to solve the production optimization problem, is to compute the derivative of the objective function with respect to the variables using techniques such as numerical finite differences and adjoint-based techniques as discussed in section 4.1.1. Adjoint-based techniques, although more efficient than numerical finite differences, require extraction of information from the reservoir simulator during the course of the computations, and therefore is computable with detailed knowledge of the simulator source code. The computations associated in developing the adjoint code, even with the knowledge of the simulator code is significantly high, and hence derivative free techniques are preferred. Production optimization was considered in [Ise09], [ECID11] where they compare various gradient based and derivative free techniques for the production optimization problem.

5.2.2 Well placement and Well control optimization

In order to maximize the net present value of the production optimization, we are often required to determine the optimal values of parameters such as well rates and BHP's. However since these are continuous varying parameters, the optimization problem is non-linear and derivative free techniques are required to calculate the optimal values for the optimization problem. Optimizing over these continuous varying parameters such as BHP's and well rates is referred to as well control optimization. Well placement optimization refers to determining the optimal location for the wells to be drilled in the field and the type of well that should be drilled at that location. Since the reservoir considered could exhibit rough discontinuous optimization surfaces

[2] Stock tank barrel (STB) is the volume of treated oil stored in stock tanks.

[3] Water injection or water flooding refers to the method in the oil industry where water is injected into the reservoir to increase the pressure and thereby stimulate production.

with multiple local minima, derivative free techniques have the natural advantage of escaping such local minima compared to gradient based techniques and hence are preferred. Often in literature the well placement optimization and well control optimization, have been addressed separately making assumptions about the unconsidered factors, however in [IDEC13] they consider both these problems simultaneously called the *joint well placement and well control* optimization. The motivation for this joint study is that it has been observed lately that the optimal parameters of these wells are dependent on the location and type of the well.

5.2.3 Reservoir simulation

Reservoir simulation is a field devoted to the study of the complex behavior of the natural porous geological formations. This is carried out using computer programs called reservoir simulators to solve equations governing the heat and mass flow in porous media which are characterized by near-elliptic pressure equations and hyperbolic saturation equations. The reservoir simulator, referred to as the black-box, takes the geometry and properties of the reservoir as input to evaluate the function by solving the non-linear partial differential equations in order to evaluate the objective function. The non-linear equations in the simulator are solved by the Newton-Raphson method with each iteration involving the calculation of a Jacobian matrix in the simulator. These Jacobians are linearized and result in a system of linear equations which are sparse in nature, and can be solved relatively easily through the linear solver. Due to the sparsity of the system, the performance of the linear solver is heavily dependent on the robustness of pre-conditioners that are used to speedup the convergence. The commonly used pre-conditioners are, diagonal scaling, block diagonal scaling, incomplete LU decomposition, algebraic multi-grid, constrained pressure residual, etc. The simulator that is considered in this section is Stanford's general purpose research simulator (GPRS) [Jia07].

5.2.4 Numerical Experiments

In this section we present two cases of field development optimization problem presented in [IDEC13] to illustrate the quantum speedup in derivative free optimization as discussed in section 4.2. The geological models in figure 5.1 correspond to a fluvial deposition system, where the first model contains 40×40 grid blocks and displays two times the variation in permeability. The second model is relatively larger, contains 100×100 grid blocks and displays four times the variation in permeability. In both the models considered below, the reservoir simulation time considered is 3000 days. The simulation time has to be long enough to ensure that the water breaks through at the production well, or the water front must reach the production well to make the optimization problem worth considering. Hence the simulation time i.e. 3000 days implies

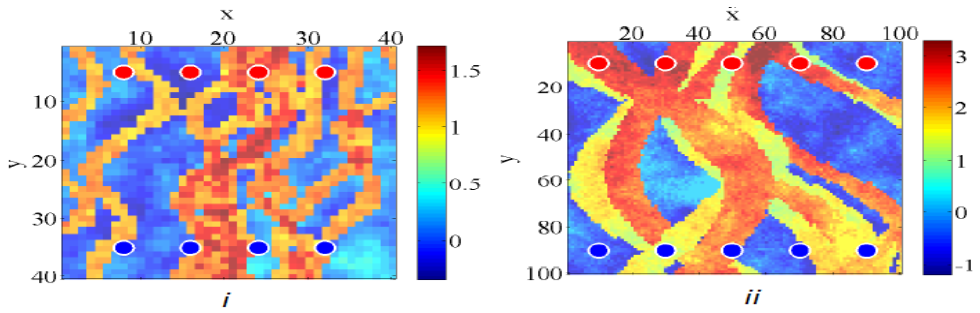


Figure 5.1: [IDEC13] Geological model for Model 1(i) and Model 2(ii) with injection (blue) and production (red) wells.

the fluid flow is simulated for about 9 years, whose data is the input to the reservoir model. Running the model actually takes a few seconds to a few minutes depending on the size of the model (the number of grid blocks) and the complexity of the model (oil-gas-water flow in highly heterogeneous models takes longer to evaluate than oil-water flow in a near homogeneous model).

Parameters	*Reservoir Model 1* (40 × 40)	*Reservoir Model 2* (100 × 100)
Grid Cell Dimension	$50 \times 50 \times 15$ ft^3	$50 \times 50 \times 15$ ft^3
Initial pressure p_i	5080 psi at 8500ft	5080 psi at 8100ft
Rock compressibility c_R	0.5×10^{-5} l/psi	0.5×10^{-5} l/psi
Well drilling cost	$ 4 million per well	$ 5 million per well
Injector BHP	6000-9000 psi	6000-9000 psi
Producer BHP	2500-4500 psi	1500-4500 psi
c_{wp}	$18/STB	$10/STB
c_{wi}	$36/STB	$5/STB
Max. field water injection rate	270 STB/day	1200 STB/day
Min. field oil production rate	200 STB/day	500 STB/day
Max. field liquid production rate	320 STB/day	1200 STB/day
Max. water cut in any production well	0.5	0.8

Figure 5.2: Simulation and Optimization parameters for both reservoir models [IDEC13] .

In the development of simple channelized reservoir (figure 5.1(i)), the optimization problem considered is, given eight wells (represented by the dots) on the geological surface, determine the wells that are meant to be drilled and how to control the wells that are drilled. In this model, the BHP remains constant through the simulation (effectively with one control period). The optimization problem has 16 variables, the first 8 variables are binary variables to decide if a well should be drilled or not and the next 8 variables are to decide the optimal well BHP's. Effectively, the variables in the mixed integer non-linear program involve computing 8 well BHP's for the corresponding configuration (well placement) of the 8 binary variables. In [IDEC13], the

technique adopted to decide the optimal well placement is by a brute force technique. Since there are 2^8 possible combinations of well placement, they evaluated the MINLP for all the combinations before deciding which configuration satisfied all constraints. The ranges of the BHP in the table (6000-9000 psi and 2500-4500 psi for injection and producer BHP's) define the bound constraints for the optimization problem and the max./min. field parameters and water cut parameter define the non-linear optimization constraints in the problem 5.6. The technique of brute-force enumeration is tractable in this problem, but generally not viable when the system gets more complex with over 15 or 20 variables, in which case derivative free techniques are required to find the optimum to the optimization problem. Employing MADS for the same optimization problem instead of brute-force would require at most 32 polling points in the algorithm since there are 16 optimization variables.

The complex heterogeneous channelized reservoir (figure 5.1(ii)) involves the maximum placement of five injectors and five producers. The objective remains the same to optimize the number of wells, their locations and the control parameters. Since there are at most 10 wells, there are 10 binary variables, and 20 well locations variables (since there are two variables (x, y) in the coordinate of each well). In this model, the BHP's are updated every 1000 days and there are 3 control periods, and hence 30 (3×10) control variables. Overall, there are 60 variables in the optimization problem which is relatively high when compared to the first model. Using MADS to solve this optimization variable there will be at most 120 polling points (refer to section 4.1.5.1). The results for this problem have been omitted here, however we have analyzed the explicit implementation of the derivative free MADS algorithm for both the simple and complex models. The following table summarizes the time taken for the MADS algorithm on a single processor for two instances of the small grid (5.1 (i)) and one instance of the large grid 5.1 (ii). [4] In the last row, we speculate the parameters required for optimization of a problem with 125 variables.

	Grid size	No. of Iterations (MADS)	No. of Black Box evaluations /iteration	Time taken/ iteration (2GHz)	**Overall time** (2GHz)
Problem instance					
Small Grid,Problem 1 ($\mathbb{R}^{16}$)	10^3	53	29	29 s	12.5 h
Small Grid,Problem 2 ($\mathbb{R}^{16}$)	10^3	55	27	18 s	7.6 h
Large Grid ($\mathbb{R}^{60}$)	10^4	162	113	490 s	103 days
Huge Grid(Speculated Results) ($\mathbb{R}^{125}$)	10^5	600	250	1 h	18 years

Table 5.2: Summary of Classical time for running MADS on single machine.

[4] The data-set consisting of the number of iterations, number of black-box evaluations, time taken etc. for MADS in [IDEC13] was provided by Obiajulu.

5.2.4.1 Quantum improvement

In each iteration of MADS, in the poll step, the algorithm evaluates the function at *all* poll points (in complete polling) to find if a point exists with a lower functional value. We showed in section 4.2.2, by employing the quantum minimum finding algorithm in this step, the number of queries made to the black-box can be quadratically reduced. In all the cases that we have analyzed here, the exact number of black-box evaluations required per iteration as well as the approximate time taken per function evaluation is known. This allows us to approximately compute the time it would take a quantum computer to perform the same optimization on a single core. The following table also accounts for both cases of a quantum computer having the same clock speed as a classical computer and being 10 times slower.

	Grid size	No. of Iterations (MADS)	No. of evaluations/ iteration	Time taken/ iteration (200 MHz)	**Overall time** (200 MHz)	Time taken/ Iteration (2 GHz)	**Overall time** (2 GHz)
Problem instance							
Small Grid,Problem 1 ($\mathbb{R}^{16}$)	10^3	53	5	290 s	20 h	29 s	2 h
Small Grid,Problem 2 ($\mathbb{R}^{16}$)	10^3	55	5	180 s	13.7 h	18 s	1.37 h
Large Grid ($\mathbb{R}^{60}$)	10^4	162	9	4900 s	83 days	490 s	8.3 h
Huge Grid(Speculated Results) ($\mathbb{R}^{125}$)	10^5	600	13	72000 s	17 years	7200 s	1.7 years

Table 5.3: Summary of quantum time for DFO on a single machine.

5.2.5 Practical flipside

In reality, these functional evaluations in each iteration are generally carried out on clusters where each processor in the cluster is given a particular point in the poll set to evaluate the objective function. The maximum number of processors required for each iteration is at most twice the number of variables in the problem (since there are at most $2n$ polling points for an optimization problem with n variables). Consequently, the overall time taken to carry out the optimization problem is much faster. To give an idea of the numerical difference between serial and multi-core computations, assuming a particular function (black-box) takes 30 seconds to evaluate, *without* access to a cluster, it would have taken 20 hours to find the optimal solution, however using 30 parallel processors it would take less than half an hour. The following table illustrates the separation in the time taken for implementing the algorithm on a classical distributed architecture versus a single processor in 5.2. Since the number of variables in the optimization problems for field development is relatively low, it is possible to allocate the necessary number of processors for each iteration of the algorithm.

