Using a Quantum Genetic Algorithm to Optimize the Spending of a Cyber Security Budget

Ayush Hariharan
Department of Computer Science
Academy of Science
Loudoun County, USA
ahariharan@berkeley.edu

***Abstract*—As difficult choices amidst technological advancements highlight the importance of optimization, the single-choice Knapsack problem emerges as a valuable model for scenarios where inputs are discrete and limited. In this paper, I propose a novel algorithm, Quantum Save, that uses quantum computing and genetic algorithms to optimize this Knapsack problem. With IBM's Qiskit Toolkit, Quantum Save begins by initializing a qubit population and using distributions from quantum measurement as inputs for the Knapsack problem. Reproduction is then simulated by adjusting probability amplitudes based on Knapsack output and implementing a disaster algorithm. After comprehensive testing of Quantum Save with different hyper-parameters, a probability amplitude reduction of 10% and a disaster-step of 5 generations was chosen. Quantum Save's rearrangement of qubits and hyper-parameter tuning resulted in a 49.2% improvement of Knapsack output compared to current implementations of quantum-genetic algorithms. Qubit representation of discrete inputs helps store multiple states while maintaining high accuracy. To emphasize Quantum Save's efficacy in real-world applications, this paper analyzes the algorithm's performance in allocating limited budgets. Using patterns in data collected from recent cyber attacks, Quantum Save suggested the implementation of specific budget-friendly controls to minimize a company's damages. By analyzing the distribution of countermeasures selected at each budget range, there was strong evidence that Quantum Save increased mitigation from attacks. Also, Quantum Save scales linearly, with $\mathcal{O}(n)$ complexity, making it fast and efficient. So, in the context of this case-study, Quantum Save harnesses the computational advantage of quantum computing to effectively optimize the Knapsack problem.**

***Index Terms*—Quantum Computing, Optimization, Cybersecurity, Knapsack Problem, Budget Allocation, Quantum Genetic Algorithm**

I. Introduction

The difficult choices created by technological growth highlights the importance of optimization scenarios. In these scenarios, the single-choice Knapsack problem provides a valuable model because inputs are often discrete and limited. However, the burden of complex maximization functions accompanying the Knapsack problem hinder the development of optimization algorithms. Despite quantum computing's effectiveness in dealing with complex functions during maximization, very few quantum algorithms exist to optimize problems like the Knapsack or the Traveling Salesperson, let alone models that are accurate and efficient.

In this paper, I propose a novel algorithm (Quantum Save) for optimizing the Knapsack problem. Building upon the evolutionary principles in a genetic algorithm and harnessing quantum computing's inherent randomness, the algorithm is effective in optimizing the Knapsack problem. During the development process, comprehensive testing was undertaken to select accurate hyper-parameters for probability amplitude reduction and disaster algorithm implementation. Quantum Save's uniqueness also expands beyond a theoretical premise by testing the algorithm's effectiveness against a case-study in cybersecurity. By analyzing patterns in recent cyber attacks, Quantum Save selects countermeasures to protect a firm against potential hackers. The accuracy and the time complexity of this algorithm was determined by simulating an attack on a firm with Quantum Save's controls implemented. The paper begins by discussing the development of Quantum Save and its implementation on a real-world application. Ultimately, Quantum Save offers an insight into the potential of quantum algorithms in the field of optimization.

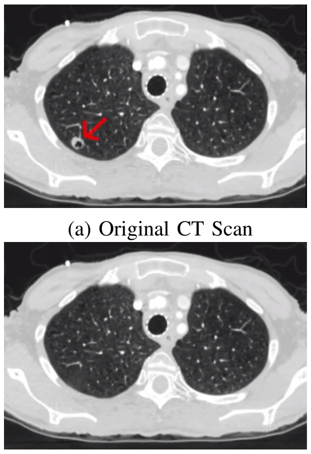
II. Background

A. Technological World

Technology is a powerful agent of change that facilitated the prominence of online communication over face-to-face interactions. The start of the millennia, 2000 to 2004, saw internet traffic expand by 77.6% [1] and online retail sales grow by 225 billion USD [2]. In the public sector, hospital networks create a real-time stream of electronic health records that improve a doctors' ability to take action on critical patient data. CT scanners and x-ray machines help hospitals manage risks and improve organizational performance [3].

Accompanying this emerging technology is the prevalence of cybersecurity – the defense of an organization's data from potential attackers. Hackers gain unauthorized access to a network by targeting a company's vulnerabilities, including susceptible databases and weak firewalls. One security breach endangers the financial information of 20,000 consumers [4] and the medical data of 28,400 patients [5]. With a third of healthcare professionals storing patient data on laptops, smartphones, and USB memory sticks, medical data is easily accessible and more valuable on the black market [5].

However, the leverage held by attackers with medical records is enormous. As Israeli researchers Mirsky et al. [6] proposed, deep learning is an effective tool to add or remove the evidence of medical conditions from 3D medical scans [6]. To understand the danger of existing malware in the practice of tampering, the researchers infiltrated a local hospital, intercepted CT scans, and altered the images [Fig. 1]


(a) Original CT Scan

(b) Altered CT Scan

Fig. 1: Modification on CT Scans by Mirsky et al. [6]

Training a generative adversarial network on 888 CT scans, real-time images from CT scanners were successfully altered with deep learning. Verified by a radiologist's diagnosis during a blind and open trial, the attack had an average success rate of 99.2% for cancer insertion and a 95.8% for cancer removal [6]. This application of malware places lives in danger and emphasizes the far-reaching repercussions of cyber attacks.

Despite the growing danger, 88% of data breaches can be traced back to insider negligence [5] with 75.5% of chief security officers citing a lack of sufficient budget as the top challenge [7]. Technology choices are increasingly complex [8] and this complexity drives the importance of optimization. Optimization plagues many disciplines from cybersecurity to semiconductor development and current models have some faults that are addressed by research proposed this paper.

B. Related Work in Optimization

Optimization problems can take many forms, but often have a function for either maximization or minimization [9]. The objective function is either linear or non-linear and its classification determines both the difficulty and real-world applicability of the problem. In optimization, an effective algorithm's exploration process can search the function landscape to find global maxima instead of local maxima. Evaluation functions can have ambiguous peaks and plateaus along with varying levels of separability and dimensionality [10]. In fact, minimization algorithms used in clustering get rooted at local minima, and are unable to find an accurate solution [11].

One scenario, the Traveling Salesman Problem (TSP), is an NP-hard problem that optimizes a course for a salesman across N cities, visiting each city once, and ultimately returning home [12]. Like the Knapsack Problem, TSP is has real-world potential and many classical algorithms have attempted optimization. However, these algorithms take time and achieve a complexity close to brute force. To address this issue, Srinivasan et al. [12] developed a quantum circuit using qubit eigenstates to approximate unique routes taken by the salesman. The research uses quantum computing to overcome the difficulties present in TSP optimization. Although the quantum implementation was successful, the complexity achieved was quadratic and circuit execution was only successful for four cities [12]. In thethe real-world, run time of this circuit for large input sizes would make the algorithm infeasible. More specifically, Srinivasan et al.'s [12] proposed algorithm would not produce results fast enough to be considered practical.

