Ayush Hariharan
10/18/18

Using the Nelder-Mead Method to Optimize the Hyper-Parameters of Higher Order Quantum-Inspired Genetic Algorithms (QIGA-2)

I. Introduction to Quantum Systems

With the increasing complexity of mathematical problems, quantum computing has become a popular solution to avoid long periods of computation. A quantum computer uses fundamental principles of quantum mechanics, including superposition, to maximize computing power. In a traditional computer, data is encoded in bits, 0 and 1, which represent the on and off positions of a transistor; however, a quantum computer uses quantum bits, or qubits, to represent any of the infinite states between on and off. Each qubit is mathematically notated as a vector, with a horizontal component that represents the probability of being a 0 and a vertical component that represents the probability of being a 1. From this point onward, the horizontal component will be referred to as alpha, and the vertical component will be referred to as beta. Although it is impossible to determine the exact state of a qubit, vector notation allows scientists to approximate this value using the direction and magnitude of the qubit. In a quantum system, or a space with multiple qubits, a major characteristic is that the magnitude of all of the probability vectors in the system must equal 1. In this experiment, Dirac notation will be used where a bra vector represents a row of qubit vectors and a ket vector represents a column.

Another important aspect of any quantum system is the idea of measurement, which allows for a transformation of quantum states into classical states. A quantum entity, or system, only comes into existence and acquires definite properties when an observation or measurement is made (Forrester, 2017). An important part of quantum mechanics are wave functions which gives the mathematical probabilities of the state of a quantum entity before an observation has made (Forrester, 2017). Each possible probability in the wave function is commonly referred to

as an Eigen function. In order to obtain a measurable value from the system, the wave function must be collapsed into a specific Eigen function. For example, when a function is anti-differentiated, the antiderivative obtained could have various vertical translations. The abstract notation, using C to represent the translation, is an example of a wave function with a specific antiderivative being an Eigen function. However, to collapse this wave function a value for C must be determined.

In quantum mechanics, a collapse of the wave function cannot be done by substituting a value; instead, energy must be added to a system. Energy is usually represented by a Hamiltonian quantum operator, which is a parameter used to translate quantum wave functions into classical Eigen functions. By adding energy to the system, the possibilities of a wave function are localized into a single particle and this localization results in measurement. However, the results from a measurement of the wave function is not random. According to the Copenhagen interpretation, every Eigen function has a certain probability of occurring during the measurement process. In terms of quantum computing, the separate values of 0 and 1 can be represented by a discrete particle while a qubit, or the infinite values between 0 and 1, can be represented by a wave function. Therefore, during measurement, if alpha squared is higher than beta squared, the measurement of the system returns a 0; however, if beta squared is higher than alpha squared, the measurement of the system returns a 1.

II. General Application of Quantum Systems to Computing

By using the idea of probability and taking advantage of a qubit's unique properties, a quantum computer will arrive at an accurate answer much quicker than a classical computer. With algorithms such as the Fourier Transform and Shor's Algorithm, quantum computers have solved problems that have stumped classical computers for many years (Montanaro, 2016).

Although a quantum computer can increase the rate of computation, a caveat with this system is that quantum computers heavily rely on probability which means that an answer has the possibility of being incorrect (Montanaro, 2016). For example, the black-box function used in Shor's algorithm provides an output with an error range, thereby allowing a scenario to exist where the returned value is incorrect.

As different aspects of quantum computing are mastered, problems have started to increase in difficulty, and algorithms have started to increase in complexity. One problem that has become prominent in the field of optimization is the knapsack problem. The knapsack problem is a combinatorial 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 (Bossaerts & Murawski, 2016). In this problem, a value has to be optimized while constrained. A common interpretation of the knapsack problem is to maximize the fuel available to use in a rocket while remaining within a certain budget. Scientists have tried to solve the knapsack problem classically and failed; however, with quantum computers providing a drastic increase in computational ability, new approaches have been designed to take advantage of a quantum computer's speed and general reliability.

