Experiments on Quantum Circuits for Compact Genetic Algorithm

1st Kamonluk Suksen
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand
6071401321@student.chula.ac.th

2nd Naphan Benchasattabuse
Graduate School of Media and Governance
Keio University Shonan Fujisawa Campus
Kanagawa, Japan
whit3z@sfc.wide.ad.jp

3rd Prabhas Chongstitvatana
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand
prabhas.c@chula.ac.th

***Abstract*—Programming a quantum computer is still a challenge for researchers. Quantum computers operate based on a fundamentally different paradigm from classical computers. Programming a quantum computer today is somewhat like programming a classical computer in the 1950s, in a manner close to assembly language. This paper presents a Grover-assisted compact genetic algorithm to solve some real-world problems on a quantum computer. This method is a combination of Grover's search algorithm initializing with the probability distribution, and a compact genetic algorithm with elite (cGA*), which keeps the current best individual. The compact genetic algorithm (cGA) is a powerful and popular method in evolutionary computation. The results of the proposed algorithm are compared with the cGA running on classical computers in terms of the solution quality and the number of function evaluations. The compared results illustrate that the proposed methodology can successfully reach 100% accuracy of the solution with a higher number of function evaluations than the cGA.**

***Index Terms*—Quantum computing, Grover's search algorithm, compact genetic algorithm (cGA)**

I. Introduction

Quantum computation is an emerging interdisciplinary practice of science where processing of information is based on quantum mechanics. It has been shown that quantum algorithms can provide dramatic advantages over the classical ones. Two of the most well-known algorithms are the number factorization algorithm [1] (Shor's algorithm), which can be used to break the cryptographic system that we use to send credit card data in today's internet efficiently, and a general unstructured search algorithm [2] (Grover's algorithm). Although Grover's algorithm provides only polynomial speed up over classical brute force method, they can be used to speed up wide range of applications.

The task of finding the minimum or the maximum of a given function, the optimization problem, is one of the most important problem in computer science and has a wide range of applications to engineering and finances. In this work, Travelling Salesman Problem (TSP) is presented as a test problem since it is a common optimization issue seen in a variety of fields of science and engineering. The problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. One approach to tackle the optimization problem is to use evolutionary algorithm where the main mechanism that is used to search for solutions comes from biological evolution [3]. It is thought that mutation, recombination, and natural selection of genes from the population are keys to make them survive and progress through generations, survival of the fittest. The idea has been applied to optimization, by defining a *fitness function* where it output the score (or *fitness*) of a given solution candidate. If one were to encode possible solutions into a population, simulate the evolution procedure, and the entire population would evolve and producing newer generations of candidate that will converge into the best candidate with the highest fitness.

Genetic algorithms [4], [5] (GAs) are one of the most widely used type of evolutionary algorithms. They encode the input to the optimization into a structured string that which utilizes the randomization process to produce new sets of strings from bits and parts of fittest of the old. A memory efficient variant, compact genetic algorithm [6] (cGA), which instead of storing all the population strings in memory, only represents the population as a probability distribution over a set of solutions. Although cGA behaviors only mimic that of uniform cross over, order-one simple GA, its simple structure can be implemented directly onto hardware with low memory to boost its performance [7].

There have been attempts to combine quantum computing and genetic algorithms in the hope that their quantum versions would provide speed up or new ideas to the classical algorithms. Quantum-inspired genetic algorithms [8]–[10] are one of the attempt where the algorithms are still executed purely in the classical computers but took some of the ideas of quantum phenomena such as interference and superposition and translate them into classical analogue. Another approach, quantum-assisted genetic algorithms, [11]–[14] and quantum-assisted compact genetic algorithm [15] are to delegate some tasks of the algorithm like mutation operator or probabilistic elements to quantum computers while still performing crossover and population update on the classical side. The last approach is the attempts to redefine the GA in the context of quantum computation [16]–[18] by creating a population with superposition of all states, measuring the fitness to reduce the

978-1-6654-1197-4/21/$31.00 ©2021 IEEE

dimension of the space, applying crossover while parts of the population are still entangled, and re-expanding the space into a superposition of larger dimension. The process is repeated until the population converged or termination condition is met.

