A Quantum Computer's Guide to Saving Money

Using a Quantum Genetic Algorithm to Optimize the Allocation of Cybersecurity Budget

The Value of Data – Financial Ruin

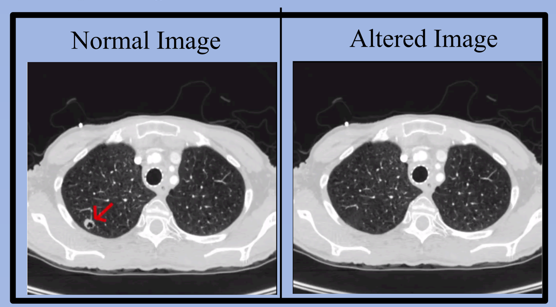
The Value of Data – Life or Death

How difficult is it for a doctor to tell a patient that he or she is dying from lung cancer?

- The patient did everything right. CT Scans from early screening came back as normal
- Malware manipulated the CT scan causing a misdiagnosis of the patient

Israeli Researchers Mirsky et al. investigated this growing fear by successfully infiltrating a hospital network, intercepting CT Scans, and manipulating them.

- Exploited one of five potential weaknesses in the infrastructure
- Used a CT-GAN to cover up and introduce cancerous nodules in lung scans
- Success rate is 99.2% for cancer injection and 95.8% for cancer removal



With malware, hackers can not only steal your personal information, but also have your life in their hands

Consolidating Cybersecurity Data

In wake current attacks, third-party organizations like KnowBe4 publish statistics on the specifics of each attack in Endpoint Protection Reports.

Rakes et al. used one of these reports to compile a dataset with three major parts:

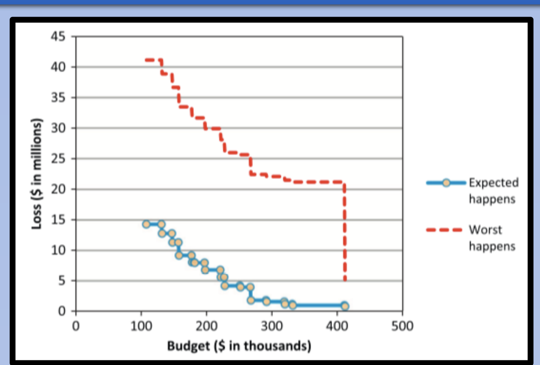
1. Survival Probabilities – Array in White
   - Survival probability, P(X, Y) is the probability of an attack (X) succeeding when a countermeasure (Y) is implemented
2. General Statistics – 3 Grey Columns
   - Expected number of attacks in a year, $E[T_i]$
   - Worst-case number of attacks in a year
   - Damage incurred from of a successful attack, L_i
3. Countermeasure Price
   - The grey row at the bottom of the table
   - The price of installing each countermeasure

		Countermeasures (j) S_{ij}										
			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			

Previous Research

Using the compiled cybersecurity data, Rakes et al. developed a linear optimization model with:

- INPUT: Company's Budget
- OUTPUT: Expected Damage in the Case of an Attack



The linear optimization model followed three major steps in order to allocate the budget effectively:

A. Proportion of Surviving Threats

- The proportion (P_i) of threats that survive if countermeasure j is implemented at level k
- Each countermeasure has an effectiveness (e_{jk}) and a decision variable (Y_{jk})
- $P_i = \sum (1 - Y_{jk}\, e_{jk})$

B. Damage to Company by Surviving Threats

- The monetary loss (both explicit and implicit) from the surviving threats after a countermeasure is implemented
- Loss = $P_i * L_i * E[T_i]$

C. Cost of Implementing Countermeasure

- Cost = $\sum c_{jk} * Y_{jk}$
- The total cost of all countermeasures must remain under a certain budget, B.

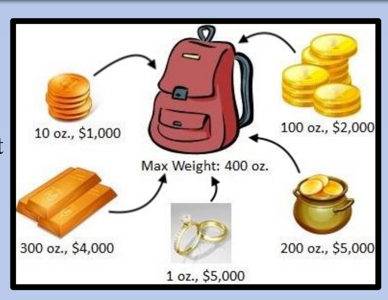
The model works better for higher budget firms NOT small to medium sized enterprises

A Unique Solution

This is a classic example of the Knapsack Problem

<u>What is the Knapsack Problem?</u>

- An NP-Complete Problem where a certain value is maximized while remaining under a certain budget



In the application of cybersecurity, the amount of money saved is maximized but the company is forced to remain under a budget constraint