The knapsack problem can be put into function notation. The function for total profits is represented by the sum of the individual values multiplied by the measurement of the quantum system. Also, each individual value is represented by an assigned weight, a random number between 1 and 10, plus 5. Finally, the weight restriction is represented by half the sum of the individual weights. In other words:

$$w_i = random[1,10)$$

$$p_i = w_i + 5$$

$$x_i = binary\ string\ obtained\ from\ measurement$$

$$f(x) = \sum_{i=1}^{m} p_i x_i$$

$$C = \frac{1}{2}\sum_{i=1}^{m} w_i$$

III. Use of Simple Quantum-Inspired Genetic Algorithms for Optimization

An efficient and quick approach to this problem uses quantum-inspired genetic algorithms (QIGA). A standard genetic algorithm is a population-based search method which consists of 5 crucial components: initialization of a population, evaluation, parent selection mechanism, variation operators and survivor selection (Garg & Goyal, 2017). Once the population is initialized and the first evaluation is completed, a genetic algorithm takes advantage of the concepts of evolution and modifies a selected part of the population. After the selected portion is determined using a parent selection mechanism, those individuals are modified using variation operators and the algorithm is run again. Eventually, the population should "evolve" towards an optimal solution. Although genetic algorithms were a good way to optimize problems, classical computers did not have the computational ability to cycle through enough generations for the algorithm to be effective.

When applied to quantum mechanics, however, a quantum genetic algorithm can be used based upon the concepts of quantum bits and quantum superposition states. Using qubit encoding, where each qubit uses a vector to represent binary code, a quantum algorithm can use a quantum system with multiple qubit vectors as an initial population (Fu, Liu, Wang, & Zhi, 2013). Specifically, each quantum gene is represented by a qubit vector, each quantum chromosome is represented by a bra vector, and each quantum population is represented by a

column of chromosomes. When the quantum population is first created, each vector is initialized with a linear superposition of states, essentially setting both alpha and beta to $\sqrt{2}/_2$, so that the probability of getting a 0 or a 1 during measurement is the same.

Once the initialization of the population has been completed, the entire quantum population is measured, using the Copenhagen interpretation, with each chromosome considered a single binary string, and each string evaluated with respect to the knapsack problem. The string which produces the highest profit while remaining under the constricting value is stored as best individual in that generation. The fitness of the best population is then plotted on a curve for future use. In order to create a new generation from the existing quantum population, each quantum gene, represented by a qubit vector, is rotated in the search space and this rotation results in new values for alpha and beta. The probability amplitudes (alpha and beta) are plotted and the resulting will eventually start to reach a local maximum. When the probability amplitude reaches the asymptote of the curve, the evolutionary algorithm is terminated (Kucharski & Nowotniak, 2014). Thus, the theory of the genetic algorithm remains the same but a QIGA manipulates qubit vectors, instead of classical values, to create increase optimization.

IV. Use of Disaster Algorithms in Tuning of the QIGA-1

In recent years, the standard QIGA has been used as a foundation for newer algorithms that improve optimization and efficiency. A major research breakthrough in this field decreased the time of convergence in later stages of the evolutionary algorithm, and improved local search performance by implementing a self-adaptive rotating angle strategy and a disaster algorithm (Fu et al., 2013). The rotating strategy exploited different concepts of linear algebra to decrease the number of parents with each call of the algorithm. The disaster condition, however, imposed a self-made disaster once the genetic algorithm reached a premature convergence value. For

example, in some functions, there are various relative maxima and the genetic algorithm might mistake a relative maximum for an absolute maximum. In a quantum system, the disaster was usually resetting the probability amplitudes of some vectors to a linear superposition of states.

Although these researchers, from the College of Field Engineering in Nanjing, China, did not test their algorithm using the knapsack problem, their method required similar computational abilities. After testing, researchers concluded that the addition of the rotating technique and the disaster condition improved the times of convergence of the quantum-inspired genetic algorithm, and returned a higher optimization value (Fu et al., 2013). However, the research that was performed by this team used isolated qubit vectors to encode qubit strings, as common with QIGA 1 algorithms, and the simplicity of the design could have caused a lower optimization value than initially expected. In addition to that, the evolutionary algorithm was called a maximum of 200 times after the manipulation of the amplitudes and that is too little to reach an optimum condition.