C. Importance of Vulnerability Protection

In cybersecurity, optimization struggles vary depending upon the target company's characteristics. Bigger targets, or departments, have a constant influx of money from external sponsors and are not limited in monetary spending. Small-Medium targets (SME's), however, are heavily restricted with the available funding for cybersecurity, generally working with a fixed budget [7] that forces trade-offs within the company. Regardless, Small-medium targets (SME's) are an integral aspect of all market economies, employing over 50% of the workforce [13]. In the business-to-business space, SME's are conduits to a larger company's data assets and are repositories of sensitive consumer information. For example, Target's data breach in 2013 occurred when the company's HVAC vendor was infiltrated [14]. Also, hospitals with small cybersecurity budgets contain valuable medical information and patient data.

Despite the value of SME data, 72% of cyber breaches occur at SME's highlighting the insecurity of the information collected [7]. Chief Security Officers (CSO's) are not confident in fixing vulnerabilities under the company's budget constraints, and a Deloitte & NAISCO report [15] finds that 49% of firms only have 6 to 15 specialists working to protect data.

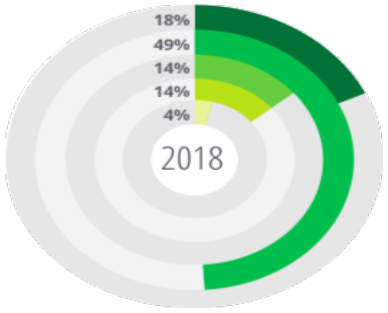
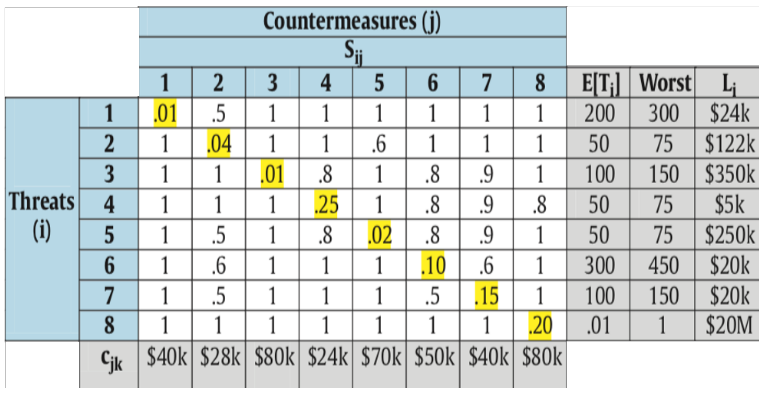


Fig. 2: Top Restriction for Small Targets [15]

Each color in [Fig. 2] signifies a range of cybersecurity specialists. From the top (18%) to the bottom (4%), the first bar (18%) represents 1-5 full-time employees, the second bar (49%) represents 6-15 full-time employees, the third bar (14%) represents 16-25 full-time employees, and the fourth bar (4%) represents 26-50 full-time employees. With a shortage of specialists in cybersecurity, it is important to develop a system that will analyze the trade-offs of each control and advise the implementation of controls that offer the greatest mitigation.

D. Related Work in Cybersecurity

To optimize the implementation of controls, third-party organizations, like KnowBe4, collect data surrounding each security breach and publish these statistics in Endpoint Protection reports [16] [17] [18]. Endpoint Protection reports highlight the frequency of cyber attacks in a given year and the effectiveness of potential countermeasures on company safety. With the 2009 Endpoint Protection report, Rakes et al. [19] transformed qualitative attributes such as countermeasure effectiveness into quantitative values like survival probabilities. The compiled data set expresses the relationships between the different threats and countermeasures in the report.



		Countermeasures (j)										
		S_{ij}										
		1	2	3	4	5	6	7	8	$E[T_i]$	Worst	L_i
Threats (i)	1	.01	.5	1	1	1	1	1	1	200	300	$24k
	2	1	.04	1	1	.6	1	1	1	50	75	$122k
	3	1	1	.01	.8	1	.8	.9	1	100	150	$350k
	4	1	1	1	.25	1	.8	.9	.8	50	75	$5k
	5	1	.5	1	.8	.02	.8	.9	1	50	75	$250k
	6	1	.6	1	1	1	.10	.6	1	300	450	$20k
	7	1	.5	1	1	1	.5	.15	1	100	150	$20k
	8	1	1	1	1	1	1	1	.20	.01	1	$20M
	c_{jk}	$40k	$28k	$80k	$24k	$70k	$50k	$40k	$80k			

Fig. 3: Compiled Cybersecurity Data by Rakes et al. [19]

In Fig. 3, three characteristics summarize the execution of and defense against a cyber attack – survival probabilities, general attack statistics, and cost of implementing countermeasures. Survival probabilities (array in white) are the probability, $P(X, Y)$, of an attack (X) succeeding when a certain control (Y) is implemented with diagonal terms highlighting the optimal control for each threat. General attack statistics (gray-shaded columns) represent the expected ($E[T_i]$) and worst-case ($W[T_i]$) number of attacks in a given year, and the damage incurred from each successful attack (L_i). The gray-shaded row the monetary cost to implement a control.

Following data collection, Rakes et al. [19] developed a linear optimization model to allocate the budgets of cybersecurity firms using quantitative factors such as threat levels and control effectiveness [19]. With company budget as an input, the model selected appropriate countermeasures and projected the expected damages after an attack. To ~~the~~ simulate decision making process of firms, Rakes et al. [19] determined the probability, P_i, that a threat, i, would cause a monetary loss.

$$P_i = \prod_j (1 - Y_i e_{ij}) \tag{1}$$

To linearize, the effectiveness parameter, e_{ij}, is replaced with the proportion of threats that survive an attack. Constraints and other variables are added to simulate an attack's continuation against multiple countermeasures. The model then estimates the monetary loss incurred using the number of surviving attacks and the loss per attack [19].

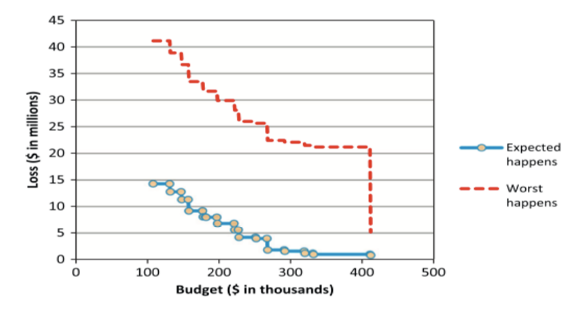


Fig. 4: Model Allocation at Each Budget Range [19]

The process is repeated for controls in which the binary selector, Y_i, is true. The projection of company losses allows Rakes et al. [19] to approximate savings caused by each control. A final constraint prevents the model from exceeding the specified budget.