	Grid size	No. of Iterations (MADS)	No. of Parallel processors	No. of evaluations/ iteration	Time taken/ iteration (2GHz)	**Overall time** (2GHz)
Problem instance						
Small Grid,Problem 1 ($\mathbb{R}^{16}$)	10^3	53	29	30	29 s	26 m
Small Grid,Problem 2 ($\mathbb{R}^{16}$)	10^3	55	27	27	18 s	17 m
Large Grid ($\mathbb{R}^{60}$)	10^4	162	112	112	490 s	22 h
Huge Grid(Speculated Results) ($\mathbb{R}^{125}$)	10^5	600	150	250	2 h	84 days

Table 5.4: Summary of classical time for DFO on parallel architecture.

Ideally in the quantum setting, we might hope that Grover's search uses $O(\frac{\sqrt{N}}{M})$ queries to find the marked element when there are M machines running in parallel. However, Zalka showed in [Zal99] that Grover's search algorithm requires $\Omega(\sqrt{\frac{N}{M}})$ queries when there are M processors in parallel and this was shown to be optimal. Effectively, Zalka showed that $O(\sqrt{N})$ queries cannot be divided between the M parallel processors but instead the best one can do with having parallel quantum processors is to divide the search space between M processors and have each processor perform quantum searches individually of their subspaces.

In section 4.2.2, when we showed that quadratically fewer number of evaluations is required in each iteration of MADS, we would have ideally expected the same optimal solution to MADS with fewer number of parallel machines. However, the result of Zalka shows that in the quantum setting, a quadratic speedup to the number of function evaluations in the poll step does *not* improve upon the scenario in the classical setting for the problem cases considered.

	Grid size	No. of Iterations (MADS)	No. of Parallel Processors	No. of evaluations/ iteration	Time taken/ Iteration (200MHz)	**Overall time** (200MHz)	**Overall time** (2GHz)
Problem instance							
Small Grid,Problem 1 ($\mathbb{R}^{16}$)	10^3	53	30	5	290 s	4.2 h	25.2 m
Small Grid,Problem 2 ($\mathbb{R}^{16}$)	10^3	55	27	5	180 s	2.75 h	17 m
Large Grid ($\mathbb{R}^{60}$)	10^4	162	112	9	4900 s	9.2 d	22 h
Huge Grid(Speculated Results) ($\mathbb{R}^{125}$)	10^5	600	150	13	72000 s	1.06 y	64.5 d

Table 5.5: Summary of Quantum time for DFO on parallel architecture

The only speedup that can salvaged is if the optimization is performed on a grid where the number of MADS iterations (or the number of variables) is very much higher than the number of affordable processors in the cluster. This scenario however is unlikely in a realistic setting, since one of the most complicated grids the industries currently handle are grids with 10^6 grid cells for which the speculated data was presented in table 5.4. Even though the numbers of parallel processors are fewer than the number of poll points, the advantage of quadratically improving the number of queries is negligible with these values. In fact with these parameters, the time taken in both the classical and quantum setting seems infeasible.

5.3 Inserting Dampers in between adjacent buildings

The progress of technology, growth in population, limited land availability and a growing demand for business and residential complexes, has led to buildings being constructed in close proximity these days. However promising this sounds for the economy and industrial organizations, during an earthquake, the effect of *pounding* between these closely constructed buildings could be destructive. *Pounding* is the impact of adjacent buildings on one another when the separation gap in between them is less than the required amount for them to vibrate individually without affecting the other building. This effect of pounding has been witnessed [Ber87],[CDT12] repeatedly in many catastrophic earthquakes such as the 1985 Mexico earthquake, 1988 Sequenay earthquake in Canada, 1989 Loma Prieta earthquake, 1992 Cairo earthquake, 1994 Kobe earthquake, 2001 Bhuj earthquake and the recent 2011 New Zealand earthquake.

The simplest way to reduce the damage of pounding is to provide enough separation between buildings, however due to the high cost of land and increasing demand, this is not viable and other alternatives need to be considered. Buildings which are older are more probable to get damaged during an earthquake due to older designs. A possible solution is to destroy these old buildings and build newer buildings based on better design codes, however this would not be economical. Another feasible option would be to improve the stability of these older buildings to make them withstand a seismic event, which would be more economical. In 1972 [KCS73], engineers considered a possible solution of using mechanical devices used to join adjacent buildings called *viscous dampers*. Dampers (could be thought of as giant shock absorbers) are effective and economical devices used to connect adjoining structures to absorb the vibrational energy of the buildings during an earthquake. The absorbers are attached to the base of each floor in the building to keep both the buildings at a reasonably fixed distance apart and not pound into each other during an earthquake, thereby reducing the damage enormously. In this section we have analyzed the concept of introducing dampers in between adjacent buildings, the optimization of number of dampers to be connected, the characteristics of the dampers and impact of introducing dampers. In section 5.3.1 we provide motivation for the need of optimization in the problems. In section 5.3.2 we give the mathematical model and simulation to represent the physics of the buildings and the dampers. In section 5.3.3 we introduce the damper location problem, present the well-known classical algorithms from [BHT12] and present quantum techniques which can be employed to improve these algorithms. In section 5.3.4 we consider the damper coefficient optimization problem solved using derivative free techniques implemented in [BHT12] and employ quantum techniques to speed up derivative free optimization for this problem.

5.3.1 Motivation

The major hurdle in installing dampers in between adjacent buildings is the cost factor. We need to install dampers from the second floor up to the highest floor of the smaller building and the cost of each damper [Pal04] is between $ 50,000 and $ 100,000 excluding costs such as installation, labor, material, value of the currency etc., which could overall become exorbitant. Ideally we would like to minimize the cost of installing dampers and get the maximum benefit out of the budget. There is a need to optimize the number of dampers and the coefficients of the dampers to be installed in between adjacent structures without reducing the efficiency of the construction. There have been many experimental studies such as [YXL03], [BJ06], [PJ10] which showed that the number of dampers could be reduced without affecting the safety of the structures, and there are practical implementations of dampers in buildings in California and Japan, which have validated the efficiency of using dampers as a connectives in between structures to mitigate the effect of pounding.

There are several papers in the literature which have considered the mechanical behavior of dampers to improve the seismic stability of coupled buildings, however very few of them have considered optimization tools for the design of such buildings. Many papers which focused on the damper location problem, assume that all the dampers have the same damper coefficients and the papers which focused on optimizing damper coefficients assume the dampers need to be placed on all floors. We present the bi-level optimization presented in [Big12] which considers the *joint damper placement and damper coefficient optimization*. The first step of this bi-level optimization problem is to find the optimal arrangement of the dampers and second step to optimize the mechanical properties of the dampers in between structures. Our main contribution in this thesis has been to implement quantum techniques to improve both steps of the bi-level optimization problem and present results that improve the best known classical algorithms. This quantum improvement is integral since with the average height of structures increasing these days, even the classical heuristics are taking time in the order of days to solve these optimization problems.

5.3.2 Model and Optimization

The methods of modelling the dampers to be placed in between adjacent buildings can be sub-categorized into single degree of freedom (SDOF) and multiple degree of freedom (MDOF). Note that there are other techniques such as the finite element method and experimental method of analyzing the model, but SDOF and MDOF have been considered and studied extensively in the literature. SDOF is the basic representation of these adjacent structures as a lump of mass connected by dampers with no consideration of the number of floors and dampers with

variable coefficients. Due to its simplicity, it can be solved analytically easily but in reality the solutions are inaccurate and far from being implementable. The MDOF is a more accurate model [XHK99] to predict the response of such structures during earthquakes. In this model, each floor is considered as separate lump of mass, connected together by dampers which vary in their coefficients. Although the MDOF model is accurate, this model is complicated and hard to solve analytically.

5.3.2.1 Mathematical model of MDOF

In the previous sections, we gave an overview for the need of optimizing the cost of installing dampers between adjacent structures. However, in structural control optimization, there are many factors which could be considered to optimize such as maximum drift, maximum acceleration, cumulative drift, energy absorbed by the structure, damping ratio, cost of the retrofitting system, risk of damage, etc. The objective function considered in the latest papers ([BHT12], [Big12], [BHNT13]) are the inter-storey drift and the same has been adopted here.

Consider two buildings with n and $n+m$ floors and n_d dampers to be placed as shown in figure 5.3 There are $2n+m$ degrees of freedom, and a vector $x(t) \in \mathbb{R}^{2n+m}$ associated to represent the displacement of the floor at any time instant t. From [BHT12], the governing equation of this mechanical system in order to simulate the actual system is,

$$Mx''(t) + (C + C_d)x'(t) + Kx(t) = MEg(t) \tag{5.7}$$

where M, C, K are $\mathbb{R}^{(2n+m)\times(2n+m)}$ matrices generated by the mass, damping and stiffness coefficient of each of the buildings respectively, E is a vector with all ones, $g(t)$ is the ground acceleration during an earthquake, C_d represents the damping matrix constructed using the n_d damper coefficients and c_d is the vector of damping coefficients for each floor where floors (represented by the second index) without dampers are assigned $c_{d,i} = 0$. The respective matrices have been elaborately outlined in [BHT12] and omitted here. Equation 5.7 in the frequency domain can be written as

$$ME e^{i\omega t}\sqrt{S_g(\omega)} = -\left(-M\omega^2 X(\omega) + (C + C_d)i\omega X(\omega) + KX(\omega)\right)e^{i\omega t} \tag{5.8}$$

where S_g is spectral density function of ground acceleration and the response of the building is

$$X(\omega) = -\frac{\left(ME\sqrt{S_g(\omega)}\right)}{(-M\omega^2 + (C + C_d)i\omega + K)}. \tag{5.9}$$

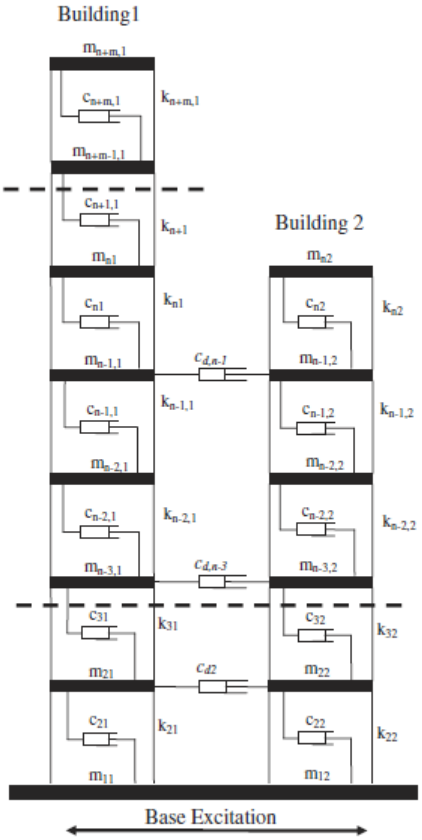


Figure 5.3: Model of adjacent buildings [BHT12] with 5 and 7 floors respectively, connected by 3 dampers.