V. Higher-Order Quantum-Inspired Genetic Algorithms for Optimization

Previously, the QIGA had countless flaws, including the use of single qubit vectors for optimization and the minimal calls of the function. Researchers from Poland fixed many of these issues by developing a higher-order quantum-inspired genetic algorithm, that uses quantum registers to handle the encoding instead of isolated qubit vectors (Kucharski & Nowotniak, 2014). A quantum register is essentially an operation that transposes the vectors of two different qubits onto each other. The benefit of using quantum registers instead of single qubits is that error in the algorithm is drastically minimized. QIGA algorithms with Order-*r* can give a more accurate representation of a solution to the knapsack problem since the ability to model relations between separate genes increases (Kucharski & Nowotniak, 2014).

Apart from the use of a quantum register instead of isolated qubits, the QIGA-2 also has other features that help streamline the optimization process. After the initialization of the population, essentially the same process as the QIGA-1 except the genes are grouped into 2-bit quantum registers, the population is evaluated using a couple of methods. The measurement function returns one of the binary strings 00, 01, 10 and 11 with a probability of $|\alpha_0|^2, |\alpha_1|^2, |\alpha_3|^2, |\alpha_4|^2$ respectively (Kucharski & Nowotniak, 2014). Simply put, instead of measuring each individual vector, as in most quantum algorithms, the QIGA-2 assigns a two-bit value, from 0 – 3 in binary, to each register in the chromosome. The creation of a new generation is also handled differently. If the probability amplitude, of any of the four coefficient values in the quantum register, does not correspond to a specific pair of bits currently in the best individual, the probability amplitude will be decreased (amplitude contraction) according to the rule: $\alpha_{new} = \mu * \alpha_{old}$ where $\mu \in \{0,1\}$ and μ is a parameter of the algorithm (Kucharski & Nowotniak, 2014). Once the manipulation is completed, the transposed vectors in each register end up rotated and the rotation value is noted.

After testing the new algorithm with different mathematical problems, including the knapsack problem, the polish team concluded that the stabilization, or optimization, value for their algorithm was significantly higher than previous genetic algorithms proposed to solve similar problems (Kucharski & Nowotniak, 2014). However, this algorithm still has countless flaws. The current setup of the code does not account for the stabilization that occurs when an environment reaches a local optimum. A local optimum is when the genetic algorithm prematurely converges onto a single value for a certain period of time giving the illusion that the absolute minima, or extremity point, has been determined (Neves & Miguel, 1999).

VI. Parallelization and Meta-Optimization of the QIGA-2

After realizing the flaws of the previous algorithms, the researchers decided to tune the QIGA-2 by optimizing hyper-parameters, such as rotation value and contraction value. The first approach to increasing optimization is parallelization of the experimentation procedure. Parallelization of the experiment takes advantage of a CUDA architecture, which is present in modern GPU units, where processing threads are grouped in blocks, and blocks are arranged in a two-dimensional grid. Specifically, two separate parallelization algorithms have been created by the researchers. in a block of threads, each thread transforms a separate individual or different gene; the other approach was that in each block, a separate experiment with a different population is conducted (Kucharski & Nowotniak, 2012). However, for the purposes of the current experiment, a new population will be run on each block in the CUDA architecture and the second method of parallelization will be employed.

