Simulation of implementable quantum-assisted genetic algorithm

Jirayu Supasil\textsuperscript{1}, Poramet Pathumsoot and Sujin Suwanna\textsuperscript{2}

Optical and Quantum Physics Laboratory, Department of Physics, Faculty of Science, Mahidol University, Bangkok, 10400, Thailand

E-mail: \textsuperscript{1}jirayu.sup@student.mahidol.ac.th; \textsuperscript{2}sujin.suw@mahidol.ac.th

Abstract. Quantum-assisted algorithms are expected to improve the computing performance of classical computers. A quantum genetic algorithm utilizes the advantages of quantum computation by combining the truncation selection in a classical genetic algorithm with the quantum Grover’s algorithm. The parallelism of evaluation can create global search and reduce the need of crossover and mutation in a conventional genetic algorithm. In this work, we aim to demonstrate and simulate the performance of an implementable quantum-assisted genetic algorithm. The algorithm was tested by using quadratic unconstrained binary optimization (QUBO) for 100 iterations; and the results were compared with those from a classical counterpart for 2000 iterations, where both simulations were performed over 100 repetitions. The results showed that the quantum algorithm converges to the optimal solution faster. While the variance is higher at early stage, it quickly and greatly reduces as the algorithm converges. The histograms of possible solutions consistently exhibits this behavior.

1. Introduction
Quantum technology has shown great potential to improve performance and efficiency of computation. Based on principles of quantum mechanics, quantum computation can expand the capability of computing and information processing. Studies have shown that quantum algorithms can possess advantages of computation over classical ones [1–6]. A large-scale quantum computer is purposed to solve problems not possible by a classical supercomputer. Nowadays, quantum computing development and finding problems to demonstrate quantum advantages have tremendously expedited this exciting research field [7–10].

Optimization is a common and important problem whose main purpose is to find a minimum or maximum point from a predetermined set, called a search space. This problem practically occurs in many applications, for instances, in sciences, engineering and finance [11–13]. An algorithm will be efficient if it uses less resource such as time and memory; and the algorithm with lower complexity will use less resource. Thus, the efficiency, or complexity, of an algorithm for solving a given problem is usually at the heart of algorithm analysis and deployment [14].

A genetic algorithm is an optimization algorithm which was inspired by the concept of natural selection to find the survival of fitness [15]. This algorithm finds an optimal point by searching and selecting good solutions from samples and uses mutations and crossovers to increase the variation of samples in each generation. In subsequent generations, the samples will have higher tendency to be good solutions, so the algorithm has greater probability to escape local trapping minimum points than other algorithms, such as a gradient descent method [16], to converge to
a global optimum. Whilst a quantum computer is not yet deployed in practice, there have been many algorithms developed to combine the principles of quantum computation with classical algorithms. This is evident from recent research to create a quantum-assisted genetic algorithm using reverse quantum annealing as a mutation operator [17]. It is also possible to implement a genetic algorithm on a circuit-based quantum computing platform and improve its performance. One possible approach is to combine the well-known quantum Grover’s search algorithm and a truncation selection process in a classical genetic algorithm [18]. The former has been proved to be more efficient than its classical counterpart, where quantum parallelism can lower the algorithm’s complexity [18–20]. In addition, the Grover’s search algorithm can be employed to sample the entire population, which would give a quantum-assisted genetic algorithm for global search without using crossovers and mutations, also known as a reduced quantum genetic algorithm [20]. In this case, it is known that the algorithm’s complexity has order $O(\sqrt{N})$, which is polynomial speedup from a classical genetic algorithm [18–20].

Combinatorial optimization is an NP-hard discrete problem which combines various binary parameters to form an objective function. The direction of objective function is generally complicate and hard to solve. However, this problem appears in various applications. One of the mathematical formulation of this optimization is quadratic unconstrained binary optimization (QUBO) whose objective function is formed by a quadratic equation of binary variables [21]. Recently, a well-known nurse-scheduling problem, which has been challenging to classical computers over the years, can be formulated into QUBO and promisingly solved by quantum annealing [22].