Rakes et al. [19] proceed to test the linear model on both the expected-case and worst-case number of attacks [Fig. 3] focusing on the relationship between company budget and monetary loss in both cases. The monetary loss incurred by the company was projected at each of 16 different budgets, comprising bigger firms and small-medium targets (SME's).

The analysis of the linear model [Fig. 4] for both the expected and worst case relation revealed that the budget and monetary loss seem to have an inverse relationship. A higher budget creates flexibility for implementing countermeasures and reduces the monetary loss. But patterns in the predictions [Fig. 4] also highlight that Rakes et al. [19] created an inefficient model for small-medium targets, firms that are most susceptible to malware, projecting a high monetary loss with a limited budget. So, a new algorithm is needed to work within the limitations of small-medium targets and accurately select countermeasures that would help the company.

E. Knapsack Problem

As the limitations of small-medium targets emphasize the necessity for optimization, the single-choice knapsack problem is a good model for scenarios where inputs are discrete and limited.

$$\text{maximize} \sum_{j=1}^{n} p_j x_j$$
$$\text{subject to} \sum_{j=1}^{n} w_j x_j \leq W$$
$$1 \geq x_j \geq 0, x_j \text{integral for all} j \tag{2}$$

The single-choice knapsack problem is an NP-complete optimization problem with the goal of finding, in a set of items with given values and weights, the subset of items with the highest total value, under a weight restriction [20]. Once an item is placed in the Knapsack, it may not be selected again.

Both the discrete and choice-based characteristics of the Knapsack problem highlights its validity in the field of optimization. Multiple scenarios that emphasize this model's effectiveness include the design of a system to handle a large influx of data while remaining under budget or the assembly of a fuel-efficient car while choosing from limited parts. In cybersecurity, the Knapsack problem allows the selection model to maximize an attack's mitigation while constrained to a strict budget of a SME. Solutions to the Knapsack problem exist classically; however, algorithms are slow and time-consuming. With quantum computers providing a drastic increase in computational ability, new approaches can be designed to take advantage of a quantum computer's unique properties like superposition and entanglement.

F. Motivating Examples

The complexity and time-consuming nature of optimization problems can be addressed by quantum computing – a methodology that uses fundamental principles of quantum mechanics (superposition and entanglement) to overcome the limitation of a two-state classical system. With the quantum gate model, qubits are mathematically notated as vectors with a horizontal component, α, representing the probability of "0" and a vertical component, β, representing the probability of "1". Inside any quantum system, the squared-magnitude of all probability vectors is 1. When applying quantum computing to the knapsack problem, measurement transforms quantum states into classical states. A quantum system, only acquires definite properties when a measurement is made [21]. Measurement collapses a qubit vector into one of its eigenvalues (0/1) depending upon α and β for that particular qubit.

A current application of quantum computing in optimization involves a quantum-genetic algorithm. To start, standard genetic algorithms are population-based search methods consisting of 5 crucial components: initialization of a population, evaluation, parent selection mechanism, variation operators, and survivor selection [22].

Algorithm 1 Standard Genetic Algorithm

```
 1: $t \leftarrow 0$
 2: $f(x) \leftarrow$ Evaluation function
 3: $population \leftarrow$ Initial Population
 4: while $generation <$ final generation do
 5:     $values \leftarrow f(population)$
 6:     $survivors \leftarrow \max(values)$
 7:     for $individual \in survivors$ do
 8:         mutate(Survivors)
 9:         $population \leftarrow$ Survivor Offsprings
10:     end for
11:     $generation += 1$
12: end while
```

In Algorithm 1, the skeleton of a genetic algorithm is outlined with a generic population "evolving" towards an optimal solution. With a classical computer, reproduction is complex since genetic operators such as crossover probability and mutation effects requires additional steps to mutate the population and achieve an optimal solution. The genetic operators also require hyper-parameter tuning and inaccurately represent the random variability in most populations. A classical genetic algorithm requires complex method definitions and nested loops to replicate evolutionary processes.

With a transition to quantum computing, however, quantum bits and superposition states enhance a genetic algorithm's effectiveness. Following Algorithm 1, an initial population is still created via a quantum system with multiple qubit vectors and a binary digit encoded onto each qubit [23]. A quantum individual is represented by a qubit vector, a quantum family is a row of individuals (row of qubits), and a quantum population is a column of families (array of qubits). When creating the quantum population, each qubit vector is passed through a Hadamard gate, setting both α and β to $\sqrt{2}/2$ resulting in an equal probability of measuring "0" and "1".

Initialization is followed by the measurement of each qubit vector which involves the evaluation of each quantum individual based upon an evaluation function unique to the optimization scenario. In the Knapsack problem, the evaluation function is the maximization expression defined by Equation 2. Directly after measurement, the quantum family (qubit row) which produces the highest Knapsack profit while remaining under the constraint is stored as the "best individual". Reproduction is then initiated by multiplying the probability amplitudes of each qubit vector with a constant, μ, (a hyper parameter in the algorithm). Natural variability in this next generation is simulated using the randomness inherent within quantum computing. This process of evaluation and reproduction proceeds until one of two conditions is met: the population has evolved for 10 generations or it has reached a global maxima [24]. Once the genetic algorithm has been terminated, evolutionary patterns are determined by observing relationship between the best quantum families across generations. Thus, a genetic algorithm's fundamental theory has been adapted with a quantum genetic algorithm (QGA) manipulating qubit vectors, instead of classical values, to increase optimization. Higher-Order QGA's use quantum registers to handle encoding instead of isolated qubit vectors [24].

Despite the QGA's [24] effectiveness, the current setup does not account for stabilization that occurs when an environment reaches a local optimum. A local optimum is when the QGA prematurely converges onto a single value for a long period of time giving the illusion that the absolute maxima has been determined [24]. Therefore, a major caveat of all genetic algorithms is the time of convergence in later states and the potential of finding a local optimum.

Fu et al. [23] addresses this issue and improves local search performance by implementing a self-adaptive rotating angle strategy and a disaster algorithm. The rotating strategy uses linear algebra to decrease the parent-size and the disaster condition imposed a self-made disaster once a premature convergence value is reached. Disasters are simulated in quantum systems by resetting the probability amplitudes of qubit vectors to a linear superposition. After testing on a maximization function, Fu et al. [23] concluded that the rotating technique and the disaster condition improved the times of convergence of the quantum genetic algorithm (QGA), and returned higher optimization values. However, Fu et al. [23] used isolated qubit vectors and the design's simplicity lowered optimization.

G. A Quantum Solution

The research proposed optimizes the Knapsack problem by harnessing the complexity of an IBM Q32 simulator to mimic random variability. Also, the reorganization of a quantum population promotes qubit interactions improving the simulation of evolution. Quantum Save tunes major hyper-parameters and implements a disaster condition via quantum catastrophe operators to prevent premature convergence. In fact, social disasters technique, or applying a catastrophe operator at local maxima, is effective to obtain global extremities [25]. By implementing these modifications, the probability of reaching global maximum increases, and Knapsack profit increases.