While the genetic algorithm is running on multiple different populations at the same time, rotation values and contraction values along with the final profit for each population is recorded. A process called meta-optimization is the performed and meta-optimization has two possible definitions: the fitness value after 5000 fitness evaluations must be maximized or the number of evaluations to reach a certain fitness value must be minimized (Kucharski & Nowotniak, 2012). The easiest way to meta-optimize a genetic algorithm is to tune the hyper-parameters for the algorithm. In application, Nowotniak and Kucharski complied basic summary statistics for each parameter and manipulated a couple of the hyper-parameters based upon these statistics. For example, if a rotation value has a low standard deviation across the multiple populations, the rotation can be set as constant and eliminate the need for multiple calculations. However, the meta-optimization process used has many flaws including the simplicity of the algorithm. By using summary statistics to manipulate the hyper-parameters, certain relationships between the

rotation values are not considered during the process. Instead, a complex method of optimizing these parameters should be created.

VII. Nelder-Mead Method for Hyper-Parameter Optimization

Although this method for optimizing the hyper-parameters of the QIGA-2 is somewhat successful, there are other machine learning methods that can be used. However, one important factor to take into consideration is the complexity of the machine learning algorithm. If the hyper-parameter optimization algorithm is too complex, it will have its own hyper-parameters that will affect the accuracy of the model. On the other hand, if the algorithm is too simple, the model will not be able to accurately optimize the hyper-parameters using the data collected. Researchers from IPSJ Transactions on Computer Vision and Applications have determined that the Nelder-Mead method is an effect way to optimize the hyper-parameters for both a deep neural network, DNN, and a support vector machine, SVM (Ozaki, Onishi & Yano, 2017). The In theory, the Nelder-Mead method is a downhill simplex method, or amoeba method, that is commonly applied to find the minimum or maximum of an objective function in multidimensional space (Nelder & Mead, 1965). Since the Nelder-Mead is a simplex algorithm, it can be applied to nonlinear optimization problems for which derivatives may not be known. Therefore, this method has a perfect application in hyper-parameter optimization. As the researchers above discovered at the end of their experiment, out of all hyper-parameter optimization methods, the Nelder-Mead method had the most success, allowing the DNN to classify 87.20% of the objects. (Ozaki, Onishi & Yano, 2017).

However, the researchers still used the Nelder-Mead method to optimize the hyper-parameters of a classical machine learning algorithm. But, if simplex algorithms, such as the Nelder-Mead, have had great success in optimizing the hyper-parameters for deep neural

network, the method could be useful when applied to the QIGA-2. In fact, the hyper-parameters for both of the algorithm have similar levels of complexity; therefore, the Nelder-Mead method could successfully optimize the vector rotation angles after 100 generations in the QIGA-2.

VIII. Different Libraries for Implementation

Although quantum computing in general is a fairly new field of programming and data analysis, there are a couple of libraries that can be used to run the QIGA-2. The first library, PyEvolve, is the quantum evolutionary library that was used by a majority of the researchers in this field. Using PyEvolve, most parts of a genetic algorithm can be taken into account including variation and parent selection mechanism. In this experiment, the PyEvolve library will be used as more of a control and used for comparisons with other research in this field. However, a new quantum computing library specialized for optimization that was released by IBM recently is Qiskit. Qiskit is an open source framework for quantum computing which accessible for people with different background including quantum researchers, teachers and developers. However, a major difference between PyEvolve and Qiskit is that researchers can run the algorithm on an IBM two-qubit quantum computer instead of on a classical computer. This, in theory, should greatly increase the speed of optimization for problems such as the knapsack. Another important feature of Qiskit is the visualization abilities of the library. Since IBM wants to make the software understandable to a general audience, the API is filled with ways to graphically represent the results from the algorithm and this ability makes the results easier to understand or interpret for the researchers.

IX. New Research

The new research performed addresses the flaws in the previous research and allows for greater theoretical optimization of the knapsack problem. In the new algorithm, a disaster

condition and quantum mutation will be added to the higher-order QIGA to prevent premature convergence. In fact, socials disasters technique, which applies a catastrophic operator once a local minimum is reached is one of the best methods to obtain the global extremity point instead of the local extrema (Neves & Miguel, 1999). In the algorithm that will be created, the quantum mutation will occur when the probability amplitudes of different qubits in the register fluctuate, and the disaster condition will implement a simple quantum catastrophe operator. Therefore, by tuning the higher-ordered QIGA using a disaster algorithm and a quantum mutation, the probability of reaching a global maximum will increase drastically, and the optimization value for the knapsack problem will be much higher than previous algorithms that have been designed. However, apart from that, the majority of the project will be focused on using the Nelder-Mead method to optimize the hyper-parameters of the QIGA-2. The optimization of hyper-parameters will increase the efficiency of the algorithm and result in a higher profit output. Researching this topic will allow the scientific and engineering community to accomplish countless tasks that have a limitation factor and an optimization value.