In this study, we aim to demonstrate a quantum-assisted genetic algorithm by using QUBO as a test problem. The quantum and classical algorithms will be run with specific iterations, and equal number of evaluation. Then the results from both algorithms will be compared in terms of convergence, variance and histogram. Our proposed algorithm is implementable on a quantum simulator such as the quantum assembly (QASM) simulator, and it is aimed to improve the speed of convergence (or performance) of a genetic algorithm. To demonstrate such advantage, we used Qiskit to implement the proposed quantum-assisted genetic algorithm and simulate a noiseless quantum device. Qiskit [23] is an open source python library for quantum computing software development framework whose QASM simulator is capable of simulating ideal and real noisy quantum devices such as IBM Q quantum processors. The results will be used to study a situation when noise is applied in the future before the implementation on a real device.

2. Methodology

In this work, a quantum-assisted genetic algorithms is based on a truncation selection of a classical genetic algorithm, where this selection method will sort the samples, and the fraction $T$ of best individual samples can be selected with the same probabilities [18,20]. For a classical genetic algorithm, crossovers and mutations are required in order to create variation of samples and to expand a search space [15]. However, in a quantum version, we can employ the Grover’s search algorithm for multiple solutions to select the fraction $T$ of best solutions from the entire population [20]. By creating a superposition of all possible states, the quantum algorithm can search through the global population simultaneously. Thus, the quantum genetic algorithm can operate without crossovers or mutations.

The Grover’s search algorithm performs searching from unsorted data. In order to find good solutions, we create a threshold and allow this algorithm to search for solutions which give objective function below the threshold (if the problem is finding minimum). This algorithm requires approximately $(\pi/4)\sqrt{N}/t$ oracle calls as a proper number in order to find one of $t$ good solutions from the population $N$ with the highest probability [19]. In this setup, the number of good solutions $t$ is unknown; thus, it is difficult to find the proper number of oracle calls. Previous researches reasoned that the unknown solution could be found in an expected time of
3

$O(\sqrt{N}/t)$ by using one-shot Grover’s algorithm with BBHT algorithm [18, 19]. However, in the era of noisy intermediate scale quantum (NISQ) devices, the result from a quantum device is not accurate [24, 25], and one-shot results can be misled by noises in the system. Thus, we need a strategy for a multi-shots Grover’s algorithm to create samples for a quantum-assisted genetic algorithm to reduce the noise effects by using statistical results. Using an unknown number of oracle calls will not give the highest probability of finding a solution, and some numbers may give higher probability of finding a wrong solution. However, while the number of oracle calls can be less than an actual one, the probability of selecting good solutions is still higher than others, and the probability will only increase when the oracle is called more until it reaches the actual proper number. Thus, our strategy will initialize the number of oracle calls equal to 1. When the mean objective function of samples is less than the threshold value, which means that the probability is not highest, we will increase the number of oracle calls in the next iteration. At the end of each iteration, if the minimum of the objective function of the selected samples is lower than threshold (in case of finding minimum), the minimum will be selected as a new threshold to search for even better solutions until the algorithm find the optimal solution.

For the strategy of updating the number of oracle calls, if $T$ denotes the fraction of good solutions on which the objective function yields value below a given threshold, the proper number of oracle calls will be $(\pi/4)\sqrt{1/T}$. After one iteration, if the new threshold (i.e. a minimum of the objective function of randomized samples from solutions which already yield the objective function value below the old threshold) still has the same fraction (that is, we obtain the fraction of a fraction), the proper number of oracle calls will be $(\pi/4)\sqrt{1/T^2}$. After $k$ iterations, if the fraction between the new selected solutions, and the old solutions is the same, the number of oracle calls is expected to be $(\pi/4)\sqrt{1/T^k}$. This implies that if the fraction of the selected solutions in the next iteration and the current selected solutions is constant, the proper number of oracle calls should grow exponentially. Thus, in our strategy, the update of the oracle calls should be updated with exponential growth. If the initial threshold is randomly selected, the probability of a selected solution will follow the uniform distribution where the expected selected threshold is half of all population or at $T = 0.5$. After operating the Grover’s search algorithm to find solutions whose objective function values are above the threshold, the probabilities of new samples are in the uniform distribution with the expected new threshold in the middle of all selected samples from initialization which also give the same fraction $T = 0.5$. At any iteration, the probability of selected samples still has uniform distribution; thus, the expected fraction between new selected solutions and the former ones still remains at $T = 0.5$. After $k$ iterations, the proper number of oracle calls is expected to be $(\pi/4)\sqrt{2^k}$. If the number of oracle calls is above the proper number, the probability of selecting a good solution will decrease; thus, we have to avoid it by using the growth rate of oracle calls less than $\sqrt{2}$.