The second unique aspect of this research is Quantum Save's testing on real-world case studies. The algorithm improves upon previous investment models that base predictions on simulated values or game theory. Quantum Save addresses issues since existing models only succeed in companies with budget flexibility. Rakes et al.'s [19] model can select controls to maximize mitigation but never faces the full effect of budget constraint. However, Quantum Save uses threat factors and mitigation values from Fig. 3 to improve the budget allocation of SME's who are vulnerable to malware from adversaries.

III. Method

The proposed research is divided into two distinct phases: the development of the Quantum Save algorithm and its testing on a cybersecurity case-study. Phase I involves the replication of the quantum-genetic algorithm, the implementation of a disaster condition to prevent premature convergence, the utilization of randomness inherent with quantum measurement to enhance variability, and the tuning of major hyper-parameters. Phase II involves the transformation of Fig. 3 data into inputs, the comparison of Quantum Save's mitigation with the linear optimization model [19] in both the expected and worst case, and the analysis of Quantum Save's run time at various input-sizes. The results from phase II will validate the results from phase I and explain that Quantum Save is both an efficient and applicable algorithm for optimization.

Phase I consists of six major steps performed multiple times throughout the evolution process. A single iteration of these steps is a generation and through 10 generations, the population should evolve to produce a greater Knapsack profit, thereby optimizing it. The steps in each generation derive from Darwin's Theory of Natural Selection which explains that evolutionary processes are overpopulation, competition, and survival of the fittest. Before implementing Quantum Save on the IBM Q32 computer, various libraries including Qiskit by IBM and NumPy must be imported locally to connect with the quantum computer and execute the custom quantum circuit.

A. Development of Quantum Save

To begin Quantum Save development, a quantum population of 32 qubits (maximum qubits available on IBM Q32 simulator) is initialized with each row representing a quantum family. In Quantum Save, there are four quantum families and each quantum family has eight quantum individuals, or qubits. Multiple quantum families creates natural tension and simulates the impact of evolutionary stress. Depending upon Quantum Save's real-world application, each quantum family can assume multiple identities. With the testing in phase II, the quantum population simulates a Think-Tank where a quantum family represents a Chief Security Officer from a cybersecurity firm and a quantum individual represents a selected countermeasure. The Think Tank's goal is to create a recommendation list of n countermeasures for defense contractors and SME's to protect against adverse attacks.

Algorithm 2 Quantum Save – Part 1

```
1: $n \leftarrow$ number of countermeasures
2: $b \leftarrow$ budget
3: $qregisters \leftarrow n/2$ quantum registers
4: $cregisters \leftarrow n/2$ classical registers
5: while $generation < 10$ do
6:     if $generation = 0$ or $g$ then
7:         $probAmp \leftarrow [0.5, 0.5, 0.5, 0.5]$
8:     end if
9:     $c \leftarrow 0$
10:    for $quantumRegister$ in $qregisters$ do
11:        $classicalRegister \leftarrow cregisters[c]$
12:        $qCirc \leftarrow$ Circuit($quantumRegister, classicalRegister$)
13:        $qCirc \leftarrow$ Hadamard($qCirc$)
14:        $job \leftarrow$ Measure($qCirc, shots = 1024$)
15:        $c \leftarrow c + 1$
16:    end for
17:    *** Algorithm 3 Beginning
18:    *** Algorithm 4 Beginning
19: end while
```

As emphasized by Algorithm 2, the quantum population is initialized with a linear superposition of states (all eigenvalue coefficient are 0.5) and there is an equal probability of measuring the two eigenvalues (1 or 0). Passing the population through a Hadamard gate simulates natural variability and enhances the algorithm's ability to mimic evolution. In Phase II, the first gate in this quantum circuit ensures that the model has an equal probability of implementing the control and omitting it. In fact, the values selected by each quantum family during the first generation is entirely random.

The next step in Quantum Save's development and the first in the reproduction cycle is selecting the fittest quantum families, or those who produce the highest Knapsack profit. This process begins with the measurement of a quantum register through wave-function collapse. Since registers consist of two qubit vectors, depending on α or β of each vector, a value in the set [00, 01, 10, 11] is assigned to the register.

Algorithm 3 Quantum Save – Part 2

```
1: $knapsackProfit \leftarrow []$
2: for $qFamily = 1$ to $4$ do
3:     $binaryString \leftarrow$ " "
4:     for $result$ in $job$ do
5:         $eigenVal \leftarrow$ mostFrequent($result$)
6:         $binaryString \leftarrow binaryString + eigenVal$
7:     end for
8:     $binaries \leftarrow$ Append($binaryString$)
9:     $controlArr \leftarrow$ split($binaryString$)
10:    for $control$ in $controlArr$ do
11:        $item \leftarrow$ [Profit($control$), Weight($control$)]
12:        if Weight($knapsack + item$) $< b$ then
13:            $knapsack \leftarrow$ Append($item$)
14:            $profit \leftarrow profit +$ Profit($item$)
15:        end if
16:    end for
17:    $knapsackProfit \leftarrow$ Append($profit$)
18: end for
19: $bestInd \leftarrow max(knapsackProfit)$
```

The measurement process terminates after the wave function (superimposed state of 1 and 0) is collapsed 1024 times and the most consistent binary pair is returned. The binary pairs from every four registers are combined until four binary strings are obtained, one for each quantum family. Each digit in the binary strings is converted into a set of Knapsack items with a digit "1" corresponding item selection and a digit "0" corresponding to item removal. At each step, the total weight of the Knapsack is compared to the original constraint and items are removed until the constraint is satisfied. With the final set of Knapsack items, a final profit is determined with the evaluation function and mapped to each binary string. For the case-study in phase II, the evaluation function is mitigation received by implementing a certain control and the constraint is the company. The binary string, and respective quantum family, with the highest profit is stored as the fittest and used for probability amplitude manipulation.

Algorithm 4 Quantum Save – Part 4

```
1: for $qFamily = 1$ to $4$ do
2:     $eigenArr \leftarrow$ Split($binaries[qFamily]$, len $= n/2$)
3:     $bestArr \leftarrow$ Split($binaries[bestInd]$, len $= n/2$)
4:     $c \leftarrow 0$
5:     for $eigenVal$ in $eigenArr$ do
6:         if $eigenVal$ != $bestArr[c]$ then
7:             $probAmp \leftarrow probAmp * \mu$
8:             $qReg \leftarrow qRegisters[c]$
9:             $qRegisters[c] \leftarrow$ init($qReg, probAmp$)
10:        end if
11:        $c \leftarrow c + 1$
12:    end for
13: end for
```

Using the fittest quantum family, the probability amplitudes of each qubit register is manipulated. The binary string from each quantum family is split up into pairs with a binary pair corresponding to a quantum individual. The pairs are then grouped based upon the column designation of that individual within the population and the pair from the fittest family (b) is marked for comparison as the "best". For all individual in the column group, except the "best", probability amplitudes for eigenvalues (00, 01, 10, 11) different from b will decrease by a factor of μ and the probability distribution for the register will be normalized to one. To simplify manipulation, as described in Algorithm 4, the probability amplitudes for eigenvalues are stored locally and later initialized onto the register.