Most approaches to combine GA and quantum so far focus on the traditional GA where crossover, recombination, and mutation are presented and often require a huge number of qubits to operate. In this work we aim to utilize Grover's algorithm to enhance the performance of cGA as briefly outlined in the work from [15] which requires fewer number of qubits than traditional GA. We demonstrate a Grover-assisted compact genetic algorithm solving the traveling salesman problem of size 3 and 4 city. The simulations were executed using Qiskit library [19], software package for simulating quantum computers provided by IBM. The results are compared with the classical cGA in terms of solution quality and number of function evaluations.

II. Compact Genetic Algorithm

The Genetic Algorithm (GA) is a powerful optimization algorithm inspired by natural evolution [4]. The GA is performed by creating a population of solutions and produced the offspring using genetic operators. The solutions are continuously improved by a selection scheme that selects the survivors to the next generation based on their fitness values defined by users. Conversely, the cGA manipulates the probability vector instead of the actual population. This dramatically reduces the number of bits required in cGA to store the population. However, both algorithms still encounter premature convergence to local optima of the objective function. Thus, we propose the cGA storing the current best individual to ensure that it gets no worse individual at each generation. The steps of cGA are described below.

A. Compact Genetic Algorithm

The compact genetic algorithm (cGA) [6] represents the population as a probability vector over the set of solutions. The vector contains each bit with a real number from 0.0 to 1.0 represents the probability of that bit to be one. At each generation, cGA samples two individuals according to the probability value and calculates their fitness using the fitness function. Next is to determine the *winner* by comparing their fitness values. The winner's chromosome will be used to update the probability vector so that the distribution will converge towards the best fit solution. This is an iterative process until we reach a termination condition.

B. Compact Genetic Algorithm with Elite

The main concept is using the current best individual to get the second individual. The second individual is sampled from probability vector as normal but if its fitness is lower than that of the current best individual, it will be replaced with current best individual. Thus, the fitness of the second individual is always greater than or equals to the fitness of the current best observed fitness. A cGA with elite is shown as Algorithm 1. If the search space has *N* entries, then the time taken to complete a linear search is $\mathcal{O}(N)$ (on average, $N/2$). The quantum Grover's search algorithm can do better, which is completed in time $\mathcal{O}(\sqrt{N})$ [2].

Algorithm 1: The cGA with elite

1) initialize probability vector:
for $i \leftarrow 1$ *to* l **do**
  $p[i] \leftarrow 0.5$;
end
2) initialize the current best individual:
$curBestIdv \leftarrow$ 000..00;
3) generate two individuals from the vector:
$a \leftarrow generate(p)$ $b \leftarrow generate(p)$;
4) evaluate the second individual's fitness:
if $curBestIdv.fitness > b.fitness$ **then**
  $b \leftarrow curBestIdv$;
end
5) let them compete:
$winner, loser \leftarrow compete(a, b)$;
6) update probability vector towards winner:
for $i \leftarrow 1$ *to* l **do**
  if $winner[i] \neq loser[i]$ **then**
    if $winner[i] \neq i$ **then**
      $p[i] \leftarrow p[i] + 1/n$;
    else
      $p[i] \leftarrow p[i] - 1/n$;
    end
  end
end
7) update the current best individual:
if $curBestIdv.fitness < winner.fitness$ **then**
  $curBestIdv \leftarrow winner$;
end
8) check if the vector has converged:
for $i \leftarrow 1$ *to* l **do**
  if $p[i] > 0$ *and* $p[i] < 1$ **then**
    return to step 3;
  end
end
l is a chromosome length and n is a population size.

III. Grover's Search Algorithm

Grover's search algorithm can provide improvements over classical algorithms as a quadratic speedup to solve the problem of unstructured search on quantum computers [2]. The oracle and amplitude amplification stages in the algorithm help to get the target solutions with high probability. The oracle does not find the solution but simply recognizes them. The key to quantum search is that we can look at all solutions simultaneously: the oracle just manipulates the state coefficients using a unitary operator. Thus, we exploit this advantage by applying Grover's search algorithm to generate individual in order to increase the chances of observing a target solution. The first step in Grover's algorithm is that all qubits are set to be in superposition. After this operation, the amplitude of each

state is $1/\sqrt{2^n}$, where *n* is the number of qubits. Next, create an oracle function to performs a phase flip on the marked state. Then the diffuser operator performs an inversion of the average of the amplitudes. The qubits are measured in finally. Grover iteration (repeat oracle and amplification stages) can be repeated to increase the probability of finding the target solution. It requires approximately $(\frac{\pi}{4})\sqrt{\frac{N}{t}}$ where *N* is the number of states and *t* is the number of target solutions.