For a given damping coefficient c_d, the standard deviation of the displacement response for the i^{th} floor of building b can be numerically approximated as

$$\sigma_{ib} = \left(\int_{-\infty}^{\infty} \|x_{ib}(\omega)\|^2 \, d\omega \right)^{1/2} \tag{5.10}$$

where $x_{ib}(\omega)$ is the component of $X(\omega)$ corresponding to the i^{th} floor of building b. Using σ_{ib} the inter-storey drift can be computed as,

$$f_{ib} = (\sigma_{ib} - \sigma_{(i-1)b})^2 \tag{5.11}$$

and adopting the above formula for all floors, the maximum inter-storey drift between the two structures can be computed as,

$$F = \max\{\max\{(\sigma_{i1} - \sigma_{(i-1)1})^2 : i = 1, \ldots, n+m\}, \max\{(\sigma_{i2} - \sigma_{(i-1)2})^2 : i = 1, \ldots, n\}\}. \tag{5.12}$$

The first term in the above expression is the contribution of the inter-storey drift in the first structure and the second term in the second structure. Ideally n_d dampers need to be placed in between the structures to minimize the maximum inter-storey drift F between both structures as computed from equation 5.12. The numbers of variables involved in this optimization problem are $2n + m - 2$ (dampers do not need to be placed on the ground floor since they are already connected by the base). In short, the optimization problem addressed is to place n_d dampers between two structures with n and $n + m$ floors by solving

$$\begin{aligned} &\min_{c_d \in \mathbb{R}^n} \max\{f_{ib}(c_d) : i = 1, \ldots, n+m, b = 1; i = 1, \ldots, n, b = 2\} \\ &\text{subject to : } c_{d,j_1} = 0, c_{d,j_2} = 0, \ldots, c_{d,j_{n-n_d}} = 0 \end{aligned} \tag{5.13}$$

Effectively, subjecting the damper coefficients of the floors without dampers as zero, we are minimizing the maximum inter-storey drift between both the buildings by varying the damper coefficients. In order to compute the objective function or $f_{ib}(c_d)$, knowledge of the integral σ_{ib} is required and solving this integral is extremely difficult. If derivatives of the objective function with respect to all variables were present, the optimization problem could have been solved, but due to the complexity of the objective function that is not an option. Hence, to evaluate the objective function, simulation is used to obtain a numerical approximation. A sequence of earthquake models are simulated which is an input for the simulation software. Based on different damper coefficients, the response of the buildings is then evaluated by the software. Initially, engineers used genetic algorithms to optimize this problem before algorithms such as MADS (Mesh Adaptive direct search), RAGS (robust approximate gradient sampling), proved to solve this problem better. There has been a study analyzing various gradient-free techniques in [BHNT13], concluding that RAGS achieves the best tradeoff between solution time and quality.

The technique employed to solve the bi-level optimization problem is to break the optimization problem into an outer discrete combinatorial problem of determining the j for which $c_{d,j} = 0$ and an inner continuous optimization problem of determining the specific values of $c_{d,j} \neq 0$. Effectively, the outer loop finds the optimal configuration of the dampers to be installed (or floors where dampers are not present and $c_{d,j} = 0$) and the inner loop finds the optimal set of damping coefficients (floors with non-zero coefficients $c_{d,j}$) for this arrangement of dampers. Both these problems have been separately addressed in the following sections.

5.3.3 Damper Location Optimization

The discrete optimization problem of deciding where to place the dampers between adjacent buildings has been specifically considered in [BHT12]. Disregarding the cost of dampers, it is intuitive to believe that placing the dampers on all floors would provide maximum safety, however

it was shown this is necessarily not the case and in fact placing lesser number of dampers can cause the structures to pound with a lower probability. In this section we shall go over the techniques mentioned in their paper and analyze the most effective algorithms. The assumption made in this work is that the damping coefficients of all dampers are equal, which will however be generalized in section 5.3.4.

5.3.3.1 Classical Algorithms

1. **Exhaustive Search**: The technique of exhaustive search is the most naive technique to find optimum floors at which the dampers need to be placed.

Input: 2 unconnected structures with n and $n+m$ floors, n_d dampers
Output: Arrangement of n_d dampers between the 2 structures.

(a) **For** k=1:$\binom{n}{n_d}$
(b) Choose n_d floors from the smaller structure to place dampers.
(c) $\min_k$F(x)
(d) **End** Loop

Figure 5.4: Exhaustive enumeration algorithm.

where $F(x)$ refers to the inter-storey drift from 5.12. The total number of simulations are

$$\binom{n}{n_d} = \frac{n!}{(n-n_d)!n_d!}. \tag{5.14}$$

This brute force search naturally results in the best possible configuration but the drawback is the intractability as the number of floors and dampers increase. For example if there were two buildings with 50 and 40 floors, with 10 dampers, it would take approximately 8.5×10^8 simulations to be solved. Assuming each simulation requires 200 black-box queries with each black-box evaluation requiring 5 seconds, the total time would be 27 centuries.

2. **Inserting Dampers**: Instead of considering all possible combinations of n_d dampers in the exhaustive search algorithm, in this heuristic technique, the dampers are sequentially inserted in between the structures. The two buildings are initially considered as unconnected structures and the algorithm starts by sequentially inserting dampers. The algorithm initially finds the best location to place the first damper by evaluating the objective function for each floor to find the best allocation. It then proceeds to find the optimal location of the second damper and this is carried on till n_d dampers have been placed in between the two structures.

Input: 2 unconnected structures with n and $n+m$ floors, n_d dampers
Output: Arrangement of n_d dampers between the 2 structures.

(a) **For** t=1:n_d
(b) **For** k=1:n
(c) T $\leftarrow \min_k$ F(x)
(d) **End** Loop
(e) Fix the damper location T.
(f) Goto step (b), reducing n to $n-1$.
(g) **End** Loop

Figure 5.5: Inserting dampers algorithm.

The total number of required simulations is

$$\begin{aligned} N &= n + (n-1) + \ldots + (n-(n_d-1)) \\ &= nn_d + n_d - \left(\frac{n_d^2 + n_d}{2}\right). \end{aligned} \tag{5.15}$$

In this heuristic unlike exhaustive enumeration, the number of simulations does not grow exponentially with the number of floors.

3. **Inserting Floors**: In contrast to the previous method of inserting n_d dampers sequentially at appropriate floors, in this heuristic technique a random arrangement of dampers for both the structures is assumed and sequentially adjacent floors are inserted to both the structures till the maximum number of floors for the smaller structure is reached.

Input: 2 structures with n_d and n_d+m floors, n_d dampers
Output: Arrangement of n_d dampers between the 2 structures with n and $n+m$ floors.

(a) **For** t=1:$n-n_d$
(b) **For** t=1:n_d+1
(c) $T \leftarrow \min_k$ F(x) (compute best location to insert adjacent floors)
(d) **End** Loop
(e) Insert pair of floors at T, resulting in 2 buildings with n_d+1 and n_d+1+m floors.
(f) **End** Loop

Figure 5.6: Inserting Floors algorithm.

The total number of simulations in the above algorithm is $(n_d+1)(n-n_d)$, since for each of the $(n-n_d)$ floors to be placed, n_d+1 simulations have to be carried out to find the optimum

location to insert the floors without dampers. Note that the input to this algorithm are 2 structures with n_d and $n_d + m$ floors instead of the conventional n, $n + m$ floors. In this technique, since it is known that n_d floors have to be connected between both the structures, initially when inserting a pair of adjacent floors the simulation is carried out on smaller structures and towards the end of the algorithm the simulations are carried out on full-scale structures.

4. **Maximum Velocity**: This naive technique relies on the mechanical fact that the force generated and energy dissipated by a linear viscous damper are linear functions of the relative velocity across the damper. Since it would be ideal if the stress caused in the buildings during earthquakes is dissipated through the dampers, the dampers should be placed in adjacent floors with the highest relative velocity. Hence with one simulation on the unconnected buildings, floors with highest relative velocities can be found to place the n_d dampers. Compared to all prior techniques, this method requires only *one* simulation but the drawback is the optimal arrangement is completely independent of the objective function to be minimized in equation 5.12.

5. **Genetic Algorithm**: These are heuristic approaches to solving optimization problems where initially a large number of possible assignments are tried to see which ones seem better and then manipulate those that are not working by trying other combinations. This process is repeated until a satisfactory optimum to the objective function is achieved. The following genetic algorithm is often employed to solve the damper location problem

Input: 2 structures with n and $n + m$ floors, n_d dampers
Output: Arrangement of n_d dampers between the 2 structures.

(a) **Till** stopping criterion is satisfied
(b) Select random combinations of dampers (as initial population).
(c) Evaluate the objective function at all points of the current population.
(d) *Selection :* Carry forward the best few evaluations to the next generation.
(e) *Crossover :* Perform crossover on the points from the previous generations.
(f) *Mutation :* A mutation step randomly alters some points in successive generations.
(g) Evaluate the objective function at all points of the new generation.
(h) **End** Loop

Figure 5.7: Genetic algorithm.

5.3.3.2 Summary

It was shown in [BHT12] that exhaustive searching described in figure 5.4 is theoretically the most robust method but with the increase in the number of levels of the structures, the problem becomes intractable. Intuitively, with a single simulation the maximum velocity method in figure 4 is the fastest method to optimize the damper location but it was shown that this method did not lead to an optimal damper placement. By simulation, it was shown that the inserting floors technique in figure 5.6 method is faster but less accurate than the inserting dampers method in figure 5.5. Although the genetic algorithm in figure 5.7 behaves similar to the inserting floor method, since it isn't reliable and there are no guarantees on the solution obtained, it is generally not preferred. In order to compare or benchmark the robustness and applicability of each of these classical methods, the respective *performance profiles* [DM02] were computed. To plot the performance profile, the performances of all algorithms are compared with the performance of the best algorithm on particular problem instances, varying the parameters of the respective algorithms. Consequently, it was shown that the *inserting dampers* method was the most practical method in terms of speed and robustness to find the best configuration to place the dampers.

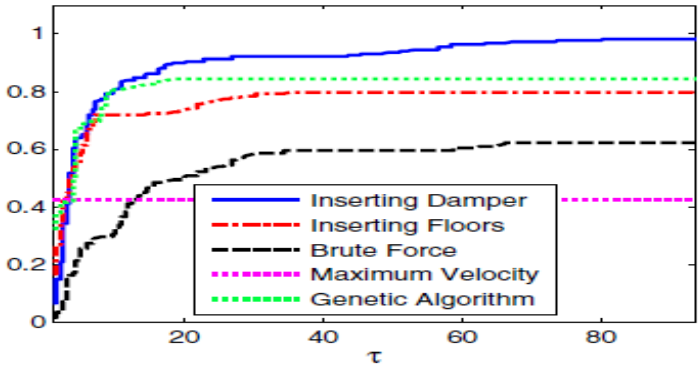


Figure 5.8: Model of adjacent buildings connected with dampers [BHT12].

5.3.3.3 Quantum Techniques

In this section, we are primarily concerned with the heuristic *inserting dampers* technique which was proven to be the fastest as well as effective for the damper location problem and the *exhaustive search* algorithm which intuitively is the most effective deterministic technique to generate the ideal damper combination. The choice of the first algorithm is primarily because it was demonstrated to achieve the best trade-off between solution time and quality and the reason for consideration of *exhaustive search* is to demonstrate that problems which would have taken

centuries to solve by classical algorithms could be done in less than few days using quantum techniques. This is essential for multi-billion dollar projects of skyscrapers in which accuracy of implementation is absolutely necessary and exhaustive search is the only technique that provides it. The essential theorem that is required for the quantum speedups is the *quantum minimum finding* algorithm shown earlier in section 2.2.1

1. In *exhaustive searching*, instead of querying the objective function for each of the n_d damper combinations, a superposition of all possible damper configurations can be created. Although, this step could take exponential time by preparing each configuration, it is also possible to do the following. The classical Fischer-Yates($\mathcal{FY}_k$)[5] shuffle algorithm takes some seed s and generates a string of length n of Hamming weight k by computing $\mathcal{FY}_k$(s) in $O(n)$ time. We can therefore prepare a superposition and employ the classical algorithm,

$$\sum_s |s\rangle |0\rangle \rightarrow \sum_s |s\rangle |\mathcal{FY}_k(s)\rangle \rightarrow \sum_s |0\rangle |\mathcal{FY}_k(s)\rangle \tag{5.16}$$

   to obtain a superposition of strings with Hamming weight k. This algorithm takes $\widetilde{O}(n)$ (up to poly-log factors) time to compute the symmetric state. Another technique is to employ the efficient quantum algorithm presented in [KM04], to prepare an arbitrary symmetric state with a predefined Hamming weight. They provide a near-optimal circuit that performs the same. Given this symmetric superposition state, using theorem 1 the minimum functional value can be obtained with high probability in $\sqrt{\binom{n}{n_d}}$ queries.