In this case, the probability of choosing the optimal Knapsack items increases while the probability of choosing incorrect items decreases. The factor of probability manipulation, μ, is a hyper parameter of Quantum Save and underwent a comprehensive tuning process where different values were tested and the optimization of the Knapsack problem was observed. In the end, by manipulating the probability amplitudes of non-optimal eigenvalues, selections that resulted in higher Knapsack profits were encouraged while selections that resulted in lower profits were discouraged.

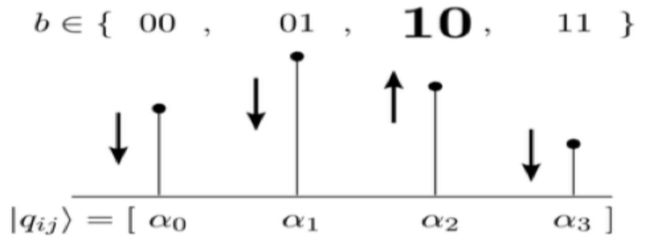

$b \in \{$ 00 , 01 , **10** , 11 $\}$
$|q_{ij}\rangle = [\ \alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \alpha_3\ ]$

Fig. 5: Manipulation of Probability Amplitudes

This cycle of initialization and reproduction that comprises a single generation is completed 10 times to achieve an optimization of the Knapsack problem. To prevent premature convergence onto a local optimum, a disaster condition is implemented at generation g. The disaster condition resets the population to a linear superposition of states thereby introducing random variability into the evolutionary process. The generation of disaster implementation, g is a hyper-parameter of Quantum Save and also underwent a tuning process where different values are tested and Knapsack profit is observed. After 10 generations, the final Knapsack profit is obtained and compared to that of previous research [24]. In the context of the case-study, that research is the linear optimization model proposed by Rakes et al. [19] for the selection of countermeasures under a certain budget. The final profit is also used to determine the values of two major hyper parameters: the value of probability amplitude reduction and the generation of disaster algorithm implementation.

B. Importance of Quantum Computing

When outlining a genetic algorithm, like Quantum Save, the population's is critical to algorithm execution. Evolution

TABLE I: Researcher's Simulation and Tuning Results of QIGA Hyper parameters

Statistics	**Probability Manipulation**					**Disaster Algorithm**			
Trial	80%	90%	95%	99%	*Baseline*	5	6	7	*Baseline*
1	8957	9528	9416	7004	5662	7503	7689	8853	9528
2	7509	7723	8039	5885	6406	7017	7117	8432	7723
3	8631	10436	7108	7261	6920	8642	7277	9290	10436
4	8005	9352	7040	6135	5980	9920	7793	9696	9352
5	8149	7566	7462	8179	9678	8390	8398	8722	7566
6	8299	7324	6941	7030	8775	9323	8031	8154	7324
7	7798	8368	9753	7076	9258	8762	8338	8459	8368
8	9303	8504	7077	8418	7070	7945	8484	6486	8504
9	7799	6020	7161	8649	6954	7044	8232	7910	6020
10	8230	7008	9684	9064	4097	8448	8008	7736	7008
AVG	8150	8252	7975	7641	7086	8518	8235	8292	8252
SE	156.51	310.67	266.80	293.33	414.21	236.19	241.81	214.09	310.67

First 10 trials of 15 total are shown. Values are in dollars of Knapsack profit. Disaster Algorithm baseline is Quantum Save's performance with probability amplitude manipulation. Probability Manipulation baseline is Quantum Save's performance without the manipulation. SE is Standard Error.

is founded upon random variability with mutations and gene crossovers driving natural selection. A generation that contains the same traits is susceptible to either mass extinction or survival while one with random variability has certain traits emerge over others. In mimicking evolutionary processes, a quantum computer enhances the effect of a genetic algorithm.

In a classical genetic algorithm, where bits are isolated to either 0 or 1, random variation is controlled by unique constants and separate methods to simulate mutations and crossovers. These additional steps increase the algorithm's complexity and prevents the efficient execution needed for a real-world application. With quantum computing, random variation is inherent within superposition which simulates a weighted probability distribution between 0 and 1. So, qubit measurement is never concrete but fluctuating based upon the probability amplitudes assigned to each eigenvalue. Although the amplitudes indicate the expected state of the system, there is always uncertainty. With IBM quantum simulation, which mimics a pulse being shot at a suspended electron, the state of an electron cannot be determined beforehand. Slight variations in conditions can return in unexpected values simulating nature's unpredictability. A 90% chance of returning a certain eigenvalue involves a 10% chance of a mutation.

IV. Results

A. Phase I - Hyperparameter Tuning

Phase I of research is accompanied by tuning the model hyper-parameters. The first hyper-parameter is the amplitude reduction value, μ, which describes the extent of probability amplitude manipulation during reproduction. The hyper-parameter's importance derives from its description of the probability amplitude manipulation that occurs between each generation. It helps determine which Knapsack items result in the highest Knapsack profit. In this paper, four values for μ are tested including 0.8, 0.9, 0.95, and 0.99 with 15 trials per value and the final profit recorded after 10 generations.

Initial hyper-parameter search values are based on Nowotniak and Kucharski's initial research [24] explaining that higher probability manipulation (0.99 and 0.95) will increase successful selections by heavily limiting random variability. However, to expand the search radius in hyper-parameter optimization, a lower value of 0.8 is chosen because excessively high values will result in over fitting, where quantum families are too reliant on the best individuals. The final hyper-parameter value of 0.9 is included as a middle ground to prevent over fitting while minimizing the impact of random variability providing unfavorable solutions. This healthy range of values allows for an optimal hyper-parameter value.

The second hyper-parameter is the generation, g, of disaster algorithm implementation. Depending on the generation, the selection of Knapsack items and the ultimate profit will change. A disaster condition implemented at later generations will prevent the behavior change to impact overall optimization and the algorithm will have already converged. However, a disaster condition implemented at the earlier generations will cause unexpected changes in behavior and prevent the algorithm from reaching a potential maxima. To resolve this trade-off, the hyper-parameter search values in Table 1 included high values (7, 6) and then decreased to a lower value of 5. For each value, Quantum Save, with all 10 generations, is run for 15 trials and the final output is recorded.

B. Phase II - Cybersecurity Testing

In phase II of this research, Quantum Save is applied to a cyber security investment problem where a SME has to implement controls that result in low company damages. The Knapsack problem is adapted for cybersecurity investment and the data from Fig. 3 is used as an input. The total mitigation across 10 generations is recorded for a total of 10 trials to determine whether Quantum Save performs better in countermeasure selection than existing models, specifically Rakes et al.'s [19] linear optimization model.