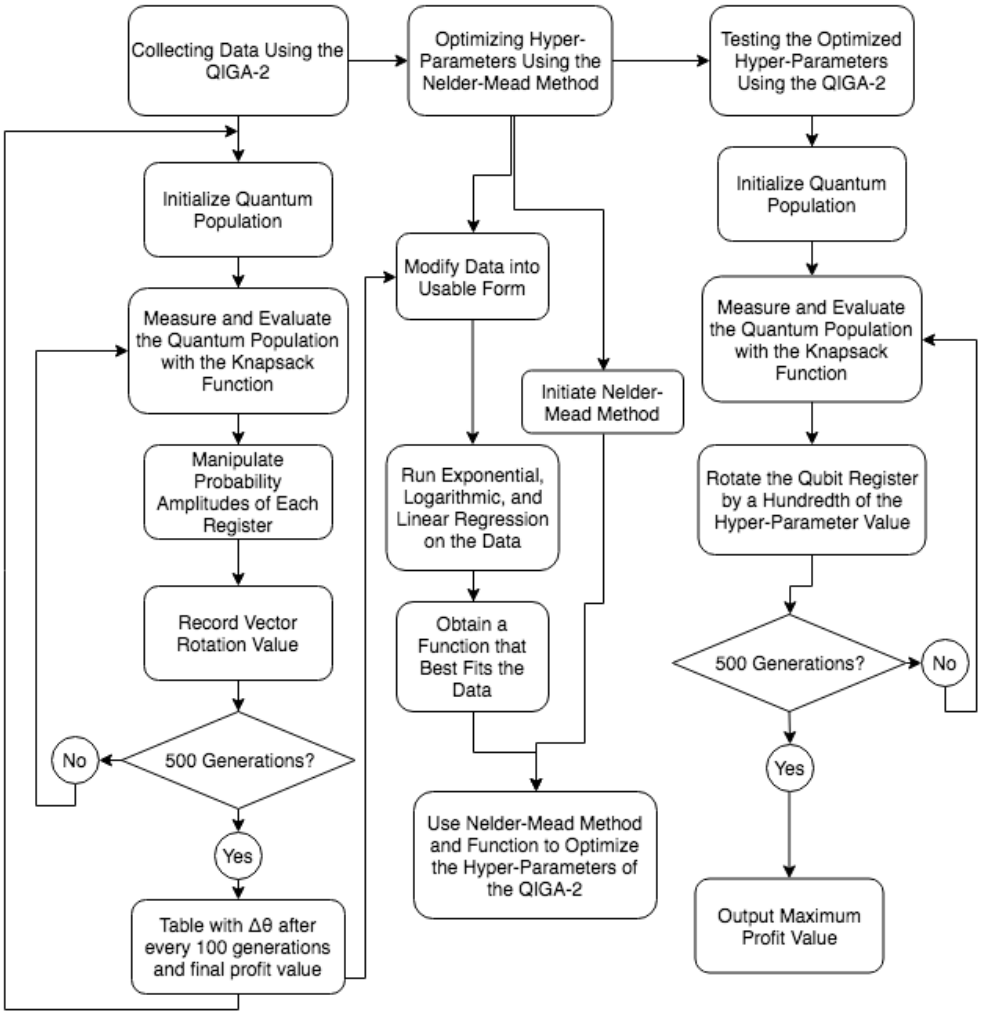
Experimental Design

In this research problem, the extent to which the Nelder-Mead method can be used to optimize the hyper-parameters of a QIGA-2 during the Knapsack Problem is investigated. If a Nelder-Mead method is used for the optimization process, then the QIGA-2 output, or the optimum profit for the Knapsack Problem, will be maximized. This method will work since the Nelder-Mead method is a gradient-free method of optimization, and does not have additional hyper parameters. Also, the Nelder-Mead is a simplex amoeba algorithm which results in minimal error between the final output and the expected value. Therefore, using this method, the hyper-parameters of the QIGA-2 can be easily manipulated to achieve the optimum values. The