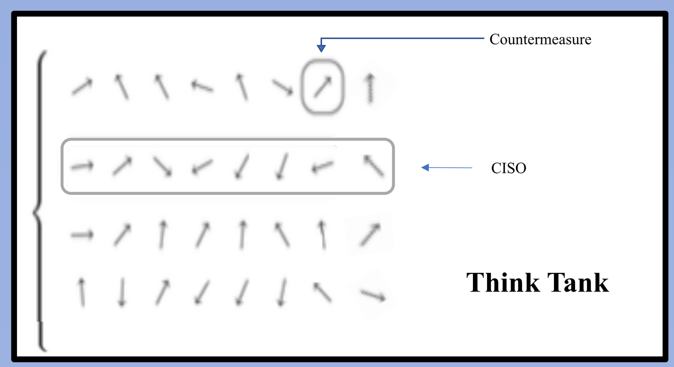
<u>Quantum Computing– The New Solution</u>

- A quantum algorithm can break a 2048 RSA encryptions in a matter of seconds
- But now, this emerging technology in cybersecurity can be used to defend against potential attacks

Quantum Computing could be used to optimize the Knapsack Problem and allocate cybersecurity budgets more effectively

Quantum Genetic Algorithm

The Quantum Population



1. <u>Quantum Population Specifics</u>
   - The quantum population is created on the IBM Q32 simulator using the IBM Qiskit API
   - Why a Quantum Simulator instead of a Quantum Computer?
     - COMPUTER: Can only maintain a quantum state for a short period of time before quantum decoherence.
     - SIMULATOR: Can simulate a quantum state for an extended time period
     - In the future, Quantum Computers can maintain superposition and algorithms, like Quantum Save, can be transferred onto these new devices.
   - Each population uses an array of 32 qubits arranged in 4 rows and 8 columns to simulates a Think Tank
   - The goal of this Think Tank is to create a recommendation list of 8 countermeasures that US companies can implement
     - ROW: Corresponds to a Chief Security Officer in this hypothetical Think Thank
     - COLUMN: Corresponds to a potential countermeasure on the recommendation list
2. <u>Linear Superposition</u>
   - After the creation of the population, each qubit is measured and assigned a binary value, either 0 or 1.
     - If the qubit is measured as a 1, then the countermeasure is added to the recommendation list
     - If the qubit is measured as a 0, then the countermeasure is removed from the recommendation list
   - Depending on the values of the qubits in each row, the CISO puts together the list

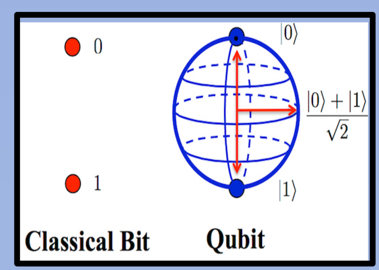
Random Variability

After the population is created, the second aspect of Darwin's Theory of Natural Selection is random variability in genetic traits.

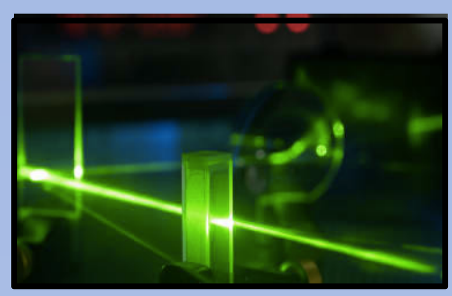
- If every generation contained the same traits, then evolution would never occur.
- Instead, mutations and gene crossovers drive natural selection.
- The reproduction process consists of a high-degree of randomness which is often difficult to replicate in a genetic algorithm without extensive hyperparameter tuning.

This is where quantum computing significantly enhances the speed and accuracy of a genetic algorithm

1. <u>The Power of the Qubit</u>
   - A bit is isolated to 0 or 1, but a qubit is the infinite states between 0 and 1
     - Each qubit is initialized with a 50% chance of returning 1 and a 50% chance of returning 0



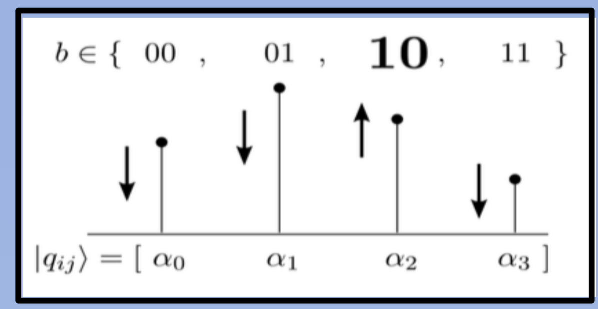
   - As the algorithm progresses, the probabilities change, with a potential 90% chance of returning 1 and a 10% chance of returning 0.
     - This change is controlled by manipulating the probability amplitudes of each qubit or qubit register in the population
   - In a classical genetic algorithm, random variation is controlled by unique hyperparameters
     - However, by using quantum computing, the random variation is inherent
     - The measurement value of a qubit is never concrete but fluctuates based upon probability amplitudes
2. <u>The IBM Quantum Simulator</u>
   - Uses a dilution refrigerator to suspend an electron