The data in Table 2 shows the relationship between a company's budget (within a SME budget range) and the

TABLE II: Quantum Save vs. Other Budget Allocation Models

Budget	Quantum Save	Linear Model	Brute Force
80k	26.57 M	32.20 M	14.29 M
108k	14.26 M	14.29 M	11.24 M
132k	11.95 M	12.78 M	8.294 M
148k	11.31 M	11.32 M	5.244 M
158k	8.340 M	9.184 M	5.244 M
178k	7.714 M	8.071 M	5.244 M

monetary loss (in dollars) incurred by that company during a potential cyber attack. The relationship between the variables reveals Quantum Save's effectiveness and accuracy when selecting countermeasures. Apart from accuracy, the run time of Quantum Save is a necessary metric to test Quantum Save's speed and time-complexity in the context of this case-study. Data surrounding Quantum Save's scalability is gathered by manipulating the input-size to the Knapsack and observing the change in the algorithm's run time. In phase II, this data describes a relationship between the size of the recommendation list and the Quantum save's run time. The data collection included 10 trials for each input size, varying anywhere from 1 countermeasure on the recommendation list to 8 countermeasures on the list. The run time is measured with internal clocks on both the IBM Q32 quantum simulator and the local machine. The same data collection procedure is also completed on a Brute Force algorithm for baseline comparison. The Brute Force algorithm's ability to test every permutation of countermeasures and return the best selection given the budget constraint allows for an accurate control in both accuracy and scalability of Quantum Save.

V. Analysis

A. Phase I - Hyperparameters

When analyzing Quantum Save's efficiency in Knapsack optimization, the data in Table 1 is compared to a previous attempt at Knapsack optimization by Kucharski and Nowotniak [24]. The previous algorithm differed from Quantum Save with its lack of hyper-parameter tuning, the omission of a disaster condition, a different organization of the quantum population, and a premature version of amplitude manipulation. In fact, with the same profit-weight values mapped to each Knapsack item, the highest profit obtained by the previous model executing on a CUDA architecture was only $5709.

A comparison between Quantum Save and related work in Knapsack optimization [24]) is facilitated through a 1-sample t-test with a significance level of 0.05 using the baseline probability manipulation in Table 1 and the highest profit obtained by a previous model [24] as inputs. Since both p-values (one-tailed and two-tailed), 0.003 and 0.005 respectively, were less than the established significance level, there is statistical evidence to suggest that Quantum Save performed better in optimizing the Knapsack problem when compared to previous research [24]. In fact, the p-values emphasize that there is only a 0.274 to 0.548 percent chance that previous models can obtain a profit higher than Quantum Save's based upon random sampling variability. Quantum Save's performance is likely attributed to changes in population structure, with four quantum families as opposed to three, and the inclusion of a disaster algorithm to prevent premature convergence. These factors increased genetic variability and the chance of finding non-obvious solutions. The modifications resulted in a higher profit and a theoretical optimization of the Knapsack problem.

Quantum Save's improvement is partly attributed to the tuning of an amplitude reduction hyper-parameter, μ and the generation, g at which the disaster condition should be implemented. To demonstrate that tuning is integral to Quantum Save's efficiency, an analysis of variance (ANOVA) test is run with a significance level of 0.10 using the data collected from Table 1 as inputs. The significance level of the ANOVA test is lower than the 1-sample t-test because a higher alpha power is not necessary to show the significance of amplitude manipulation. With a p-value of 0.052 falling under the established significance level, there is enough statistical evidence, within the power of the ANOVA test, to suggest that probability amplitude manipulation allowed for a higher Knapsack profit. In fact, the p-value suggests that there is only a 5.72 percent chance of obtaining a profit this high or higher by employing a QGA without tuned-parameters in probability amplitude manipulation. Since the ANOVA test cannot isolate individual parameters, patterns observed in hyper-parameter data (Table 1) are used to narrow down the search space. The data indicates that a μ value of 0.9 or a 10% reduction in probability amplitudes resulted in the highest profit.

A similar process is completed to demonstrate importance of the disaster condition generation on the final Knapsack profit. Again, an ANOVA test is run with a significance level of 0.10 using the data collected from Table 1 as inputs. With a p-value of 0.848 being over the established significance level, there is not enough statistical evidence to suggest that the generation of disaster condition implementation affected the Knapsack's profit. However, statistical significance is not the best indicator regarding the disaster condition's impact as premature convergence is not guaranteed to occur in every trial. A disaster condition is only helpful when a local optimum is reached while searching. Instead, patterns from the data in Table 1 indicate that the disaster condition creates a 3.22% increase in Knapsack profit suggesting that improvement exists, just not statistically significant improvement. Also, the standard error of Knapsack profit among the 10 trials is lower with a disaster condition suggesting that Quantum Save is more consistent and the probability of finding a non-obvious solution has increased. In fact, the disaster algorithm prevents the model from converging at local maxima thereby decreasing the spread of Quantum Save's outputs.

B. Phase II – Accuracy of Quantum Save

To augment phase I results, Quantum Save is tested against cybersecurity budget allocation. When analyzing the effectiveness and real-world applicability of Quantum Save, two factors are considered: the accuracy in selecting countermeasures to minimize damages and the run-time in which those controls are selected. Accuracy is validated by comparing the allocation techniques of Quantum save with that of previous linear optimization models [19]. These techniques are quantified by analyzing company damages after implementation of the countermeasures suggested by each algorithm. The initial analysis of data in Table 2 suggests that Quantum Save performed better, but variability existed in the company damages when transitioning between budget ranges. To account for variability in output, Quantum Save's percent improvement when compared to Rakes et al.'s [19] model is described in both the expected-case (E) and the worst-case (W).

TABLE III: Percent Improvement by Quantum Save

Budget	% Improvement (E)	% Improvement (W)
80k	17.48	35.80
108k	0.24	46.85
132k	6.53	50.44
148k	0.05	50.32
158k	9.19	47.87
178k	4.42	57.44

Although the extent of percent improvement (Table 3) in the expected-case varied from 0.05% to 17.48%, the positive value range suggests that Quantum Save had better performance for any budget. However, since Table 3 only describes average output difference, Quantum Save's real-world accuracy is still unconfirmed. To determine this accuracy and better understand the variation in percent improvement, a distribution of Quantum Save's output for both the expected and worst-case at each budget range is depicted in the box plot (Fig. 6).