2.1. Simulation procedure

From our strategy, a quantum-assisted genetic algorithm can be executed as the following pseudo-code. We choose the growth rate of oracle calls, $\lambda = 6/5$ which indeed can be any number between 1 and $\sqrt{2}$. Let $m$ denote the expected number of oracle calls in each iteration.

(i) Random $K$ samples from population of $N$ samples; and initialize $m = 1$.

(ii) Evaluate the given objective function of $K$ samples, calculate the mean value of the objective function and select the minimum value of the objective function as a threshold.

(iii) Calculate a maximum integer $j$ less than or equal to $m$, and apply $j$ oracle calls of the Grover’s algorithm to search for solutions which produce the objective function value below the threshold. Run algorithm for $K$ shots as new $K$ samples.

(iv) If the new samples produces the mean value of the objective function above the threshold, then set $m$ to $\min\{\lambda m, (\pi/4)\sqrt{N}\}$
Figure 1. Flow charts of a classical genetic algorithm (a), and a quantum-assisted genetic algorithm (b) in this work. The truncation selection which is a main process of both algorithms will select the first fraction $T$ of the samples which give small value of a given objective function (in case finding minimum). In a classical genetic algorithm, the objective function will sort from minimum to maximum, and the samples at order $T \times K$ will be selected as a threshold. The samples which have objective-function values below the threshold will be selected with the same weight. In a quantum algorithm, the algorithm already search from the entire population. Thus, the minimum of samples will be updated as the threshold.

(v) If the minimum of the objective function of new samples decreases, then select and update the minimum as a threshold.

(vi) Repeat steps (ii)-(iv) until the stopping criteria (stated below) are reached.

Similar to a classical genetic algorithm, the quantum-assisted algorithm will run repeatedly until it reaches the stopping criteria. In previous studies, a classical genetic algorithm has various criteria to terminate the process, such as fixing a number of iterations, fixing a number of function evaluations or chance of the change in next iteration is too low [26]. In our demonstration, we will fix the number of iterations at 100; and the results will be recorded in every 5 iterations.
2.2. Choice of QUBO

A general formulation of QUBO is

$$f(x) = \sum_{i,j=0}^{N} A_{ij} x_i x_j,$$

(1)

where the coefficients $A_{ij}$ define a symmetric matrix. Any quadratic form satisfying equation (1) would constitute a QUBO [21]. In this work, we have chosen a problem for 10 bits, so it has 1024 samples in the population with coefficients $A_{ij}$ defining a matrix $A$ as

$$A = \begin{bmatrix}
-3 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & -5 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & -9 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & -8 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -3 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & -2 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & -4 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & -5 & 1 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & -5
\end{bmatrix}$$

(2)

Several choices of QUBO with single solutions have been tested with our proposed algorithm, but in this article we present our results for the QUBO defined by the matrix in equation (2).