   - The measurement of the electron is then simulated
   - If the electron is spin-up, the qubit returns a zero
   - If the electron is spin-down the qubit returns a one

Survival of the Fittest

1. <u>Finding the Best Individual</u>
   - Once the population is created and initialized with the probability amplitudes, each recommendation strategy developed by a CISO is analyzed
   - The CISO strategy that results in the least cybersecurity loss is recorded as the best individual
     - The best individual is used as a point of comparison for the rest of the algorithm



2. <u>Evolving the Population</u>
   - The recommendation list on the best individual is iterated through and compared to the recommendation list on the other CISO's in the Think Tank
     - This is done sequentially.
     - The first countermeasure on the best individual's list is compared to the first countermeasure on all other CISO lists
     - The iteration looks at the qubit value (1 if the countermeasure is included and 0 if the countermeasure is excluded)
   - Depending on the value of the qubit, the probability amplitude (% chance of getting 1 or 0) of the other value is decreased
     - If the best individual implemented the countermeasure, the algorithm increases the probability that all other CISO's implement the countermeasure
   - Probability amplitudes are chanced so that the sum of all probabilities remain equal to 1.

Quantum Save Can Help Hospitals and Defense Contractors Save MILLIONS by Allocating the Budgets of Small and Medium Sized Firms!

Results -- Accuracy

Budget	Improvement (%)
80k	17.48
108k	0.24
132k	6.53
148k	0.05
158k	9.19
178k	4.42

When looking at the expected case, a simple eye test suggests that Quantum Save performed better when compared to previous research.

- Extent of that improvement varied by a decent margin
- To confirm that the improvement wasn't due to random variability, I looked at the distribution of outputs at each budget range
- Outliers were determined at 95% confidence level or 1.57 IQR from the median

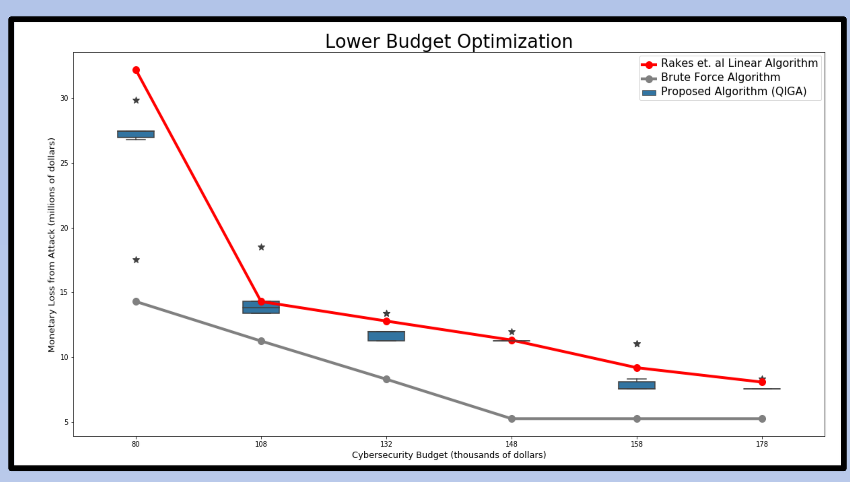
However, expected case can only provide so much information: WORST case is a better indicator of the effectiveness of an algorithm in budget allocation.