independent variable in this experiment is the hyper-parameters, or rotation values, of the higher order quantum-inspired genetic algorithm. The dependent variable in this experiment is the final output of the QIGA-2 after the hyper-parameters have been tuned. The dependent variable will be measured using the highest profit possible under the restrictions of the Knapsack Problem. The constants that are present in this experiment include the method used for hyper-parameter optimization (Nelder-Mead), the method for which the hyper-parameters are being optimized (QIGA-2), the problem used for testing (knapsack problem), the hardware used to execute the entire algorithm, the dependencies and libraries installed on the hardware, the rate at which the program is run, and the number of QIGA-2 instances that will be created

The control in this experiment is final profit of the Knapsack returned by the QIGA-2 before the Nelder-Mead method has been employed to tune the hyper-parameters. This will allow the researchers to determine whether the optimization of the hyper-parameters increased or decreased the final profit of the Knapsack. In the experiment, there will be many repeated trials to reduce the amount of error present in the results. During each phase of the process, training and testing, 30 different quantum populations are initialized using the parallelization process mentioned in the procedure. Essentially, the QIGA-2 is simultaneously optimizing the knapsack problem for each of these populations and this creates in a total of 30 repeated trials during each phase.

A procedural summary for this project is as follows:



Materials and Method

Research will be completed using a three-phase process. In phase 1, training data for the Nelder-Mead method will be created using a Higher-Order Quantum-Inspired genetic algorithm**.** However, before using the quantum computing program to create a set of training data for hyper-parameter optimization, various important libraries including Qiskit by IBM and Numpy must be downloaded and imported. Then, the Quantum population will be created using an array of two-qubit registers and the population will be initialized with a linear superposition of states. The last

aspect of initialization is the definition of the knapsack function based upon the requirements of the knapsack problem presented.

After initialization, the genetic algorithm begins. The quantum population will be classically measured by collapsing the existing wave function Each classical string obtained as a result of quantum measurement will be evaluated in the knapsack function. After the evaluation of the population, the algorithm will determine the classical string with the highest value and store the string as b. Now that the value for b has been determined, the probability amplitudes of each qubit register will be manipulated in the following manner. Initially, the first binary pair in the binary string, b, will be obtained and the probability amplitude of alpha and beta for both qubits in the first register will be measured. Then, each probability amplitudes will be compared to the binary pair. If the probability amplitude does not equal the binary pair, the amplitude will be decreased by a factor of μ (approximately 0.99). At the end of the manipulation process, the vector of the manipulated qubit register will be determined and the rotation angle between this vector and the vector of the original qubit register will be recorded. Moreover, this process is repeated with each pair of binary numbers in b. If done correctly, each binary pair in the string will be associated with a qubit register in the population.

At this point, the algorithm has completed one generation of fitness evaluations. In order to obtain an accurate optimization of the knapsack problem, this process needs to be repeated for 500 generations, or until the maximum profit on the curve has stabilized. After one instance of the algorithm has completed its runtime, the data of vector rotation values between every 100 generations and the final profit outputted by the QIGA-2 will be assembled. However, the first phase does not end at this point. Using parallelization features available in most NVidia GPU'S with CUDA structures, multiple quantum populations will be running at the same time. At the

conclusion of this phase, a data set for 5 rotation values, hyper-parameters, and a single profit value will be compiled for each population running.