- Create a superposition

$$\frac{1}{\sqrt{\binom{n}{n_d}}} \sum_{\substack{x\in\{0,1\}^n \\ |x|=n_d}} |x\rangle$$

- One query of the objective function would result in

$$\frac{1}{\sqrt{\binom{n}{n_d}}} \sum_{\substack{x\in\{0,1\}^n \\ |x|=n_d}} |x\rangle |f(x)\rangle$$

  Using *quantum minimum finding* algorithm query the objective function $O(\sqrt{n})$ times.
- Measure the first register to obtain the string which minimizes the objective function.

Figure 5.9: Quantum speedup in exhaustive searching.

[5] This is also referred to as Knuth Shuffle, refer to Knuth Volume 4, Section 7.2.1.3.

This is also optimal up to constant factors, as any quantum algorithm needs at least $\Omega(\sqrt{N})$ evaluations of $f(x)$ [BBBV97]. This speedup quadratically improves the number of queries made to the objective function and not the simulation involved in evaluating the function.

Considering the example analyzed earlier with two structures of 40 storeys and 50 storeys connected by 20 dampers, classically it would take 27 centuries to find the minimum of the objective function and assuming same clock speed for quantum gates it would take 18 days on a quantum computer. Although, both computations seem time consuming and infeasible, it can be clearly seen there is a considerable improvement using quantum searching instead of classical exhaustive enumeration. For smaller buildings with fewer dampers, the technique of quantum searching could probably produce results faster. The following plots compare the quantum and classical queries made to the black-box where the number of floors of the smaller building is predefined, and the number of dampers to be installed is a variable.

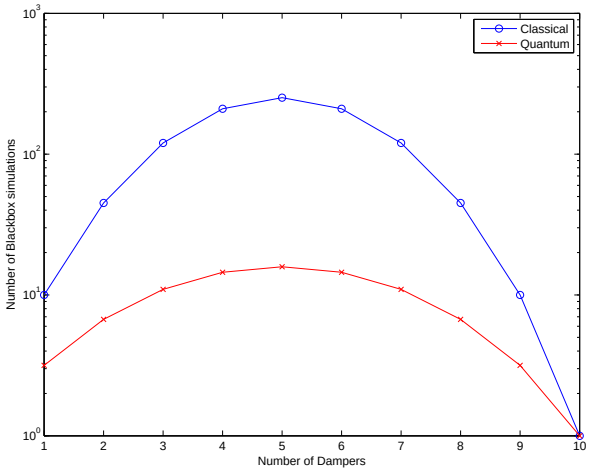


Figure 5.10: Quantum algorithm versus classical algorithm for brute force algorithm 5.4 for structures with 10 floors.

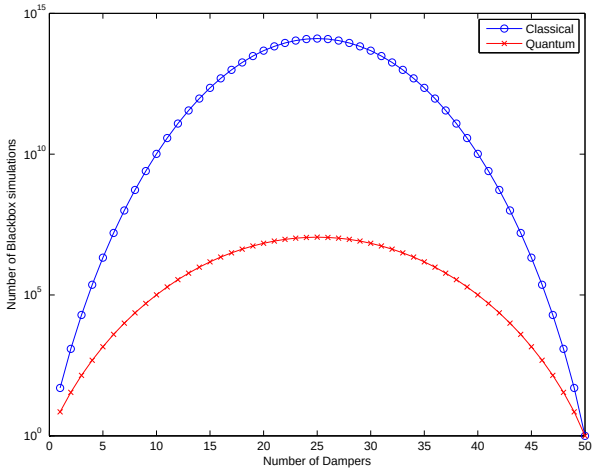


Figure 5.11: Quantum algorithm versus classical algorithm for brute force algorithm 5.4 for structures with 50 floors.

2. In the *inserting dampers* algorithm, to place the first damper in algorithm 5.5, in subroutine (c) instead of running n simulation to find the configuration which minimizes the objective function, using quantum minimum finding 1, the number of simulations required could be quadratically improved. This speedup also follows for every subsequent damper placement. The preparation of the initial symmetric state with pre-defined Hamming weight for each iteration, can be done through the similar techniques mentioned in [KM04]. Hence, the reduction in the number of queries in each iteration improves the running time of the overall algorithm,

Input: 2 unconnected structures with n and $n+m$ floors, n_d dampers
Output: Arrangement of n_d dampers between the 2 structures.

(a) Find the optimal location for the first damper.

- Prepare the superposition

$$\frac{1}{\sqrt{n}} \sum_{\substack{x\in\{0,1\}^n \\ |x|=1}} |x\rangle$$

- Query the black-box $O(\sqrt{n})$ times to obtain $\sum_{|x|=1} |x\rangle\,|f(x)\rangle$
- Measure the first register to obtain the optimal location (t_1^{th} storey) of 1^{st} damper

(b) **For** k=2:n_d

(c) Fix the position of $(k-1)^{th}$ damper

- Prepare the superposition

$$\frac{1}{\sqrt{n-k+1}} \sum_{\substack{x\in\{0,1\}^n, \\ x_{t_1}=\ldots=x_{t_{k-1}}=1, \\ |x|=k}} |x\rangle$$

- Query the black-box $O(\sqrt{n})$ times to obtain $\sum_{|x|=k} |x\rangle\,|f(x)\rangle$
- Measure the first register to obtain the optimal location (t_k^{th} storey) of k^{th} damper

(d) **End** Loop

Figure 5.12: Quantum speedup in Inserting dampers algorithm.

The overall cost of the quantum analogue of inserting dampers algorithm can be calculated up to a small constant factor,[6]

$$\begin{aligned} N &= \sqrt{n} + \sqrt{n-1} + \ldots + \sqrt{n-(n_d-1)} \\ &= i\left(\zeta(-\frac{1}{2}, -n) - \zeta(-\frac{1}{2}, n_d - n)\right). \end{aligned} \tag{5.17}$$

Where $\zeta(s,a)$ is the Hurwitz zeta function defined as $\zeta(s,a) = \sum_{k=0}^{\infty} \frac{1}{(k+a)^s}$. In the following plots, the number of queries made to the black-box is analyzed in the quantum and classical setting with a predefined number of floors, allowing the number of dampers to be installed as a variable.

[6] The constant $\pi/4$ overhead is not considered for computations.

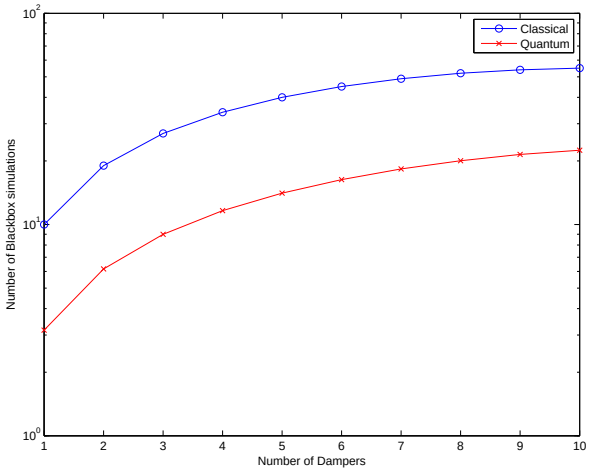


Figure 5.13: Quantum algorithm versus classical algorithm for inserting dampers algorithm 5.5 for structures with 10 floors.

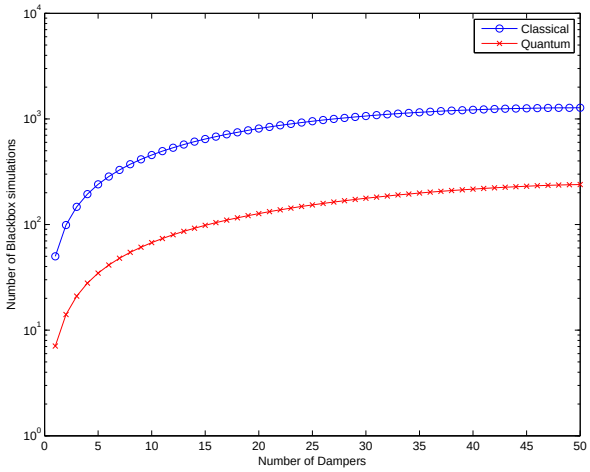


Figure 5.14: Quantum algorithm versus classical algorithm for inserting dampers algorithm 5.5 for structures with 50 floors.

5.3.3.4 Summary

Classically it was shown that the *insertion dampers* algorithm achieves the best trade-off between solution quality and running time. For over 90% of the problems, it resulted in the optimal configurations. In order to get the optimal arrangement with certainty, *exhaustive search* algorithm could be employed but the problem gets intractable with an increasing number of dampers and structure height. In this section, we showed that using quantum search the best known *insertion dampers* technique can be sped up, and can quadratically improve the *exhaustive search* algorithm as well. Effectively even the subroutine in *inserting floors* algorithm can be improved to obtain an algorithm which computes the optimal configuration in $O((n - n_d)\sqrt{n_d})$ black-box evaluations. Both classically and in quantum setting the *maximum velocity* algorithm would require 1 query to be made to the black-box and hence there no straightforward improvement is possible. Figure 5.15 compares both exhaustive enumeration as well as the *inserting dampers* algorithm in the classical as well as quantum setting.

5.3.4 Damper Coefficient Optimization

In the previous section, we made the assumption that all the dampers have equal coefficients to find the optimum placement of the dampers. In this section, this assumption will be generalized to compute the optimal coefficients of the respective dampers for each arrangement. There have been studies that have examined the effect of the non-uniform distribution of dampers

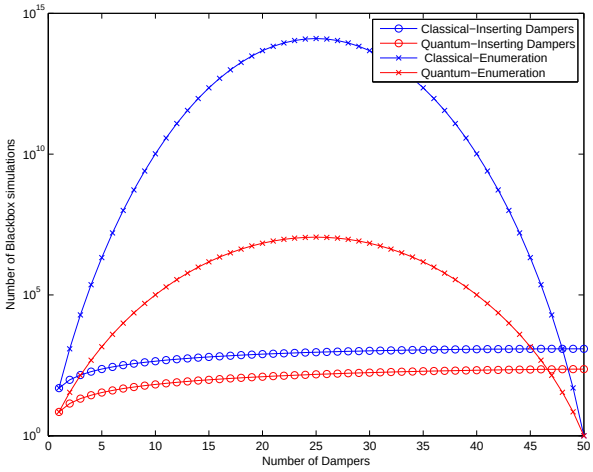


Figure 5.15: Quantum speedup compared to classical algorithms for exhaustive search algorithm 5.4, Inserting Dampers algorithm 5.5 for structures with maximum 50 floors.

in [YXL03],[BJ06], but none of them contain a comparison of the quality of their proposed solution. Disregarding the coefficients of damping device, in [YXL03],[ZGH11] they argue that all adjacent floors of adjacent buildings should be connected by identical dampers (with same damping coefficients $c_{d,1} = c_{d,2} = \ldots = c_{d,n}$). This reduces the optimization problem to a single dimensional problem which can be easily solved using divide and conquer techniques as performed in these papers. However, this claim was later refuted in [BHT12], by showing that a uniform distribution of dampers results in a sub-optimal arrangement. Hence, it is integral to consider dampers with varying damper coefficients to minimize the inter-storey drift to represent the structures realistically.