3. Results and discussion

We have implemented a quantum-assisted genetic algorithm for solving the specified QUBO and compared the results of simulations with the truncation-method classical genetic algorithm. The classical algorithm is set to have the crossover rate of 0.5 and mutation rate of 0.001. Both algorithms are run at 20 samples per iteration. The fraction of the classical truncation selection is fixed at $T = 0.5$, or median of samples. We simulated both algorithms with 100 repetitions, where each repetition constitutes 100 iterations for the quantum-assisted algorithm, and 2000 iterations for the classical algorithm. In the classical genetic algorithm, the number of evaluation per iteration is equal to the number of samples per iteration, but in the quantum-assisted genetic algorithm, each iteration has a specific number of oracle calls which can be considered the number of evaluation. Thus, with the same number of samples, the number of evaluation in the quantum case is larger than that in the classical case. When the number of oracle calls reaches the proper number (which is 25), the number of function evaluations per iteration of the quantum-assisted genetic algorithm will be 25 times larger than that of its classical counterpart at the same iteration. However, since the quantum-assisted genetic algorithm can search for global in each iteration without crossovers and mutations, while the classical genetic algorithm requires some number of iterations to increase the search capability, the quantum algorithm will require less iterations to converge the samples.

3.1. Compare mean of objective function

Figure 2a shows that the quantum-assisted genetic algorithm converges to the global minimum faster than its classical counterpart. In early iterations, the classical genetic algorithm may have selected some local optimization which possibly decrease the mean of the objective function. However, the quantum-assisted genetic algorithm will select all samples which yield the objective function value below threshold uniformly. If the number of oracle calls is not the proper number yet, the probability of randomly selecting samples which yield the objective function value
Figure 2. Comparison of results from quantum-assisted (pink) and classical (blue) genetic algorithms: (a) mean of the objective function (minimum at -25), and (b) variance of the objective function. The respective shaded regions in (a) indicate the variances from the mean of the solutions. Similarly, the shaded regions in (b) are bounded by the minimum and maximum variances among the iterations for that repetition.

above threshold will be higher in early iterations. After some iterations, the quantum-assisted algorithm will converge to the global minimum faster. We believe the lower convergence rate of the classical algorithm is due to the samples converging to a local minimum which required mutations to increase the sample variation.

Figure 2b shows that even though the mean of objective function reduces and converges to a minimum, the variance increases due to the number of oracle calls is less than the proper number, and the probability of selecting samples which yield the objective function value below the threshold is still high. However, when the samples start to converge to those corresponding to the minima, the variance obviously decreases in both algorithms. It is possible that the threshold already reaches the minimum, and the ratio of minimum samples to random samples starts to increase. Finally, when the objective function converges to minimum, the variance is reduced to near zero. Notably, even after the samples converge, there were some iterations where the samples have high variance. It is possible that the highest probability of finding a solution from the proper number of oracle calls is not 1. Thus, there is low probability that samples which yield very high value of the objective function will be selected at some iterations after the convergence.

3.2. Distribution of solutions

Figure 3 depicts how the solutions are distributed from initialization in figure 3a, which all possible binary strings are completely random, to after 20 iterations in figure 3d. The result after 10 iterations in figure 3b shows that the classical genetic algorithm has higher chance to reach the solution in the samples, and also other local minima with lower chance. After 15 iterations, as shown in figure 3c, the probability of selecting the minimum solution by the quantum-assisted algorithm is higher, while the classical one has more probability of selecting other local minima. Finally, after 20 iterations, in figure 3d, the probability of selecting a minimum from the quantum-assisted algorithm is closed to 1, significantly outpacing the classical counterpart.

Further examination suggests that the threshold at 5 iterations has already reached the minimum. This means that our proposed algorithm can update the number of oracle calls with
Figure 3. Probability distributions of solutions the classical (blue) and quantum-assisted (pink) algorithms as the number of iterations increases: (a) initialization, (b) after 10 iterations, (c) after 15 iterations, and (d) after 20 iterations. In all figures, the bit strings of solutions are represented by numbers in the decimal system; and the vertical dash line indicates the solution (i.e. global minimum).

The growth rate faster, which can reduce the number of evaluations to convergence.

This study shows that, in this exemplified problem, the quantum-assisted genetic algorithm yields advantages over its classical counterpart in long run. However, if the algorithm is implemented on a NISQ device, the results may be far from ideal in reaching the converged solution and probability distribution. Thus, the simulation of a quantum-assisted genetic algorithm on a noisy device can be potentially investigated as our proposed algorithm already consists of the implementable circuit on Qiskit’s QASM. The proposed algorithm is designed to search for a single solution in an optimization problem, so it is necessary to improve the algorithm to search for multiple solutions, which can then be applied to a more general problem including a QUBO mentioned in Ref. [22].