- Extent of improvement varied by a significant margin
- Since Quantum Save performed better in the worst-case, it will be more effective in the real-world where companies need to prepare for the worst-possible outcome
- To confirm that the improvement wasn't due to random variability, I looked at the distribution of outputs at each budget range

Budget	Improvement (%)
80k	35.80
108k	46.85
132k	50.44
148k	50.32
158k	47.87
178k	56.97

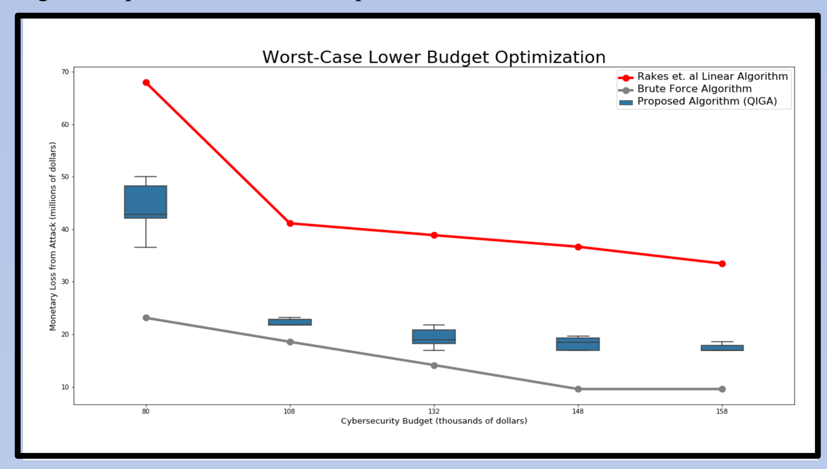
When looking at the distribution of my algorithm's output at each budget (barring outliers), my algorithm performed better than or equal to previous work

- Better is quantified as the algorithm that chose countermeasures that resulted in LOWER cybersecurity loss



My algorithm had better accuracy when compared to that of previous research

When looking at the distribution of my algorithm's output at each budget (barring outliers), my algorithm performed better than previous work in this field.



- There is a significant jump between lower-budget optimization in the worst-case when compared to the expected case
- Likely because the Knapsack problem looks at the scenario of budget allocation from a different angle.

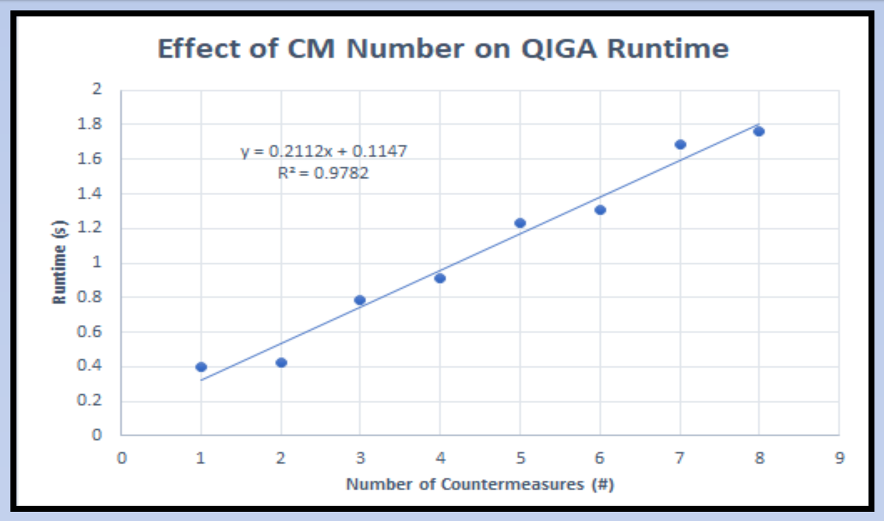
Results -- Experimental Time Complexity

Any good algorithm finds a way to balance the trade-off between accuracy and time complexity.

- Algorithms that are the most accurate often take the longest time to run
- Algorithms that run very quickly are often the least accurate models of the situation

To find the time-complexity of my algorithm, I started looking at experimental data of how the input size (number of countermeasures) affected run-time.

- I then compared the runtime patterns found in my algorithm to the runtime patterns found in brute force (a known algorithm with exponential time complexity



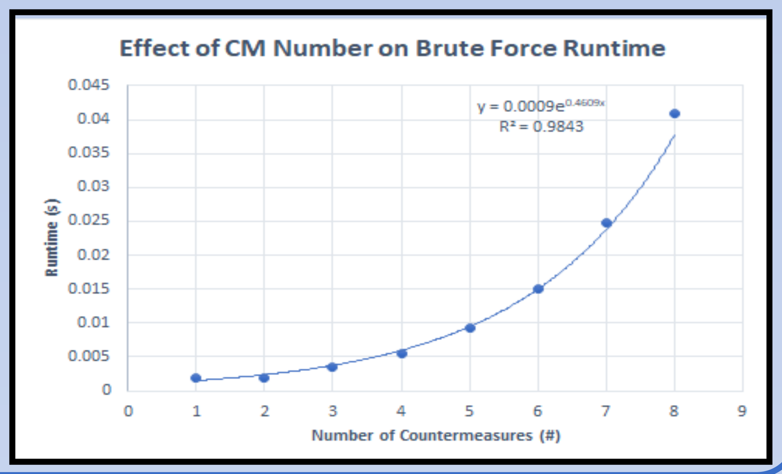
Experimental data was collected on the relationship between input size (number of countermeasures) and runtime