IV. Performing Compact Genetic Algorithm on a Quantum Simulator

The proposed algorithm consists of two methods for generating individuals. The first method, as shown in Fig. 1, is to generate an individual using qubit rotation based on probabilities and then applying measurement to generate an individual. So, the amplitudes of the individuals are updated by a rotation of quantum gates, which is used to generate the first individual. The second method initializes the system according to the probability distribution in a quantum register. This method is used for generating the second individual. The implementation of Grover's search algorithm is described in the next subsection, followed by an overview of the proposed algorithm.

A. Initial State in Grover's Search Algorithm

In this quantum version, the population is represented as a probability distribution in a quantum register. At each generation, the probability vector is updated towards the better individual and it is used to generate an individual for the next generation. Thus, the initial state of Grover's search algorithm for the second generation onwards should be adjusted according to the probability distribution. Grover's algorithm always begin with a uniform distribution. We assumed some other initial distribution $|w\rangle$, which is translated directly from probability vector of cGA. The initial state of the Grover's search algorithm is similar to the method for generating the first individual as shown in Fig. 1.

Since an arbitrary single-qubit state can be written as follows:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{1}$$

where θ, and ϕ are real numbers. The numbers $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ define a point on a unit three-dimensional sphere. To initialize stage of Grover's search algorithm based on the probabilities, we applied the qubit rotation for $angle(\theta)$, which is defined as following:

$$angle(\theta) = (probability(p) - 0.5) \times \pi \tag{2}$$

Therefore, all qubits are rotated along the *Y*-axis as $angle(\theta)$ to initialize the state according to the probabilities. A new Grover circuit is shown in Fig. 1.

B. Creating An Oracle Function in Grover's Search Algorithm

The key idea is to define a function *f(x)* such that *f(x)* = 1 if *y(x)* solves the search problem, and otherwise. This function is called an oracle. So different search problems need different oracles. In this work, travelling salesman problem (TSP) is presented as a test problem since it is a common optimization issue seen in a variety of fields of science and engineering. The objective function is to find a shortest route that a salesman visits every city exactly once and returns to the starting point. In our oracle, we only focus on defining an oracle to recognize all feasible solutions. The TSP are mapped to the Hamiltonian cycles problem, that is reducing the problem to the decision form of an Ising model with scales (*N*-1)^2 spins are required, where *N* is the number of cities, and we designate city 1 to appear first in the Hamiltonian cycle. For example, a route $1 \to 2 \to 3 \to 4 \to 1$ is represented by the matrix $(N-1) \times (N-1)$. So, it required a quantum register that contains (*N*-1)^2 qubits to represent a solution. We can unpack this matrix into a vector $X = [1,0,0,0,1,0,0,0,1]$. Each cell of the matrix is a variable $X_{i,p}$ where *i* represents the node and *p* represents its order in a prospective cycle, and has a value of 1 if city *i* is visited at order *p*, and 0 otherwise. This encoding scheme helps to create a quantum circuit for the problem easier [20]. The binary variable $X_{i,p}$ = 1 is converted to the spin value $\sigma_{i,p} = 1$, and $X_{i,p}$ = 0 is converted to $\sigma_{i,p} = -1$ by the following formula:

$$X_{i,p} = (\sigma_{i,p} + 1)/2 \tag{3}$$

Although we requires a lot of qubits to represents all solutions, Grover's algorithm can look at all solutions simultaneously utilizing quantum parallelism and the oracle helps to increase the probability of all feasible solutions, which is total $(N-1)!$ feasible solutions from all $2^{(N-1)^2}$ solutions. The total energy of the system is called Hamiltonian (*H*). The energy of the Ising model which TSP is mapped to is $H = H_A + H_B$, with H_A the Hamiltonian given for the undirected Hamiltonian cycles problem [20] which is the total distance for each route. We then simply add H_B the constraints for the rows and columns with the following formula:

$$H_B = B\sum_p \underbrace{\left(1 - \sum_i X_{i,p}\right)^2}_{} + B\sum_i \underbrace{\left(1 - \sum_p X_{i,p}\right)^2}_{} \tag{4}$$

where the terms in underbraces are squared so that the lowest possible minimum value is zero. The factor *B* is a weight of penalty term, has large enough weight to avoid an infeasible solution. So, the fitness function of TSP is represented by the energy of the Ising model which we need to minimize the energy to get the shortest feasible route.

To check a feasible solution on the quantum state, we simply need to create a classical function on a quantum circuit to check down both columns and across both rows have "1" appearing in one place because every city can only appear once in the cycle, and for each time a city has to occur. We compile this set of comparisons into a list of clauses and check these clauses computationally using the XOR gate. For instance, the oracle to verify a solution of TSP 3-city is shown in Fig. 2.

In part of computing clauses, we complete a checking circuit to provide a single qubit to be "1" of each clause, and then we repeat the XOR gate for each pairing in the list of clauses. The output qubit is flipped when all the clauses are satisfied. Finally, all clauses qubits are reset by repeating the part of the circuit that computes the clauses.

C. Revising Diffusion Operator in Grover's Search Algorithm

Grover's search algorithm's initial state is based on the probability distribution in a quantum register, so the diffusion operator needs to be changed as well [21]. We assume the initial probability distribution $|w\rangle$, which is a subset of a superposition. So, the new Grover diffusion operator is written $2|w\rangle\langle w| - I$. The new diffuser circuit is drawn in Fig. 3

Algorithm 2 shows the pseudocode of Grover-assisted cGA*. The two candidates are generated by the first method and the second method in steps 4 and 5 respectively, which were run on the IBMQ QASM simulator. Grover iterations in the second method ensure to get the second individual which is a feasible solution. Then the candidates are converted as spin values according to equation (3) to represent the solution with the Ising model. The fitness value is a Hamiltonian energy of the Ising model $(H_A + H_B)$. Then the evolutionary process in the cGA is used to update the probability vector towards the better individual and the new probability value will affect the angle value (θ) used for the quantum preparation state in the next iteration. It is repeated until the vector converges.

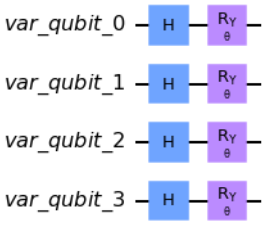

Fig. 1. A circuit to initial the state of the system based on the probability distribution which encoding problem with 4 qubits.

V. Testing Problems and Experimental Setup

From the previous section, we illustrated how to encode the TSP as a quantum state and build an oracle in Grover's algorithm. It can be seen that the proposed algorithm uses a binary encoding as an Ising model that requires $\mathcal{O}(N-1)^2$ qubits. On the other hand, encoding the problem to be addressed on the classical cGA is a different way. We can minimize the number of required bits by adapting the path representation model which represents a feasible tour as possible permutations of the N cities. Thus, a total number of feasible edges between cities are defined as:

$$\text{Number of feasible edges} = \frac{(N-1)}{2} \times N \qquad (5)$$

A total number of feasible edges is a set of l-bit binary string. The probability value of each bit $(P_{i,j})$ represents the probability value of edge between city i and city j.

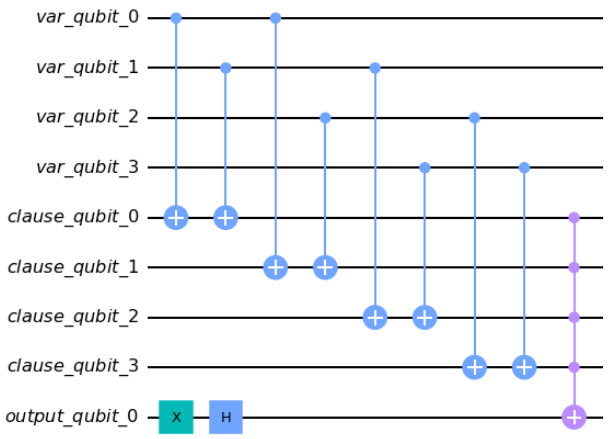

Fig. 2. An example of oracle circuit for TSP 3-city.