In equation 5.12, it was observed that the inter-storey displacement is highly dependent on parameters such as damping and stiffness matrices, C_d, K_d, whose entries are in turn dependent on damping and stiffness coefficients $c_{d,i}, k_{d,i}$ as discussed in section 5.3.2.1. Hence the optimal set of damping and stiffness coefficients is indirectly dependent on the dampers used and the placement of the dampers in between the structures. The placements of the dampers have been analyzed in the previous section and it remains to optimize for optimal damping and stiffness coefficients for each placement. Allowing the dampers to have varying damping and stiffness coefficients, the optimization problem results in an n-dimensional optimization problem, for which the objective function is evaluated by simulation. As mentioned earlier, with each configuration of damper placement and damper coefficients, the simulator evaluates how the buildings respond. With the derivative information being difficult to obtain, gradient free techniques are employed to compute the optimal damper coefficients in the second level of the bi-level optimization problem.

Three algorithms were analyzed in [BHNT13] to solve the optimization problem, genetic algorithms, mesh adaptive direct search (MADS) and robust approximate gradient sampling (RAGS). Genetic algorithms and MADS have been discussed in length in the previous chapter. RAGS is a derivative free technique which is specially designed to exploit the smooth structure of min-max optimization problems, which is the optimization problem that is being considered in this optimization problem. The robust stopping criterion of RAGS reduces the number of function evaluations for the algorithm to converge to the optimum. A detailed description of the algorithm is omitted here. An interested reader is referred to [HN13], [Nut12] .

5.3.4.1 Quantum techniques applied to the classical algorithms

In the MADS algorithm implemented in [BHNT13] the search step is avoided and only the poll step is carried out in each iteration. As explained in section 4.1.6, assuming an initial step size for the algorithm the function is evaluated on the poll set (consisting of points defined in the direction of the positive basis). If the function is evaluated at a point which is less than the present iterate the algorithm moves to this new point, creates a poll set at the new point, and the process carries on till the stopping criterion is attained. In the scenario where none of the points on the poll set are less than the present iterate, the current point is the local optimizer and it is checked if the tolerance is achieved, if not the step size is reduced and the poll points are evaluated again. As discussed earlier, the convergence of this algorithm was proven in [AA06] primarily relying on the theory of positive bases. It was shown in section 4.2 that the classical searching in both the search step as well as the poll step in derivative free algorithms could be quadratically improved. In this section, we articulate the speedup provided to the running time of MADS using quantum techniques and compare it with RAGS.

5.3.4.2 Test Problems and results

We consider 2 cases with differing building heights and similar mechanical properties from [BHT12] and analyze the running time as shown in their paper.

Damping coefficients								
	Building 1				Building 2			
Case	f_1	m_1 (kg)	k_1 (N/m)	c_1 (Ns/m)	f_2	m_2 (kg)	k_2 (N/m)	c_2 (Ns/m)
1	10	1.29×10^6	4.00×10^9	1.00×10^5	20	1.29×10^6	2.00×10^9	1.00×10^5
2	40	1.29×10^6	4.00×10^9	1.00×10^5	10	1.29×10^6	2.00×10^9	1.00×10^5

Figure 5.16: Mechanical properties for Case 1 (f_1=10, f_2=20) and Case 2 (f_1=40, f_1=10).

For both sets of data in [BHT12], the mechanical properties of all floors in each building are assumed to be the same, and the ground parameter accelerations are considered as $S_0 = 4.65 \times 10^{-4} m^2/rads^3$, $\omega_g = 15 rad/s$, $\zeta_g = 0.6$, $\omega_k = 1.5 rad/s$, $\zeta_k = 0.6$.

Table 5.6 analyzes the number of function calls and the objective value obtained for the two test cases in figure 5.16 for the placement of 10 dampers. This table takes into consideration both the queries in the outer loop as well as the inner loop of the optimization problem. For all the cases, the *insertion dampers* technique has been employed for the outer loop optimization. For example, given n_d dampers to be placed between two n storey buildings, the *insertion dampers* algorithm is used to determine an optimal configuration of dampers using $nn_d + n_d - \left(\frac{n_d^2+n_d}{2}\right)$ simulations. In each iteration while the algorithm decides the location to place the dampers sequentially, it also determines the optimal damper coefficients for the dampers through MADS algorithm. Hence the data in the Q-MADS column accounts for the quantum speedup in the discrete optimization step as discussed in section 5.3.3 as well as the quantum speedup in the poll step in the MADS algorithm. The data presented for GA, MADS and RAGS was obtained from [BHNT13] where these algorithms were rigorously analyzed.

		Number of function calls				Objective value		
Case	n_d	GA	MADS	RAGS	*Q-MADS*	GA	MADS/Q-MADS	RAGS
1	1	380	279	170	*11*	4.36×10^{-6}	4.33×10^{-6}	4.33×10^{-6}
	2	1100	873	571	*34*	1.79×10^{-6}	1.72×10^{-6}	1.72×10^{-6}
	3	2000	1883	1661	*74*	1.90×10^{-6}	1.72×10^{-6}	1.72×10^{-6}
	4	2980	3680	2965	*146*	1.87×10^{-6}	1.72×10^{-6}	1.71×10^{-6}
	5	4055	5995	4360	*400*	1.76×10^{-6}	1.72×10^{-6}	1.73×10^{-6}
	6	5105	8999	6376	*247*	1.77×10^{-6}	1.72×10^{-6}	1.73×10^{-6}
	7	6155	13127	9192	*573*	1.84×10^{-6}	1.71×10^{-6}	1.73×10^{-6}
	8	6995	16589	11644	*747*	1.88×10^{-6}	1.71×10^{-6}	1.74×10^{-6}
	9	7625	18710	13511	*868*	1.89×10^{-6}	1.73×10^{-6}	1.74×10^{-6}
	10	350	1511	1208	*73*	1.96×10^{-6}	1.71×10^{-6}	1.77×10^{-6}
2	1	385	312	153	*10*	5.96×10^{-6}	5.94×10^{-6}	5.94×10^{-6}
	2	1045	1010	756	*29*	5.86×10^{-6}	5.86×10^{-6}	5.87×10^{-6}
	3	1930	2097	1766	*63*	5.94×10^{-6}	5.86×10^{-6}	5.87×10^{-6}
	4	2930	4345	3424	*133*	5.89×10^{-6}	5.86×10^{-6}	5.87×10^{-6}
	5	3980	7131	5166	*224*	6.17×10^{-6}	5.86×10^{-6}	5.88×10^{-6}
	6	5030	11821	7045	*382*	5.96×10^{-6}	5.86×10^{-6}	5.89×10^{-6}
	7	6010	16077	8844	*536*	6.22×10^{-6}	5.86×10^{-6}	5.89×10^{-6}
	8	6850	21024	10533	*723*	5.97×10^{-6}	5.86×10^{-6}	5.91×10^{-6}
	9	7480	25831	11646	*915*	6.14×10^{-6}	5.86×10^{-6}	5.92×10^{-6}
	10	350	3545	820	*130*	6.40×10^{-6}	5.86×10^{-6}	5.92×10^{-6}

Table 5.6: Comparison of the number of black-box evaluations for GA, MADS, RAGS from [BHT12], the proposed quantum improved MADS algorithm and the objective value obtained in each algorithm for Case 1, Case 2 in figure 5.16.

To get an idea of the speedup Q-MADS offers over other techniques of optimization, the following graphs pictorially represents the quantities in the above table for both cases in figure 5.16

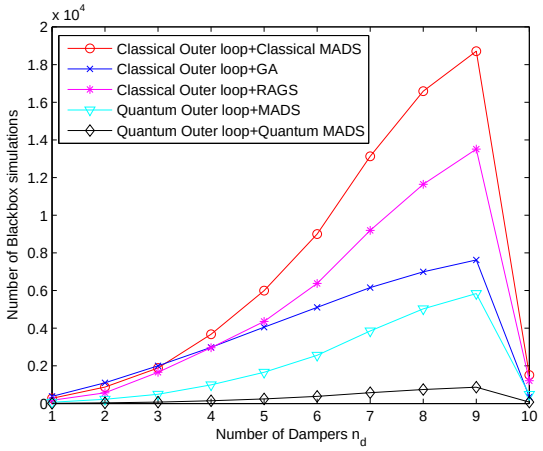


Figure 5.17: Graph comparing 5 optimization techniques for Case 1 in table 5.16.

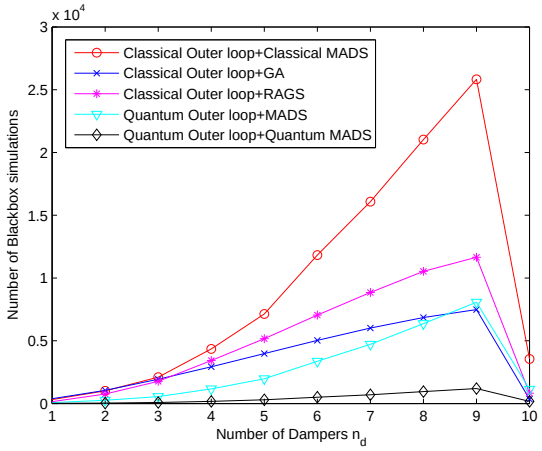


Figure 5.18: Graph comparing 5 optimization techniques for Case 2 in table 5.16.

It follows from the figure that, by using quantum techniques to simply improve the discrete optimization step, the overall algorithm performs better than all the remaining 3 techniques for Case 2, but is slower than genetic algorithms in Case 1. By employing quantum techniques to improve the entire algorithm (i.e. the outer loop as well as the inner loop) the overall number of black-box simulations required to perform the optimization is considerably less than all the other techniques which are currently used in industries. It should be noted that none of these quantum improvements, affect the performance profiles of *inserting dampers* algorithm or the convergence analysis of MADS. Effectively, without changing the values of the iterates after any iteration in the algorithm, we have improved the number of queries that need to be made to the black-box to find the optimum.

5.3.4.3 Conclusion

In this section we have successfully shown that using quantum techniques, we have improved both levels of the optimization procedure used to design damper connected structures. Classical algorithms for the outer discrete optimization of finding the optimal damper configuration was addressed in [BHT12] where they compared 5 techniques to solve the problem and showed that *insertion dampers* method had the best performance profile (trade-off between solution time and

robustness). We have shown that using quantum search we could significantly speedup this algorithm. A comparison of derivative free techniques for the inner continuous optimization for finding optimal damper coefficients were presented in [BHNT13], where the MADS algorithm was shown to produce the most robust results (due to its ability to avoid being trapped in local minima), while the RAGS algorithm was shown to produce solutions which were better in significantly less time than MADS. We have successfully shown that integrating the robust nature of classical MADS algorithm and the quantum speedup, we get a new quantum algorithm Q-MADS which proves to be more robust than RAGS with a faster running time.