- From a preliminary analysis, and an R-squared value as a metric, it appears that my algorithm scales linearly
- However, a full mathematical proof would be needed to supplement this data

Quantum Save achieved Linear Time Complexity

Again, experimental data was collected on the relationship between input size and runtime for a brute force algorithm

- From a preliminary analysis, and an R-squared value as a metric, it appears that brute force scales exponentially
- This confirms the theoretical research performed about the scalability brute force algorithms



Results -- Theoretical Proof

***Lemma*:** Quantum Save can allocate budget, b, when choosing from countermeasures with input-size, n, in O(n) time complexity with a linear time oracle for quantum measurement of a single register.

Algorithm 2 Quantum Save – Phase 1

1: $n \leftarrow$ number of countermeasures
2: $b \leftarrow$ budget
3: $probAmp \leftarrow [0.5, 0.5, 0.5, 0.5]$
4: $generation \leftarrow 0$
5: **while** $generation < 10$ **do**
6: $\quad quantumRegister \leftarrow 0$
7: $\quad$ **while** $quantumRegister <$ n **do**
8: $\quad\quad classicalRegister \leftarrow$ classicalRegister[n]
9: $\quad\quad quantumCircuit \leftarrow$ classicalRegister + quantumRegister
10: $\quad\quad quantumCircuit \leftarrow$ probAmp
11: $\quad\quad job \leftarrow measure(quantumCircuit)$
12: $\quad quantumRegister{+}{+}$

Line 1: the number of countermeasures is defined.

- Depends upon the input-size of the recommendation list = O(n) complexity

Lines 2 – 4: basic Quantum Save variables defined.

- Variables are not correlated to the input-size = O(1) complexity

Lines 5 – 6: start of Quantum Save

- Loops through the number of generations (constant)
- Quantum registers initialized.
- O(1) complexity

Line 7: second loop iterates through every quantum register. Quantum registers depend on the input size = O(1) complexity.

Lines 8 – 11: Quantum circuit created and measured. Measurement of a single register is independent of input-size = O(1) complexity.

Algorithm 3 Quantum Save – Phase 2

1: $CISO \leftarrow 0$
2: **while** $CISO < 4$ **do**
3: $\quad binaryString \leftarrow$ ""
4: $\quad result \leftarrow$ ""
5: $\quad$ **while** $result$ in job **do**
6: $\quad\quad binaryString \leftarrow binaryString$ + Frequent String
7: $\quad countermeasure \leftarrow 0$
8: $\quad knapsack \leftarrow 0$
9: $\quad$ **while** $countermeasure$ in binaryString **do**
10: $\quad\quad knapsack$ += $countermeasure$
11: $\quad\quad$ **if** $knapsack > b$ **then**
12: $\quad\quad\quad knapsack$ -= $countermeasure$
13: $\quad\quad profit \leftarrow$ mitigation[$countermeasure$]
14: $\quad bestIndividual \leftarrow max(profit)$

Lines 1 – 2: start of phase 2 of Quantum Save

- CISO Variable Initialized
- CISO's in think-tank looped through (constant number)
- O(1) complexity

Line 5: iterate through results of quantum measurement

- Job string dependent on input-size = O(n) complexity

Line 9: iterate through countermeasures in the input array

- Input array directly correlates with input-size = O(n) complexity

Line 14: the maximum profit value is determined. Number of CISO's remain constant = O(1) complexity

Algorithm 4 Quantum Save – Phase 3

1: $binaryArray \leftarrow binaryString.split()$
2: $CISO \leftarrow 0$
3: **while** $CISO < 4$ **do**
4: $\quad countermeasure \leftarrow 0$
5: $\quad$ **while** $countermeasure$ in $binaryArray$ **do**
6: $\quad\quad$ **if** countermeasure != bestIndividual **then**
7: $\quad\quad\quad probAmp_{man} \leftarrow$ manipulate[$probAmp$]
8: $\quad\quad probAmp \leftarrow probAmp_{man}$

Proof: After looking at the three algorithmic blocks, it was noticed that the algorithm was split into 5 different section. When analyzing each of the 5 major sections of Quantum Save, it was concluded that the time complexity is O(n). **Quantum Save was able to achieve linear time complexity by taking advantage of quantum computing.**

Conclusion

References