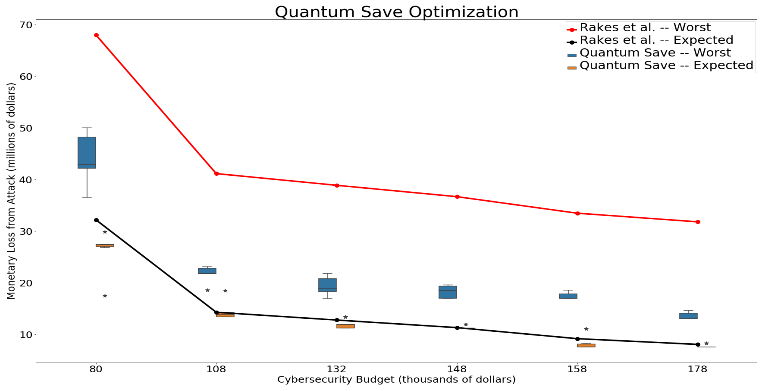
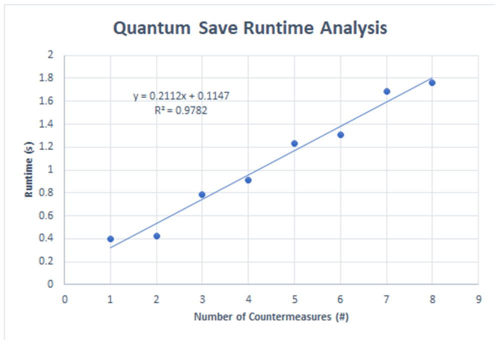
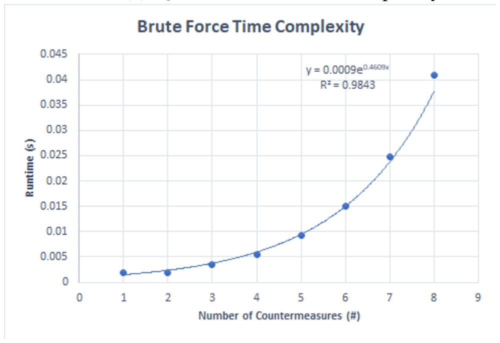


Fig. 6: Compiled Cybersecurity Data by Rakes et al. [19]

After omitting outliers (stars on the box plot), or data points that are outside of a 95 % confidence interval and 1.57 IQR, analyzing Quantum Save's output distribution in the expected-case (orange box-plot) yields that damages under Quantum Save's allocation were either less than or equal to those of the linear model [19] (black trend line). However, the discrepancy between the two models becomes obvious in the worst-case where, barring outliers, damages under Quantum Save's allocation (blue box-plot) were significantly lower when compared to those of previous models (red trend line). In fact, in Table 3, the extent of improvement increased drastically for the worst-case compared to the expected suggesting that



(a) Quantum Save Time Complexity



(b) Brute Force Time Complexity

Fig. 7: Time Complexity of Quantum Save vs. Brute Force

Quantum Save performs better in the worst-case. Since SME's have to prepare for the worst-case when dealing with control selection, the box-plot analysis emphasizes Quantum Save's real-world applicability over existing models.

C. Phase II – Time Complexity

To emphasize model's efficiency in optimization, Quantum Save's accuracy needs to be accompanied by a friendly time-complexity when dealing with recommendation lists between 50 to 100 controls in length. A brute-force algorithm which scales exponentially is time consuming and a luxury that SME's cannot afford. Instead, an algorithm with this target audience needs to select a high number of controls quickly. To find the run time of Quantum Save, data is analyzed to find both an experimental and theoretical relationship between the size of a countermeasure recommendation list and the run time of Quantum Save. With experimental data collected in Phase II testing, the run time of both a traditional brute force algorithm and Quantum Save is charted across different input-sizes. Again, Brute force is creates a good baseline when judging the usability of Quantum Save because it has the best accuracy of any allocation model.

To analyze relationships between input-size and run time of a Brute Force algorithm (Fig. 6), the data collected is fitted with an exponential trend line and a r-squared value of 0.9843 suggests that 98.43% of the variation in run time is explained by a change in input size. The experimental results describing Brute Force's time-complexity aligns with current research explaining that Brute Force scales exponentially. The replication of Brute Force scalability justifies this experimental method of determining an algorithm's time-complexity. So, when analyzing patterns between input-size and run time of Quantum Save, the data collected is fitted with a linear trend line and a r-squared value of 0.9782 suggests that 97.82% of the variation in run time is explained by a change in input-size. The experimental method provides preliminary evidence that Quantum Save scales linearly. However, a theoretical justification is needed to supplement the experimental evidence and establish Quantum Save's scalability in the context of cybersecurity budget allocation.

Lemma 1. *Quantum Save can allocate budget, b, when choosing from countermeasures with input-size, n, in $\mathcal{O}(n)$ time complexity with oracles for quantum measurement of a single register.*

Proof. Apart from the definition of input-size (line 1) and the creation of registers (lines 3 - 4) which are both $\mathcal{O}(n)$, setting up Algorithm 2 (lines 1 - 4) consist of variable definitions and a complexity of $\mathcal{O}(1)$. The main outer loop (beginning in lines 5-19) while iterating through generations executes at most 10 times resulting in $\mathcal{O}(1)$ complexity. The first inner-loop (lines 10 - 16) executes at most n times with quantum register length depending upon input-size creating $\mathcal{O}(n)$ complexity. Each loop iteration creates a quantum circuit (line 12) and measures a quantum register (line 14) which are both solved by $\mathcal{O}(1)$ oracles. The first part of Quantum Save, Algorithm 2, has only one nested loop and a time-complexity of $\mathcal{O}(n)$.

Setting up Algorithm 3 (line 1) has a time-complexity of $\mathcal{O}(1)$ since knapsack variable is independent of input-size. The first loop (lines 2 - 18) executes at most 4 times corresponding to 4 quantum families resulting in a $\mathcal{O}(1)$ complexity. Each loop iteration begins by defining variables (line 3) which is $\mathcal{O}(1)$. Next, a preliminary inner-loop (lines 4 - 6) executes at most n times since the job string's length is directly correlated to input-size creating $\mathcal{O}(n)$ complexity. Each loop iteration appends to an existing binary string (line 6) with an inner time complexity of $\mathcal{O}(1)$. After that, a secondary inner-loop (lines 10-16) executes at most n times since the length of the control array depends on input-size thereby creating $\mathcal{O}(n)$ complexity. Again, each loop iteration appends the countermeasure to a Knapsack resulting in a $\mathcal{O}(1)$ inner complexity. The only other potentially expensive step is *max[profit]* (line 19) in which Algorithm 3 iterates over four quantum families resulting in a complexity of $\mathcal{O}(1)$. The second part of Quantum Save, Algorithm 3, has two nested-loops and a time-complexity of $\mathcal{O}n$.

The first loop in Algorithm 4 (lines 1-13) executes at most 4 times corresponding to the 4 quantum families creating a complexity of $\mathcal{O}(1)$. As the length of $eigenArr$ and $bestArr$ (lines 2 - 3) depend upon input size, the split function is $\mathcal{O}(n)$ complexity. Next, a preliminary inner-loop (lines 5 - 9) executes at most n times since the binary array depends on input-size, resulting in $\mathcal{O}(n)$ complexity. Each iteration of the loop compares the individual to the fittest and assigns a $probAmp$ variable which are both $\mathcal{O}(1)$ steps. The third part of Quantum Save, Algorithm 4, has one nested-loop and a time-complexity of $\mathcal{O}n$.