5.4 Clock speed of Quantum computers

Calculating the exact time taken by quantum algorithms without a quantum computer is a hard problem. In this thesis so far, we analyzed the query costs of derivative free optimization for a couple of practical problems and algorithmic parameters for SAT solvers to solve practical SAT instances. Although, we do not know how the clock speed of quantum architecture is going to compare with classical hardware, in this section, we comment on the ideal ratio of the clock speeds in order to gain a significant advantage in the quantum paradigm. Majority of the speedups considered in this thesis have primarily arisen by applying Grover's search algorithm. Theoretically, considering we are required to query a black-box $O(N)$ times to ensure the algorithm succeeds with constant success probability, Grover's algorithm states that the same problem can be solved by making $O(\sqrt{N})$ queries. Assuming a slowdown of α (the ratio of the classical versus quantum clock speed) in quantum gates, the overall tradeoff is between N and $\alpha\sqrt{N}$. It is intuitive to see that, a naive quantum speedup can be achieved if and only if $\alpha^2 \leq N$, which is conceivable if we were required to make exponential queries with a constant factor slowdown. However, we highlight few hurdles in the optimization of algorithms and why this speedup isn't straightforward to the practical problems considered in the sections 3.5.1, 5.2, 5.3.

Decoherence is the decay of the state of a system when a quantum state interacts with the environment due to its fragile nature. To isolate these systems from environmental effects and make them useful for quantum computations, a long coherence time is required. Technologies relying on electrons to maintain quantum states have short coherence times and technologies which utilize nuclear effects have been shown to be comparatively more stable. However, the flipside in *good* isolation of states for quantum computation is the slow operations of two-qubit gates. It has been noticed in [LMY+05], [MI06] that gate speeds and decoherence times vary over eight orders of magnitude or more compared to classical gates.

In section 3.5.1, while analyzing satisfiability we considered parameters of Walksat which are currently being used in SAT competitions and applied amplitude amplification to these algo-

rithms. We however realized that the number of iterations which was supposed to be exponential is often not the case in practice where industries consider constant number of iterations with exponential random walk steps. Hence with the maximum number of iterations in Walksat being 100, with a slowdown of 10 times or $\alpha \geq 10$, we lose the quantum advantage. However, as highlighted earlier, it would be interesting to consider if classically the parameters can be rebalanced by considering fewer steps in the random walk and a higher number of iterations. With rebalanced parameters, the advantage using quantum amplitude amplification would be significantly higher even with a slower quantum computer.

In derivative free optimization, for the generalized field development optimization, evaluating the black-box is very costly (each evaluation could take about few hours for grids of size 10^5) and thus the number of iterations of the algorithm is relatively less. From table 5.3, it can be seen that with fewer number of iterations, even on a single processor the advantage of applying quantum techniques with a slowdown of $\alpha \geq 10$ is insignificant. Since the number of queries made to the black-box is less, it is possible to dedicate processors to evaluate the function at each search point, which makes quantum parallel searching insignificant. In section 5.3, we discuss another practical problem of inserting dampers between adjacent buildings employing derivative free optimization. We analyzed the overall number of black-box queries made in the combined outer discrete and inner continuous optimization problem. In this problem, the cost of evaluating the black-box is relatively low (approximately 1 minute per evaluation) and hence the number of iterations of the algorithm is comparatively higher than the field development optimization. From table 5.6, it can be noted that the number of black-box evaluations in Q-MADS in all cases is at most 15 times lower than classical MADS. The speedup one gains in Q-MADS would be insignificant with a slowdown of $\alpha \geq 15$ in quantum architecture.

Although, we have considered only few practical problems to compare classical and quantum architecture, we believe that these problems are extremely relevant among engineering optimization problems. It should be noted that the parameters considered in the above practical problems can be optimized further or rebalanced appropriately (as mentioned in the respective sections) for all cases. It would be interesting to see if either more practical applications of derivative free optimization can be considered or if the number of iterations in random walk algorithms can be increased by reducing the number of random walk steps. The goal is to consider practical problems such that even with a quantum computer being an order of magnitude of 2 ($\alpha \geq 100$) slower than classical architecture, the improvement seen by employing quantum techniques over state-of-the-art classical algorithms is significant.

Chapter 6

Conclusions

In this thesis, we have studied two important problems the Boolean satisfiability problem and derivative free optimization. We have studied and analyzed the quantum speedup applied to both these algorithms. In the Boolean satisfiability problem, we considered techniques such as quantum amplitude amplification and quantization of directed random walks applied to the incomplete Walksat algorithm and proposed a quantum analogue of the complete DPLL algorithm based on the nested Grover's search algorithm. Considering parameters of practical interest, we realized that amplitude amplification does not provide an immediate improvement without rebalancing the parameters of the random walk algorithms. Assuming the parameters considered in this thesis perform optimally, we raise another interesting challenge problem of quantization of directed random walks to improve the random walk algorithms for satisfiability. Finally, considering the DPLL algorithm, we realized that computing the running time of this algorithm is hard, since it involves various heuristics. Classically heuristic improvements are tested on machines to quantify the speedup, however the improvement provided by the quantum analogue of the DPLL solver proposed in this thesis is hard to articulate without access to a quantum computer.

In derivative free optimization, we employed the quantum minimum finding algorithm to improve the search step and poll step in the generalized pattern search and mesh adaptive direct search algorithms. However, we realized that often problems which employ derivative free optimization rely on distributed architecture to solve the problem, in which case we showed that a full quantum speedup was not obtained employing parallel processors. The problems lies in that derivative free algorithms is often employed in practical problems which involve costly function evaluations, with fewer number of black-box queries in order to solve the problem in realistic time. This limits the speedup that could have been obtained by using quantum techniques applied to these algorithms. A natural candidate for problems with faster black-box evaluation time and larger number of queries lies in cryptanalysis, where the black-boxes are fast to implement and

designed so that exhaustive search is essentially the best attack. However, in this thesis, we have tried analyzing problems which might be relevant to other industries apart from cryptography.

We have considered a few practically important industrial problems which involve solving Boolean satisfiability or derivative free optimization, and analyzed the asymptotic improvement to these algorithms with a quantum speedup. For these specific problems, with the knowledge of the parameters of practical interest, we computed the approximate time it would take for a quantum computer to solve these problems with different clock speeds. We analyzed the threshold parameters for these problems below which we obtain a significant speedup, and above which we lose the quantum improvement with the slower quantum architecture. Although, we have considered only a few problems in this thesis, we hope to highlight the need for optimization and the challenges in implementing quantum algorithms. It would be interesting in the future to consider more optimization problems where the quantum speedup could be employed to solve practically important problems to out-perform state-of-the-art classical techniques. We raise several open questions from the perspective of optimizing algorithms which would be worthwhile considering as future work.

6.1 Future work

In this last section, we take the opportunity to highlight a few practical problems which were either not considered in this thesis, or problems for which a quantum speedup wasn't straightforward.

1. **Protein folding**: The protein folding problem can be described as a *global minimization problem*, to obtain the native tertiary structure of a linear polypeptide chain. Computationally, the energy of the native state is the minimum compared to all possible tertiary structures given a polypeptide chain. Much of the current effort these days is directed at predicting this native structure, and predicting how this native structure is formed from the linear chain. Protein mis-folding is the major cause of diseases such as Alzheimer's, Parkinson's, Cancer, Type-2 diabetes, etc. and understanding the process of folding to the native structure is essential to preparing drugs to cure these diseases. Partly, the reason for interest in this problem is because it seems that nature has an efficient algorithm which solves this problem by finding the native structure *in seconds*, whereas the computational complexity for computers to simulate it is considered NP-hard [1]. There exists *only one*

[1] There have been proposed solutions to protein folding by D-Wave [PODDB+12] where they use the technique of simulated annealing to solve the optimization problem. Considered a polypeptide chain with *at most 40 states*, after executing their algorithm, in 10,000 measurements only 13 measurements gave outputs of the native configuration of the protein. With less than 1% of the measurements providing right outputs, there is no accuracy and precision guarantees, which is *absolutely* necessary for a problem which is essential in curing diseases and preparing drugs.

branch-and-bound deterministic algorithm for this problem in the literature and hence any quantum algorithm which improves upon this result would be interesting. Companies which require native structures of proteins often use the genetic algorithm proposed by Unger and Moult [UM93] for tractable linear chains. It would be interesting if the recent result of a fully quantum genetic algorithm in [SRN12] could be employed to improve the best known classical genetic algorithm for protein folding.

2. **Directed Quantum Walks**: A challenge in quantum algorithms remains a well-defined notion of quantum speedup to random walks on directed graphs. We have discussed in section 3.5.2 that a quantum speedup to directed walks would *significantly* improve the random walk step in Walksat and have an impact on many industrial problems which rely on solving unstructured instances of SAT.

3. **Rebalancing Walksat**: The trade-off between the number of iterations and the overall number of random walk steps in Walksat algorithm is important for SAT competitions and industries. Experimentally, it is known that setting an exponential number of walk steps performs better than a theoretical algorithm containing a finite number of steps. In table 5.6 we studied this trade-off by employing Schöning's algorithm on Matlab for small instances. It would be interesting if other powerful algorithm could be analyzed on various instances by altering the number of iterations and number of flips to compute the exact trade-off between number of iterations and flips to obtain a meaningful quantum speedup.

4. **Quantum DPLL**: It would also be interesting if the quantum version of DPLL algorithm that has been proposed in section 3.5.3 employing nested Grover search could be implemented by simulation. If a simulator is capable of exponentially adopting the working of this quantum DPLL SAT solver, we could get an idea about the scaling of the algorithm with increasing SAT instances and compare them to classical DPLL-based solvers such as *minisat* described in section 3.3.

5. **Derivative free optimization**: A few interesting aspects of derivative free algorithms that haven't been considered in this thesis are

   - We considered a quadratic speedup to the number of function evaluations in the search and poll step in derivative free algorithms using the quantum minimum finding algorithm. An improvement to the number of iterations k_{max} in algorithm 4.5 would improve the overall scaling of the algorithm *significantly*. The improvement to the number of iterations would reduce the overall run time even if the algorithm is performed on parallel processors.
   - Quadratically reducing the number of evaluations was shown to be unproductive due to *complete parallelization*. However, it is interesting to note that the equations solved

inside the simulator are a sparse set of linear equations after linearization of the Jacobians, and this is similar to the simulation of inter-storey drift in damper coefficient optimization. An interesting idea worth investigating is if we could employ quantum techniques to improve the algorithms implemented inside the black-box and hence improve the overall running time. A quantum algorithm for solving a system of linear equations was proposed in [HHL09], which scales as logarithm in the dimension of the sparse input matrix. Applying this quantum improvement to practical problems involves thorough analysis of the system of equations in the black-box.

6. **Other practical applications**: There exist many other applications which are worth analyzing and investigating. Many papers have discussed quantum speedups for various problems but a study comparing the state-of-the-art classical implementations and quantum speedup to these problems hasn't been done.