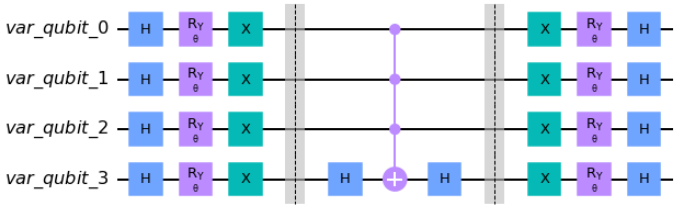

Fig. 3. A diffuser circuit which encoding problem with 4 qubits.

We employed TSP 3 and 4-city as test problems in the studies, as shown in Fig. 4, and conducted the quantum computation phases on the IBMQ QASM simulator. Since the limitation of the number of qubits supported by the simulator, so we can only demonstrate small size TSP. The classical cGA and Grover-assisted cGA* were compared in terms of solution quality and number of function evaluations that were run from population size 4–50 for TSP 3-city and population

Algorithm 2: Grover-assisted cGA*

Steps 1 and 2 are similar to steps 1 and 2 in Algorithm 1

3) initialize quantum register, classical register, circuit:
$circuit \leftarrow QuantumCircuit(qr, cr)$;

4) generate first individual using qubit rotation based on the probabilities:
$a \leftarrow generateFirstIdv(p)$;

5) generate second individual using the adjusted Grover's algorithm with oracle of the objective function:
$b \leftarrow generateSecondIdv(p)$;

6) map to an Ising model:
$a \leftarrow ising(a)$ $b \leftarrow ising(b)$;

The remaining steps are similar to steps 4 to 8 in Algorithm 1.

size 4–100 for TSP 4-city with increments of 2. The proposed method's number of function evaluations can be estimated by multiplying *Number of shots* by *Number of Grover iterations*, plus *one* for evaluation its fitness. The data were averaged over 25 runs and used tournament selection with $s = 2$. The quantum computation phases were run with the various number of shots and Grover iterations. All runs ended when the vector fully converges. We used 1 shot for TSP 3-city and used 1, 10, and 20 shots for TSP 4-city. The required number of Grover iterations for TSP 3-city (encode with 4 qubits) is approximately 2 iterations at maximum because there is a total of 2^4 population and 2 target solutions that can provide a feasible route. TSP 4-city (encode with 9 qubits) requires approximately 7 Grover iterations at maximum since there is a total of 2^9 population and 6 target solutions [2]. In this experiment, different numbers of Grover iterations and shots were used to see how they affect solution quality and performance.

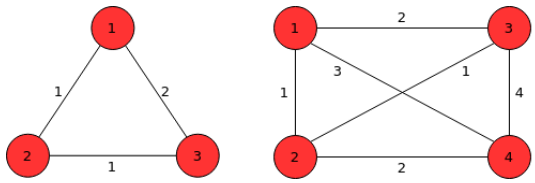

Fig. 4. TSP 3-city and 4-city.