Since the complexity of all three phases are $\mathcal{O}(n)$, the final time complexity for Quantum Save is $\mathcal{O}(n)$. So, Quantum Save can allocate budget, b, when choosing from countermeasures with input-size, n, in $\mathcal{O}(n)$ time complexity with oracles for quantum measurement of a single register. □

VI. Conclusion

The two phases of research span both the development and testing of an effective optimization algorithm. To emphasize its real-world applicability, Quantum Save has been applied to cybersecurity budget allocation. By analyzing both theoretical improvements of this model in optimization and its success during the case-study, Quantum Save can effectively optimize the Knapsack problem. With Quantum Save's success in Knapsack optimization, it has multiple practical applications apart from the phase II case-study. Quantum Save can be applied in System Design due to the impact of the Knapsack problem. System design is a process that translates a customers' needs into a buildable design that selects subsystems from an allowable set while remaining under a corporate budget. Chapman et al. [26] explains how system design problem is NP-complete similar to the Knapsack problem. Both problems attempt to maximize an evaluation function, either corresponding to required system performance or Knapsack profit, while remaining under a certain constraint. Computer science is an ever-evolving field of study and finding the optimal solution to these problems, such as the Knapsack, is extremely beneficial.

Also, phase II of testing suggests that Quantum Save is an accurate model for cyber security budget allocation. Quantum Save can create an optimal investment plan and protect a SME from malicious attacks. Known system vulnerabilities be addressed by Quantum Save while the company performs research and development without worrying about network security. In this online world, the importance of keeping information safe cannot be emphasized. Companies with vulnerable data will find it difficult to gain contracts and build meaningful business relationships. In fact, the reputation of SME's as vendors is reliant on the structural integrity of their backend system and the confidentiality of their data. With an investment plan proposed by Quantum Save, SME's can strengthen their backend and increase their reputation. In a world where information is valuable currency on the black market, companies need to invest in protecting that information. So, by optimizing the Knapsack problem effectively, Quantum Save can help SME's millions of dollars by allocating their budgets.

VII. Acknowledgements

I would like to thank Duke Writer from the Academy of Science for his generous support throughout the duration of this research endeavor. He has been with me since the beginning of this project and has helped me overcome many obstacles. Throughout this process, he introduced me to the importance of research in computer science and truly conveyed the meaning of inquiry. I would also like to thank Dr. Matt Weber from UC Berkeley for enhancing my understanding of theoretical computer science. He helped refine my project from simply experimental data to theoretical application. He helped me expand the breath of my project so that it could reach a whole new audience.

REFERENCES

[1] H. Mia, M. A. Rahman, and M. Uddin, "E-Banking: Evolution, Status and Prospect," *Journal of ICMAB*, vol. 35, 2007.

[2] A. Rohm and V. Swaminathan, "A Typology of Online Shoppers Based on Shopping Motivations," *Journal of Business Research*, vol. 57, pp. 748–757, 2004.

[3] R. Bradley, T. Byrd, J. Pridmore, E. Thrasher, and R. Pratt, "An Empirical Examination of Antecedents and Consequences of Its Governance in US Hospitals," *Journal of Information Technology*, vol. 27, 2012.

[4] R. Hasan and W. Yurcik, "A statistical analysis of disclosed storage security breaches," 01 2006, pp. 1–8.

[5] A. R. Miller and C. E. Tucker, "Encryption and the loss of patient data," *Journal of Policy Analysis and Management*, vol. 30, no. 3, pp. 534–556, 2011. [Online]. Available: http://www.jstor.org/stable/23018963

[6] Y. Mirsky, T. Mahler, I. Shelef, and Y. Elovici, "CT-GAN: Malicious Tampering of 3D Medical Imagery using Deep Learning," *arXiv:1901.03597 [cs]*, Jun. 2019, arXiv: 1901.03597. [Online]. Available: http://arxiv.org/abs/1901.03597

[7] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, "Decision support approaches for cyber security investment," *Decision Support Systems*, vol. 86, pp. 13–23, Jun. 2016.

[8] G. Yeric, B. Cline, S. Sinha, D. Pietromonaco, V. Chandra, and R. Aitken, "The past present and future of design-technology co-optimization," in *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference*, 2013, pp. 1–8.

[9] N. Krivulin, "Tropical optimization problems," 08 2014.

[10] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *Int. J. of Mathematical Modelling and Numerical Optimisation*, vol. 4, 08 2013.

[11] C. Chen and F. Ye, "Particle swarm optimization algorithm and its application to clustering analysis," in *2012 Proceedings of 17th Conference on Electrical Power Distribution*, 2012, pp. 789–794.

[12] K. Srinivasan, S. Satyajit, B. Behera, and P. Panigrahi, "Efficient quantum algorithm for solving travelling salesman problem: An ibm quantum experience," 05 2018.

[13] E. Taylor, "Haccp in small companies: benefit or burden?" *Food Control*, vol. 12, no. 4, pp. 217 – 222, 2001, third International Food Safety HACCP Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0956713500000438

[14] R. Banham, "Cybersecurity threats proliferating for midsize and smaller businesses," *Journal of Accountancy*, vol. 224, no. 1, p. 75, 2017.

[15] "Deloitte NASCIO Cybersecurity Survey | Deloitte Insights."

[16] "Nyorton endpoint Security Report." [Online]. Available: www.nyotron.com/collateral/2019-Endpoint-Security-Report-NYOTRON.pdf

[17] KnowBe4, "2018 Threat Impact and Protection Report." [Online]. Available: www.knowbe4.com/hubfs/2018ThreatImpactandEndpointProtectionReport.pdf

[18] New-Ponemon-Institute, "The 2017 State of Endpoint Security Risk." [Online]. Available: https://cdn2.hubspot.net/hubfs/468115/Campaigns/2017-Ponemon-Report/2017-ponemon-report-key-findings.pdf

[19] T. Rakes, J. Deane, and L. Rees, "IT security planning under uncertainty for high-impact events," *Omega*, vol. 40, pp. 79–88, 2012.

[20] C. Murawski and P. Bossaerts, "How Humans Solve Complex Problems: The Case of the Knapsack Problem," *Scientific Reports*, vol. 6, no. 1, pp. 1–10, Oct. 2016. [Online]. Available: https://www.nature.com/articles/srep34851

[21] R. Forrester, "The Quantum Measurement Problem: Collapse of the Wave Function Explained," *Social Science Research Network*.

[22] A. Garg and S. Goyal, "Vector Sparse Representation of Color Image Using Quaternion Matrix Analysis based on Genetic Algorithm," 2017.

[23] H. Wang, J. Liu, J. Zhi, and C. Fu, "The Improvement of Quantum Genetic Algorithm and Its Application on Function Optimization," *Mathematical Problems in Engineering*, vol. 2013, p. 730749, May 2013. [Online]. Available: https://doi.org/10.1155/2013/730749

[24] R. Nowotniak and J. Kucharski, "Higher-Order Quantum-Inspired Genetic Algorithms," *IEEE*, 2014.

[25] M. Rocha and J. Neves, "Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation," in *Multiple Approaches to Intelligent Systems*, 1999, pp. 127–136.

[26] W. L. Chapman, J. Rozenblit, and A. T. Bahill, "System design is an NP-complete problem," *Systems Engineering*, vol. 4, no. 3, pp. 222–229, 2001.