   - *Protein sequencing* relies heavily on algorithms involving string matching. Classically, the KMP algorithm solves the string matching in $O(n+m)$ queries and using quantum techniques and it was shown that the same task could be achieved in $O(\sqrt{n}+\sqrt{m})$ queries in [RV03]. The caveat in this problem however is, given a classical string it takes linear time to read the string, which offsets the quadratic speedup which is obtained in processing. To overcome this hurdle, we are studying the problem of using quantum access to read classical memory with error correction [AGJO$^+$] to faithfully perform Grover's search algorithm.
   - In *computational geometry*, quantum computation has shown promise in problems [SST02] such as forming the convex hull, nearest neighbor problem, furtherest pair problem, minimum spanning tree, segment intersection detection, etc. It would be interesting to investigate the exact numerics to analyze the tradeoffs between the algorithms employed by industries for practical problems such as rooftop designing, medical imaging, robot motion planning, and nanotechnology.
   - In [LMP13], a quantum speedup was proposed to the *shortest vector problem* by employing the quantum minimum finding algorithm as well. The shortest vector problem is widely employed in cryptanalysis and security networks; hence a comparison between the speedup provided in this paper with state-of-the-art classically employed techniques would be interesting. Another area worth considering is in integer programming [HKW03] on finite dimensions, which employs this lattice reduction algorithm as a subroutine.
   - There are many more algorithms where a quantum speedup has been observed over classical algorithms, such as *pattern recognition* [Sch02] which has applications in astronomy [DDM$^+$06], *graph-theoretic problems* [DHHM04] which have applications in

scheduling and network design problems [Dán04], *solving linear equations* [HHL09] which has applications in black-boxes in derivative free optimization [Jia07], in circuit simulations for electrical design [Lit97], *maximum* [AK99] and *minimum* [DH96] *finding algorithm* can be employed in the Concorde algorithm for traveling salesman problem [App06] and integer programming [HKW03], *collision finding* [Amb04] has applications in lattice problems and cryptography [BJLM13] etc. As we have mentioned throughout the thesis, these algorithms work perfectly to give an advantage when considered from a complexity perspective, it however remains to be seen how such algorithmic speedups compare with state-of-the-art classical techniques employed by industries.

Bibliography

[AA06] Mark A Abramson and Charles Audet. Convergence of mesh adaptive direct search to second-order stationary points. *SIAM Journal on Optimization*, 17(2):606–619, 2006. 52, 84

[AAKV01] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59. ACM, 2001. 12

[ABM04] Dimitris Achlioptas, Paul Beame, and Michael Molloy. Exponential bounds for DPLL below the satisfiability threshold. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 139–140. Society for Industrial and Applied Mathematics, 2004. 38

[Ach07] Sriyankar Acharyya. Sat algorithms for colouring some special classes of graphs: some theoretical and experimental results. *J. Satisfiability, Boolean Model. Comput*, 4:33–35, 2007. 32

[ADJ02] Charles Audet and John E Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2002. 51

[ADJ06] Charles Audet and JE Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006. 3, 50, 52

[AGJO$^+$] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priya Varshinee Srinivasan. Quantum random access memory with realistic erros. *In Preparation*. 1, 92

[AHI05] Michael Alekhnovich, Edward A Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *Journal of Automated Reasoning*, 35(1-3):51–72, 2005. 38

[AK99] Ashish Ahuja and Sanjiv Kapoor. A quantum algorithm for finding the maximum. *arXiv preprint quant-ph/9911082*, 1999. 10, 93

[Amb04] Andris Ambainis. Quantum search algorithms. *ACM SIGACT News*, 35(2):22–35, 2004. 28, 93

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007. 2

[AMR13] Srinivasan Arunachalam, Abel Molina, and Vincent Russo. Quantum hedging in two-round prover-verifier interactions. *arXiv preprint arXiv:1310.7954*, 2013. 1

[ANRV04] Pedro Alberto, Fernando Nogueira, Humberto Rocha, and Luís N Vicente. Pattern search methods for user-provided points: Application to molecular geometry problems. *SIAM Journal on Optimization*, 14(4):1216–1236, 2004. 4

[App06] David L Applegate. *The traveling salesman problem: a computational study*. Princeton University Press, 2006. 93

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM (JACM)*, 51(4):595–605, 2004. 2

[AS09] Gilles Audemard and Laurent Simon. Glucose: a solver that predicts learnt clauses quality. *SAT Competition*, pages 7–8, 2009. 27

[BBBV97] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. 9, 80

[BBHT96] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *arXiv preprint quant-ph/9605034*, 1996. 9, 10

[BDJF+99] Andrew J Booker, JE Dennis Jr, Paul D Frank, David B Serafini, Virginia Torczon, and Michael W Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, 17(1):1–13, 1999. 4

[Beb] Henry Yuen & Joseph Bebel. Tough SAT generation. http://toughsat.appspot.com. 17

[Ber87] Vitelmo V Bertero. Observations on structural pounding. In *The Mexico Earthquakes-1985@ sFactors Involved and Lessons Learned*, pages 264–278. ASCE, 1987. 70

[BH97] Gilles Brassard and Peter Hoyer. An exact quantum polynomial-time algorithm for simon's problem. In *Theory of Computing and Systems, 1997., Proceedings of the Fifth Israeli Symposium on*, pages 12–23. IEEE, 1997. 8

[BH99] Ronen I Brafman and Holger H Hoos. To encode or not to encode-i: linear planning. In *IJCAI*, volume 99, pages 988–993, 1999. 32

[BH04] Daniel G Brown and Ian M Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Algorithms in Bioinformatics*, pages 254–265. Springer, 2004. 59

[BH06] Daniel G Brown and Ian M Harrower. Integer programming approaches to haplotype inference by pure parsimony. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 3(2):141–154, 2006. 62

[BHMT00] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *arXiv preprint quant-ph/0005055*, 2000. 5, 10, 11

[BHNT13] K Bigdeli, W Hare, J Nutini, and S Tesfamariam. Optimal design of damper connectors for adjacent buildings. 2013. 72, 74, 84, 85, 87

[BHT12] Kasra Bigdeli, Warren Hare, and Solomon Tesfamariam. Configuration optimization of dampers for adjacent buildings under seismic excitations. *Engineering Optimization*, 44(12):1491–1509, 2012. ix, xi, 70, 72, 73, 74, 78, 83, 84, 85, 86

[Bie13] Armin Biere. Lingeling, plingeling and treengeling entering the sat competition 2013. *Proceedings of SAT Competition 2013; Solver and*, page 51, 2013. 27

[Big12] Kasra Bigdeli. Optimal placement and design of passive damper connectors for adjacent structures. 2012. 4, 52, 71, 72

[BJ06] AV Bhaskararao and RS Jangid. Seismic analysis of structures connected with friction dampers. *Engineering Structures*, 28(5):690–703, 2006. 71, 83

[BJLM13] Daniel J Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum algorithms for the subset-sum problem. In *Post-Quantum Cryptography*, pages 16–33. Springer, 2013. 93

[Bra03] Daniel Brand. Verification of large synthesized designs. In *The Best of ICCAD*, pages 65–72. Springer, 2003. 3

[BT95] Paul T Boggs and Jon W. Tolle. Sequential quadratic programming. *Acta numerica*, 4(1):1–51, 1995. 41

[CBRZ01] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001. 3

[CD10] Jing Chen and Andrew Drucker. Short multi-prover quantum proofs for SAT without entangled measurements. *arXiv preprint arXiv:1011.0716*, 2010. 3

[CDT12] Gregory L Cole, Rajesh P Dhakal, and Fred M Turner. Building pounding damage observed in the 2011 christchurch earthquake. *Earthquake Engineering & Structural Dynamics*, 41(5):893–913, 2012. 70

[CGW00] Nicolas J Cerf, Lov K Grover, and Colin P Williams. Nested quantum search and structured problems. *Physical Review A*, 61(3):032303, 2000. 31, 36, 37

[CK99] Pinhong Chen and Kurt Keutzer. Towards true crosstalk noise analysis. In *Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 132–138. IEEE Press, 1999. 3

[Coh11] David Cohen. *Principles and Practice of Constraint Programming-CP 2010: 16th International Conference, CP 2010, St. Andrews, Scotland, September 6-10, 2010, Proceedings*, volume 6308. Springer-Verlag New York Incorporated, 2011. 32, 33

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971. 2

[CSL13] Shaowei Cai, Kaile Su, and Chuan Luo. Improving walksat for random k-satisfiability problem with $k > 3$. *Proc. of AAAI-13*, 2013. 32

[CSV09] A Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009. x, 42, 43, 45, 46, 48

[Dán04] MARX Dániel. Graph colouring problems and their applications in scheduling. 2004. 93

[Dav54] Chandler Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76(4):733–746, 1954. 45, 48

[DDM+06] S George Djorgovski, Ciro Donalek, Ashish Mahabal, R Williams, Andrew J Drake, Matthew J Graham, and E Glikman. Some pattern recognition challenges in data-intensive astronomy. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 856–863. IEEE, 2006. 92

[Deu85] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985. 8

[DH96] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996. 5, 9, 93

[DHHM04] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. In *Automata, Languages and Programming*, pages 481–493. Springer, 2004. 2, 92

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992. 8

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962. 20

[DLT03] Elizabeth D Dolan, Robert Michael Lewis, and Virginia Torczon. On the local convergence of pattern search. *SIAM Journal on Optimization*, 14(2):567–583, 2003. 3, 45

[DM02] Elizabeth D Dolan and Jorge J More. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002. 78

[DV04] R Duvigneau and M Visonneau. Hydrodynamic design using a derivative-free method. *Structural and Multidisciplinary Optimization*, 28(2-3):195–205, 2004. 4

[DW05] Evgeny Dantsin and Alexander Wolpert. An improved upper bound for SAT. In *Theory and Applications of Satisfiability Testing*, pages 132–137. Springer, 2005. 28, 29

[ECID11] David Echeverria Ciaurri, Obiajulu J Isebor, and Louis J Durlofsky. Application of derivative-free methodologies to generally constrained oil production optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2(2):134–161, 2011. 64

[ES04] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and applications of satisfiability testing*, pages 502–518. Springer, 2004. 27

[FF04] Brian Ferris and Jon Froehlich. Walksat as an informed heuristic to DPLL in SAT solving. *Department of Computer Science, University of Washington, Seattle*, 2004. 27

[FGG+01] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001. 31

[FRK+08] Kathleen R Fowler, Jill P Reese, Chris E Kees, JE Dennis Jr, C Tim Kelley, Cass T Miller, Charles Audet, Andrew J Booker, Gilles Couture, Robert W Darwin, et al. Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743–757, 2008. 4, 42

[Fuk04] Alex S Fukunaga. Efficient implementations of SAT local search. *SAT*, 4, 2004. 32, 33

[GBH+03] Richard A Gibbs, John W Belmont, Paul Hardenbol, Thomas D Willis, Fuli Yu, Huanming Yang, Lan-Yang Ch'ang, Wei Huang, Bin Liu, Yan Shen, et al. The International HapMap project. *Nature*, 426(6968):789–796, 2003. 58, 59

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008. 1

[GN13] David Gosset and Daniel Nagaj. Quantum 3-sat is QMA_1-complete. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 756–765. IEEE, 2013. 3

[Gro97] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997. 2, 8, 9

[GS03] Alfonso Gerevini and Ivan Serina. Planning as propositional CSP: from Walksat to local search techniques for action graphs. *Constraints*, 8(4):389–413, 2003. 27

[Gus04] Dan Gusfield. An overview of combinatorial methods for haplotype inference. *Computational Methods for SNPs and Haplotype Inference*, pages 599–600, 2004. 59, 61

[Her11] Timon Hertli. 3-SAT faster and simpler-unique-SAT bounds for PPSZ hold in general. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 277–284. IEEE, 2011. 29

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009. 92, 93

[HJ61] Robert Hooke and To A Jeeves. Direct Search Solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961. 3, 42

[HKW03] Utz-Uwe Haus, Matthias Köppe, and Robert Weismantel. A primal all-integer algorithm based on irreducible solutions. *Mathematical programming*, 96(2):205–246, 2003. 92, 93

[HN13] W Hare and J Nutini. A derivative-free approximate gradient sampling algorithm for finite minimax problems. *Computational Optimization and Applications*, pages 1–38, 2013. 84

[Hog03] Tad Hogg. Adiabatic quantum computing for random satisfiability problems. *Physical Review A*, 67(2):022314, 2003. 3, 31