In phase 2 of the process, the hyper-parameters of this QIGA-2 algorithm will be optimized using a Nelder-Mead method. However, a unique feature about the Nelder-Mead method is that it requires a function to either minimize or maximize. In order to obtain that function, the data set will be manipulated so that it is in the correct format for the WEKA software. After modification of the data, WEKA will be used to perform regression analysis, experimenting with different models including linear, logistic and exponential. At the end of this process, a function will be obtained that best fits the data collected after running the QIGA-2 algorithm. Now, the optimization of the hyper-parameters begins. The GitHub Nelder-Mead library created by the researchers from IPSJ Transactions on Computer Vision and Applications will be downloaded and imported into the program. Using this library, the Nelder-Mead method will be initialized by defining certain domain restrictions for the method as well as defining the function obtained through regression analysis. After the initialization is complete, the Nelder-Mead method will be run and the hyper-parameters for which the profit outputted by the QIGA-2 should be the greatest will be obtained.

In phase 3 of the process, the optimized hyper-parameters will be tested by using the same higher-order quantum-inspired genetic algorithm. The procedure for this phase of the project is extremely similar to the first phase of the project. However, instead of manipulating the probability amplitudes in order to change the quantum register, the optimized hyper-parameters will be used. After evaluation and determination of the best classical string, b, the first quantum register will be accessed and the qubit vector of that register will be determined. Then, the vector will be rotated by a hundredth of the n^{th} hyper-parameter value obtained through the optimization

method employed. For example, if the QIGA-2 is between 0 and 100 generations, the first hyper-parameter value will be used. If the algorithm is between 101 and 200 the second hyper-parameter value will be used. The process continues, switching hyper-parameter values every 100 generations, until 500 generations have been completed or until the profit curve has stabilized. Once this occurs, the maximum profit will be returned and compared to the maximum profit before optimization of hyper-parameters. Currently, the statistics to determine whether the research was a success is not known and the researchers are still exploring different methods of validation.

References

Bossaerts, P., Murawski, C. (2016). How Humans Solve Complex Problems: The Case of the Knapsack Problem. *Nature Partner Journals*. doi: 10.1038/srep34851

Forrester, R., (2017, Jan 20). The Quantum Measurement Problem: Collapse of the Wave Function Explained. Social Science Research Network. http://dx.doi.org/10.2139/ssrn.2901820

Fu, C., Liu, J., Wang, H., & Zhi J. (2013, Mar 9). The Improvement of Quantum Genetic Algorithm and Its Application on Function Optimization. Hindawi, 2013. dx.doi.org/10.1155/2013/730749

Garg A., Goyal S. (2017). Vector Sparse Representation of Color Image Using Quaternion Matrix Analysis Based on Genetic Algorithm. *Imperial Journal of Interdisciplinary Research, 3(7).* www.imperialjournals.com/index.php/IJIR/article/view/5429

Kucharski, J., Nowotniak, R. (2012). GPU-based tuning of Quantum-Inspired Genetic Algorithm for a Combinatorial Optimization Problem. *Bulletin of the Polish Academy of Science, 3(7).* doi: 10.2478/v10175-012-0043-4

Kucharski, J., Nowotniak, R. (2014, Sept). Higher-Order Quantum-Inspired Genetic Algorithms. IEEE. doi: 10.15439/2014F99

Mead R., Nelder J. (1965, January 1). A Simplex Method for Function Minimization. The Computer Journal, 7(4), 27. https://doi.org/10.1093/comjnl/8.1.27

Montanaro, A. (2016). Quantum Algorithms: An Overview. *Nature Partner Journals.* doi: 10.1038/npjqi.2015.23

Neves, J., Miguel, R. (1999). Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation. Lecture Notes in Computer Science, 1611, 127-136. https://doi.org/10.1007/978-3-540-48765-4_16

Onishi, M., Ozaki, Y., & Yano, M. (2017). Effective hyperparameter optimization using Nelder-Mead Method in deep learning. *IPSJ Transactions on Computer Vision and Applications. 9(20).* doi: 10.1186/s41074-017-0030-7