VI. Experimental Results and Analysis

Experiments on TSP 3-city are shown in Fig. 5 and Fig. 6. The results obtained show that the two algorithms are equivalent in terms of solution quality but the cGA uses a smaller number of function evaluations. One Grover iteration is enough for solving TSP 3-city with 100% reaching the target solution because there have been just 4 qubits that represent all 16 quantum states [2]. Fig. 7 and Fig. 8 shows the experimental results on the TSP 4-city The cGA obtains higher solution quality than the proposed algorithm for the various number of shots and Grover iterations with a smaller number of function evaluations. When compared to the others, the classical cGA can achieve the target solution with 8 population sizes and approximately 22 function evaluations, whereas the cGA* with 7 Grover iteration, 20 shots, gives the solution quality as close to the classic but uses more function evaluations. From Fig. 7, it can be seen that increasing the number of Grover iterations (green line) yields a higher solution quality than increasing the number of shots (red line) and increasing both the number of shots and Grover iterations (dark green line) yields a higher solution quality than increasing the number of Grover iterations alone. It is because applying adequate Grover iterations results in a higher probability of finding the target solution, whereas the number of shots aids in obtaining a probability distribution of results. However, the number of function evaluations taken to converge as shown in Fig. 8 is a quadratic growth according to the number of shots and Grover iterations. On a classical cGA, the TSP 4-city uses 6 bits to represent all 2^6 solutions, whereas on a quantum computer, the TSP 4-city uses 9 qubits to represent all 2^9 solutions. Therefore, the classical one can achieve the target solution using evolutionary selection with a small number of function evaluations on small problem sizes. Considering TSP 10-city, the problem is encoded as 45 bits on the classical cGA that represent all 2^{45} solutions, resulting in a very wide search space with a large time complexity to solve the problem. So, quantum parallelism in Grover's algorithm could play an important role in achieving benefits over a classical computer to tackle medium problem sizes upwards. This would require improving a binary encoding scheme of the problem and the quantum resources required in quantum hardware.

Fig. 5. Comparison of the solution quality (number of correct bits in percentage at the end of the run) achieved by the classical cGA and the cGA* on TSP 3-city.

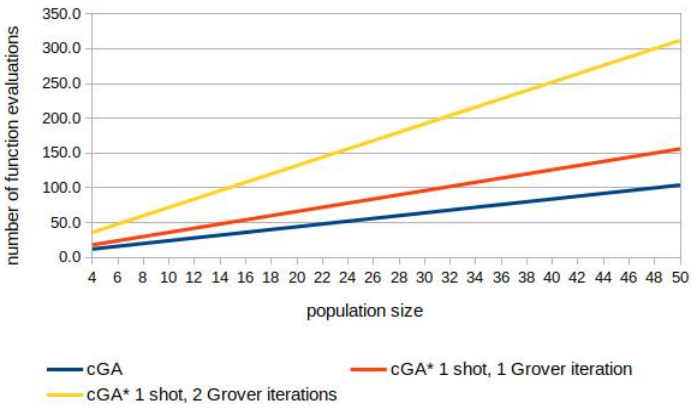

Fig. 6. Comparison of the classical cGA and the cGA* in the number of function evaluations needed to achieve convergence on TSP 3-city.

VII. Conclusion

This paper proposes a simulation of Grover-assisted compact genetic algorithm to solve some real-world problems which requires fewer number of qubits than traditional GA. It combines the Grover's search algorithm, which assigns a specific state to the system based on the probabilities, with the generation of an individual process in a cGA with elite.

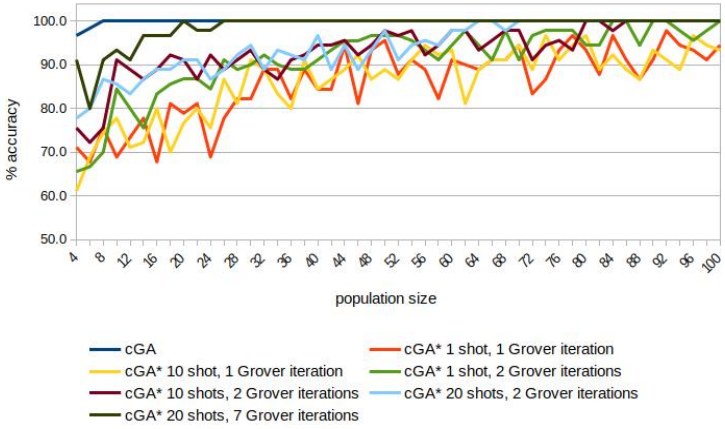

Fig. 7. Comparison of the solution quality (number of correct bits in percentage at the end of the run) achieved by the classical cGA and the cGA* on TSP 4-city.