4. Conclusion
We have simulated an implementable quantum-assisted genetic algorithm by combining the truncation selection for a classical genetic algorithm and the quantum Grover’s search algorithm. The proposed algorithm was simulated by solving quadratic unconstrained binary optimization, and its performance was compared with its classical counterpart. The results showed that, in a
long run, the quantum-assisted genetic algorithm can converge faster than the classical genetic algorithm, and the probability of finding global optimization is greater. It also has less chance of getting trapped by a local minimum. This simulated result is a good demonstration of an algorithm implementable on a real device in the future.

Acknowledgments
JS and SS thank Dr. Jirawat Tangpanitanon and Dr. Unchalisa Taetragool for suggestions of ideas and fruitful discussions. All authors acknowledge partial research funding from Quantum Technology Endeavors (QuTE) Thailand. This research was also supported by the Program Management Unit for Human Resources & Institutional Development, Research and Innovation (grant number B05F630108), Thailand.

References
[1] Grover L K 1996 28th Annual ACM Symposium on Theory of Computing STOC ’96 (New York: Association for Computing Machinery) p 212–9
[2] Shor P W 1997 SIAM J. Comput. 26 1484–1509
[3] Nielsen M A and Chuang I L 2011 Quantum Computation and Quantum Information: 10th Anniversary Edition (New York: Cambridge University Press)
[4] Haah J, Hastings M, Kothari R and Low G H 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS) pp 350–60
[5] Wang G 2017 Phys. Rev. A 96(1) 012335
[6] Srinivasan K, Satyajit S, Behera B K and Panigrahi P K 2018 e-Print arXiv:1805.10928
[7] Arute F et al 2019 Nature 574 505–10
[8] Orúes R, Mugel S and Lizaso E 2019 Rev. Phys. 4 100028
[9] Venturelli D and Kondratyev A 2019 Quantum Mach. Intel. 1 17–30
[10] Beer K, Bondarenko D, Farrelly T, Osborne T J, Salzmann R, Schiermann D and Wolf R 2020 Nat. Commun. 11 808
[11] Haggag S, Desokey F and Ramadan M 2017 Gravit. Cosmol. 23 236–9
[12] Tu S, Cheng Q S, Zhang Y, Bandler J W and Nikolova N K 2013 IEEE Trans. Antennas Propag. 61 3797–807
[13] Rotemberg J J and Woodford M 1997 NBER Macroecon. Ann. 12 297–346
[14] Levin A L 2020 e-Print arXiv:2005.06436
[15] Goldberg D E 1989 Genetic Algorithms in Search, Optimization, and Machine Learning (New York: Addison-Wesley Professional)
[16] Blum C and Roli A 2001 ACM Comput. Surv. 35 268–308
[17] King J, Mohseni M, Bernoudy W, Fréchette A, Sadeghi H, Isakov S V, Neven H and Amin M H 2019 e-Print arXiv:1907.00707
[18] Malossini A, Blanzieri E and Calarco T 2008 IEEE Trans. Evol. Comput. 12
[19] Boyer M, Brassard G, Hayer P and Tapp A 1998 Fortschritte der Phys. 46 493–505
[20] Udrescu M, Prodan L and Vladutiu M 2006 3th Conference on Computing Frontiers (ACM) pp 71–82
[21] Glover F, Kochenberger G and Du Y 2018 e-Print arXiv:1811.11538
[22] Ikeda K, Nakamura Y and Humble T S 2019 Sci. Rep 19 12837
[23] Aleksandrowicz G et al 2019 Qiskit: An Open-source Framework for Quantum Computing (software library) online access: January, 2020 (https://zenodo.org/record/2562111##.x2ga5jnhly)
[24] Preskill J 2018 Quantum 2 79
[25] Mohammadbagherpoor H, Oh Y H, Singh A, Yu X and Rindos A J 2019 e-Print arXiv:1903.07605
[26] Safe M, Carballido J, Ponzoni I and Brignole N 2004 Lect. Notes Comput. Sci. 3171 405–13