[Hoy08] Stephan Hoyer. *Quantum random walk search on satisfiability problems*. PhD thesis, Tese de Doutorado, University of California San Diego, 2008. 35, 36

[HY11] Itay Hen and AP Young. Exponential complexity of the quantum adiabatic algorithm for certain satisfiability problems. *Physical Review E*, 84(6):061152, 2011. 31

[IDEC13] Obiajulu Isebor, Louis Durlofsky, and David Echeverria Ciaurri. Generalized field development optimization using derivative-free procedures. In *2013 SPE Reservoir Simulation Symposium*, 2013. xi, 63, 65, 66, 67

[Ise09] Obiajulu Joseph Isebor. *Constrained production optimization with an emphasis on derivative-free methods*. PhD thesis, Stanford University, 2009. 4, 41, 51, 64

[Ise13] Obiajulu Joseph Isebor. *Derivative-Free Optimization for Generalized Oil Field Development*. PhD thesis, Stanford University, 2013. xi, 50, 51, 52

[IT04] Kazuo Iwama and Suguru Tamaki. Improved upper bounds for 3-SAT. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 328–328. Society for Industrial and Applied Mathematics, 2004. 26

[Jia07] Yuanlin Jiang. *Techniques for modeling complex reservoirs and advanced wells*. PhD thesis, Stanford University, 2007. 65, 93

[KCS73] RE Klein, C Cusano, and JJ Stukel. Investigation of a method to stabilize wind-induced oscillations in large structures. In *MECHANICAL ENGINEERING*, volume 95, pages 53–53, 1973. 70

[Kel99] CT Kelley. Detection and remediation of stagnation in the nelder–mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10(1):43–55, 1999. 43

[Kem03] Julia Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44(4):307–327, 2003. 14

[KLM07] Phillip Kaye, Raymond Laflamme, and Michele Mosca. An introduction to quantum computing, 2007. 7

[KM04] Phillip Kaye and Michele Mosca. Quantum networks for generating arbitrary quantum states. *arXiv preprint quant-ph/0407102*, pages 1–3, 2004. 79, 80

[KSS10] Lukas Kroc, Ashish Sabharwal, and Bart Selman. An empirical study of optimal noise and runtime distributions in local search. In *Theory and Applications of Satisfiability Testing–SAT 2010*, pages 346–351. Springer, 2010. 32

[Lev73] Leonid A Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973. 2

[Lit97] V Litovski. *VLSI circuit simulation and optimization*. Springer, 1997. 93

[LMS06] Inês Lynce and João Marques-Silva. Efficient haplotype inference with boolean satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 104. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006. 3, 59, 62

[LMP13] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Solving the shortest vector problem in lattices faster using quantum search. In *Post-Quantum Cryptography*, pages 83–101. Springer, 2013. 92

[LMY+05] TD Ladd, D Maryenko, Y Yamamoto, E Abe, and KM Itoh. Coherence time of decoupled nuclear spins in silicon. *Physical Review B*, 71(1):014401, 2005. 87

[LPR04] Giuseppe Lancia, Maria Cristina Pinotti, and Romeo Rizzi. Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. *INFORMS Journal on computing*, 16(4):348–359, 2004. 59

[LWZ07] Chu Min Li, Wanxia Wei, and Harry Zhang. Combining adaptive noise and look-ahead in local search for sat. In *Theory and Applications of Satisfiability Testing–SAT 2007*, pages 121–133. Springer, 2007. 22

[Men09] Mohamed El Bachir Menai. A logic-based approach to solve the Steiner tree problem. In *Artificial Intelligence Applications and Innovations III*, pages 73–79. Springer, 2009. 32

[MI06] Rodney Doyle Van Meter III. Architecture of a quantum multicomputer optimized for Shor's factoring algorithm. *arXiv preprint quant-ph/0607065*, 2006. 87

[MMZ+01] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001. 27

[MNRS11] Frederic Magniez, Ashwin Nayak, Jeremie Roland, and Miklos Santha. Search via quantum walk. *SIAM journal on computing*, 40(1):142–164, 2011. 14

[Mon05] Ashley Montanaro. Quantum walks on directed graphs. *arXiv preprint quant-ph/0504116*, 2005. 35, 36

[MS11] Robin A Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 245–252. ACM, 2011. 23

[MSLM09] Joao Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning sat solvers. *SAT Handbook*, pages 131–154, 2009. 20

[MSS00] João P Marques-Silva and Karem A Sakallah. Boolean satisfiability in electronic design automation. In *Proceedings of the 37th Annual Design Automation Conference*, pages 675–680. ACM, 2000. 3

[MSS07] Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007. 2

[MW09] Jorge J Moré and Stefan M Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009. 42

[MWDJM04] Alison L Marsden, Meng Wang, John E Dennis Jr, and Parviz Moin. Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering*, 5(2):235–262, 2004. 4

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010. 7

[NLBH+04] Eugene Nudelman, Kevin Leyton-Brown, Holger H Hoos, Alex Devkar, and Yoav Shoham. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Principles and Practice of Constraint Programming–CP 2004*, pages 438–452. Springer, 2004. 17

[NM65] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. 43

[Nut12] Julie Ann Nutini. A derivative-free approximate gradient sampling algorithm for finite minimax problems. 2012. 84

[Pal04] Avtar Pall. Performance-based design using pall friction dampers-an economical design solution. In *13th World Conference on Earthquake Engineering, Vancouver, BC, Canada*, 2004. 71

[Pap91] Christos H Papadimitriou. On selecting a satisfying truth assignment. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 163–169. IEEE, 1991. 16, 19

[PJ10] CC Patel and RS Jangid. Seismic response of dynamically similar adjacent structures connected with viscous dampers. *The IES Journal Part A: Civil & Structural Engineering*, 3(1):1–13, 2010. 71

[PODDB+12] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, Geordie Rose, and Alán Aspuru-Guzik. Finding low-energy conformations of lattice protein models by quantum annealing. *Scientific reports*, 2, 2012. 90

[PPSZ98] Ramamohan Paturi, P Pudlik, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 628–637. IEEE, 1998. 25

[PPZ97] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 566–574. IEEE, 1997. 29

[RNB11] Daniel Reitzner, Daniel Nagaj, and Vladimír Bužek. Quantum walks. 2011. 14

[Rol03] Daniel Rolf. Improving randomized local search by initializing strings of 3-clauses. 2003. 23

[Rol05] Daniel Rolf. Derandomization of ppsz for unique-k-SAT. In *Theory and applications of satisfiability testing*, pages 216–225. Springer, 2005. 25, 29

[RS12] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, pages 1–47, 2012. 3, 42

[RV03] H Ramesh and V Vinay. String matching in $O(n+m)$ quantum time. *Journal of Discrete Algorithms*, 1(1):103–110, 2003. 2, 92

[RWJ+14] Troels F Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V Isakov, David Wecker, John M Martinis, Daniel A Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *arXiv preprint arXiv:1401.2910*, 2014. 4

[Sar06] Pallav Sarma. *Efficient closed-loop optimal control of petroleum reservoirs under uncertainty*. PhD thesis, Stanford University, 2006. 41

[sata] The international sat competitions. http://www.satcompetition.org/. 26

[SATb] SATLIB. http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html. 17, 62

[SBSV96] Paul Stephan, Robert K Brayton, and Alberto L Sangiovanni-Vincentelli. Combinational test generation using satisfiability. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 15(9):1167–1176, 1996. 3

[Sch99] T Schoning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 410–414. IEEE, 1999. 23, 28, 29, 37

[Sch02] Ralf Schützhold. Pattern recognition on a quantum computer. *arXiv preprint quant-ph/0208063*, 2002. 92

[Sho97] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997. 8

[Sim97] Daniel R Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. 8

[SKC+93] Bart Selman, Henry Kautz, Bram Cohen, et al. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, 26:521–532, 1993. 3, 22, 28, 32, 33

[SM10] Jamie Smith and Michele Mosca. Algorithms for quantum computers. *arXiv preprint arXiv:1001.0767*, 2010. 14

[Spe93] William M Spears. Simulated annealing for hard satisfiability problems. *Cliques, Coloring and Satisfiability: Second DIMACS Implememntation Challenge*, 26:533–558, 1993. 33

[SRN12] Akira SaiToh, Robabeh Rahimi, and Mikio Nakahara. A quantum genetic algorithm with quantum crossover and mutation operations. *arXiv preprint arXiv:1202.2026*, 2012. 91

[SS97] João P Marques Silva and Karem A Sakallah. Grasp-a new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 220–227. IEEE Computer Society, 1997. 27

[SS99] Sanja Singer and Saša Singer. Complexity analysis of Nelder-Mead search iterations. In *Proceedings of the 1. Conference on Applied Mathematics and Computation, Dubrovnik, Croatia*, pages 185–196. PMF–Matematički odjel, Zagreb, 1999. 43

[SS13] John A Smolin and Graeme Smith. Classical signature of quantum annealing. *arXiv preprint arXiv:1305.4904*, 2013. 4

[SSSV14] Seung Woo Shin, Graeme Smith, John A Smolin, and Umesh Vazirani. How" quantum" is the d-wave machine? *arXiv preprint arXiv:1401.7087*, 2014. 4

[SST02] Kunihiko Sadakane, Noriko Sugawara, and Takeshi Tokuyama. Quantum computation in computational geometry. 2002. 2, 92

[Sze04] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 32–41. IEEE, 2004. 13, 14, 35

[Tor97] Virginia Torczon. On the convergence of pattern search algorithms. *SIAM Journal on optimization*, 7(1):1–25, 1997. 3, 45

[UM93] Ron Unger and John Moult. Genetic algorithms for protein folding simulations. *Journal of molecular biology*, 231(1):75–81, 1993. 91

[VDMV01] Wim Van Dam, Michele Mosca, and Umesh Vazirani. How powerful is adiabatic quantum computation? In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 279–287. IEEE, 2001. 31

[WLZ08] Wanxia Wei, Chu Min Li, and Harry Zhang. Switching among non-weighting, clause weighting, and variable weighting in local search for sat. In *Principles and Practice of Constraint Programming*, pages 313–326. Springer, 2008. 32

[WS06] Wei Wei and Bart Selman. Accelerating random walks. In *Principles and Practice of Constraint Programming-CP 2002*, pages 216–232. Springer, 2006. 28

[WX03] Lusheng Wang and Ying Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003. 62

[XHHLB09] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla2009: an automatic algorithm portfolio for sat. *SAT*, 4:53–55, 2009. 27

[XHK99] YL Xu, Q He, and JM Ko. Dynamic response of damper-connected adjacent buildings under earthquake excitation. *Engineering Structures*, 21(2):135–148, 1999. 72

[YXL03] Zhen Yang, YL Xu, and XL Lu. Experimental seismic study of adjacent buildings with fluid dampers. *Journal of Structural Engineering*, 129(2):197–205, 2003. 71, 83

[Zal99] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999. 69

[ZGH11] HP Zhu, DD Ge, and X Huang. Optimum connecting dampers to reduce the seismic responses of parallel structures. *Journal of Sound and Vibration*, 330(9):1931–1949, 2011. 83

[ZRL03] Weixiong Zhang, Ananda Rangan, and Moshe Looks. Backbone guided local search for maximum satisfiability. In *IJCAI*, pages 1179–1186. Citeseer, 2003. 32, 33

[ZY13] Yuesheng Zhu and Deke Yu. An Improved Hybrid SAT Solver for Bounded Model Checking in Circuit Design. 2013. 32