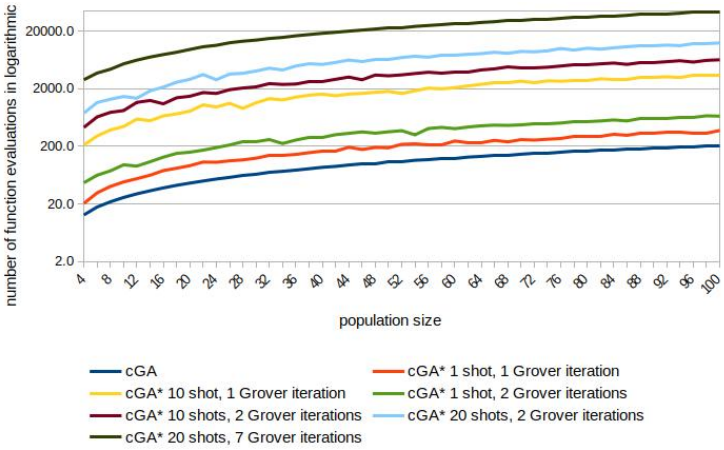

Fig. 8. Comparison of the classical cGA and the cGA* in the number of function evaluations (are plotted in logarithmic scale) needed to achieve convergence on TSP 4-city.

The studies are carried out on TSP using different encoding schemes between the classical cGA and the Grover-assisted cGA* to minimize the number of bits and qubits necessary. TSP is mapped to an Ising model for the proposed algorithm, making it easier to build a quantum circuit. The results show that the classical method outperforms the Grover-assisted cGA* in terms of solution quality and performance on small problem size. However, for medium size upwards, quantum parallelism in Grover's algorithm could play a key role in achieving benefits over a classical computer, which would necessitate enhancing the problem's binary encoding strategy and the quantum resources required in quantum hardware.

ACKNOWLEDGMENT

KS and PC acknowledge financial support from the Program Management Unit for Human Resources and Institutional Development, Research and Innovation (Grant Number B05F630108), Thailand. NB acknowledges support by MEXT Quantum Leap Flagship Program (MEXT QLEAP) Grant Number JPMXS0118067285 and JPMXS0120319794.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, Oct 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[3] D. wei Gong, N. na Qin, and X. yan Sun, "Evolutionary algorithms for optimization problems with uncertainties and hybrid indices," *Information Sciences*, vol. 181, no. 19, pp. 4124–4138, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025511002544

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[6] G. Harik, F. Lobo, and D. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, 1999.

[7] C. Aporntewan and P. Chongstitvatana, "A hardware implementation of the compact genetic algorithm," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, 2001, pp. 624–629 vol. 1.

[8] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 61–66.

[9] Z. A. E. M. Dahi, C. Mezioud, and A. Draa, "A quantum-inspired genetic algorithm for solving the antenna positioning problem," *Swarm and Evolutionary Computation*, vol. 31, pp. 24–63, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650216300293

[10] W. Chmiel and J. Kwiecień, "Quantum-inspired evolutionary approach for the quadratic assignment problem," *Entropy*, vol. 20, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1099-4300/20/10/781

[11] J. King *et al.*, "Quantum-assisted genetic algorithm," 2019.

[12] J. Supasil, P. Pathumsoot, and S. Suwanna, "Simulation of implementable quantum-assisted genetic algorithm," *Journal of Physics: Conference Series*, vol. 1719, no. 1, p. 012102, jan 2021. [Online]. Available: https://doi.org/10.1088/1742-6596/1719/1/012102

[13] A. Malossini, E. Blanzieri, and T. Calarco, "Quantum genetic optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 231–241, 2008.

[14] 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, 05 2013.

[15] S. Yingchareonthawornchai, C. Aporntewan, and P. Chongstitvatana, "An implementation of compact genetic algorithm on a quantum computer," in *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*, 2012, pp. 131–135.

[16] B. Rylander, T. Soule, J. Foster, and J. Alves-Foss, "Quantum genetic algorithms." 01 2000, p. 373.

[17] A. Layeb and D. Saidouni, "Quantum genetic algorithm for binary decision diagram ordering problem," 2007.

[18] Z. Laboudi and S. Chikhi, "Evolving cellular automata by parallel quantum genetic algorithm," in *2009 First International Conference on Networked Digital Technologies*, 2009, pp. 309–314.

[19] M. S. ANIS *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.

[20] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: http://dx.doi.org/10.3389/fphy.2014.00005

[21] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum Amplitude Amplification and Estimation," *arXiv:quant-ph/0005055*, vol. 305, pp. 53–74, 2002.