ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Compact Genetic Algorithm with Quantum-Assisted Feasibility Enforcement

Kamonluk Suksen[1], Naphan Benchasattabuse[2] and Prabhas Chongstitvatana[3]

ABSTRACT

The local optima problem is one of the biggest obstacles in compact genetic algorithms since each bit in the problem encoding string is independent of the others. We propose a quantum-assisted compact genetic algorithm that uses a quantum amplitude amplification technique in the selection process to circumvent the said problem. In addition to using elitism mechanics where a single best candidate solution is kept to drive the probability vector, the amplitude amplification subroutine also acts as a mutation operator which, with high probability, enforces the constraint that the newly generated candidate is a feasible solution. We demonstrate this idea by applying the algorithm to the traveling salesman problem of size 3 and 4 cities on IBM Qiskit simulator to show how one would construct the quantum circuit and how to encode the optimization problem into quantum states via Ising spin model encoding. We then show space and time complexity analysis based on the quantum circuit model. Finally, we discuss the number of qubits required for encoding, gate counts in the circuit model, and the practicality of applying this to a small-scale devices in the future.

Article information:

Keywords: Quantum Computing, Grover's Search Algorithm, Compact Genetic Algorithm (cGA), Traveling Salesman Problem

Article history:
Received: April 8, 2022
Revised: June 13, 2022
Accepted: July 30, 2022
Published: October 15, 2022 (Online)

DOI: 10.37936/ecti-cit.2022164.247821

1. INTRODUCTION

Genetic algorithms [1] are one of the most widely used variants of heuristic algorithms in the family of evolutionary algorithms. They try to solve the optimization problem by mimicking behaviors in nature, such as natural selection and survival of the fittest. Although genetic algorithms excel at refining the solution quality (local search) via the recombination process, one main obstacle of genetic algorithms is the good incorporation of the global search operator to circumvent the local optima problem. This is usually done via good initialization and preserving the population or using a mutation operator, which results in large memory requirements [2]-[5].

Among various techniques to reduce the memory requirement of genetic algorithms [6]-[9], the compact genetic algorithm [10] is one such approach that greatly reduces said requirements. However, it comes with a trade-off where the global search aspect of keeping large diversity of the population is lost and the only way to prevent that is via a mutation operator, shifting the memory requirement into a more complex computation task.

Quantum computing has been predicted to have advantages over classical computing for certain tasks such as factorization [11], simulation of quantum systems [12], or generating verifiable random numbers [13], [14]. Grover's search algorithm [15], the quantum analogue of classical exhaustive search, has been proven to provide strict speedups. Although the speedup is only polynomial, it has been shown to have a wide range of applications as a subroutine in classical algorithms [16]-[20] due to its query complexity being $\mathcal{O}(\sqrt{\mathcal{N}})$ compared to the classical $\mathcal{O}(\mathcal{N})$, when the problem has no structure to be exploited.

There have been attempts to combine quantum computing and genetic algorithms in the hope that their quantum versions would provide speedups or novel ideas to the classical algorithms. Quantum-inspired genetic algorithms [21]-[23] are one of the attempts where the algorithms are still executed purely in classical computers, but take some of the ideas

[1,3]The authors are with Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand, E-mail: 6071401321@student.chula.ac.th, prabhas.c@chula.ac.th
[2]The author is with Graduate School of Media and Governance, Keio University SFC, Fujisawa, Kanagawa 252-0882, Japan, E-mail: whit3z@sfc.wide.ad.jp

of quantum phenomena such as interference and superposition and translate them into classical analogues. Another approach, quantum-assisted genetic algorithms [24]-[27] and quantum-assisted compact genetic algorithm [28], delegates some of the complex tasks to quantum computers, such as the mutation operator or probabilistic elements, while still performing crossover and population updates on the classical side. The last approach attempts to redefine the GA in the context of quantum computation [29]-[31].

In this work, we take the quantum-assisted algorithm route and aim to explore the use of Grover's algorithm as a means to incorporate a global search operator into a compact genetic algorithm. Not only does the quantum search subroutine help with expanding the search from a local to a global aspect, but it also serves as a means to restrict the candidate generation process to produce a feasible solution with high probability. We use the traveling salesman problem (TSP) as an example of our proposed algorithm since it is a non-trivial optimization problem that requires a rather complex encoding and it has a tendency to get stuck in local optima without a mutation operator. We present one approach to encoding the TSP into quantum states using the Ising model [32]. We show how to translate the probability vector into quantum states, how to construct an oracle from the feasibility constraints, and how to construct the diffusion operator with respect to the probability vector. Although we use TSP in our example, the methods outlined in this paper are general and can be adapted to most optimization problems.

The paper is structured as follows. We first provide background on the classical evolutionary algorithm and the setting of the compact genetic algorithm which we will use as our baseline in comparison to our quantum-assisted proposal (Section II). We then describe the quantum search algorithm (Section III), our proposed algorithm, how one could encode the optimization problem into our setting using the Ising encoding, and how to construct the quantum circuit to realize our quantum subroutine (Section IV). We then talk about our experimental setup which was done using the IBM Qiskit simulator [33] (Section V). We discuss results comparing the classical compact genetic algorithm with elitism and our proposed quantum-assisted version (Section VI). We also provide a cost analysis of our approach in terms of the number of qubits and circuit costs which are directly related to the quantum space and time complexity (Section VII). We end the paper by discussing the possible ways of extending our approach into a more quantum analogue of the compact genetic algorithm, giving connections to QAOA [34], and how to include the building block structure of encoding (Section VIII).

2. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA) are heuristic-search algorithms based on the concepts of biological evolution, such as reproduction, mutation, recombination, and selection [35],[36]. EA often performs well in hard problems where the underlying problem structure is not known, since it does not use any knowledge of the underlying structure. EA operates by maintaining a population of candidate solutions or individuals, and then applying the principle of survival of the fittest by encoding the optimization goal to the environment where the population lives. Each individual has a respective fitness value indicating how close it is to the optimal solution. Higher values are closer. The problem space is encoded in this population representation, and a good EA should have the ability to guide the population or be initialized such that most of the space can be reached. The population will then evolve for some time in time steps referred to as generations. In each generation, a group of individuals with high fitness is selected as parents, and then through the application of the breeding operator, which utilizes the concepts taken from nature such as recombination and mutation, a new set of individuals will be generated and replace those of the current population with low fitness. From this process, the later generations should tend towards a population with higher and higher average fitness until it reaches the point of equilibrium where the average fitness stays constant. At this point the algorithm will be stopped. This process is akin to Darwin's natural selection and evolution theory, where newer individuals are better suited to the environment than their ancestors.

2.1 Genetic algorithm

Genetic Algorithms (GA) are one of the most common and well-known variants of EAs [1]. In a GA, the candidate solutions or individuals are encoded into fixed-length strings referred to as *chromosomes*. Each section of the string encodes each variable of the problem. The population size of a GA can be static or dynamic. In the simplest form of GA, the simple genetic algorithm (SGA), a set of individuals is randomly selected from the population. A fixed number of individuals with the highest fitness are selected as parents, and a new set of offspring is created by combining substrings from those parents (crossover). Mutations can also be applied to those offspring strings if one desires. One drawback of GAs is the memory required to maintain the population size. If the population size is not big enough, or the initial population is not spread enough across the entire space, it will be easier for evolution to get stuck at a local optimum as the search space gets larger.

2.2 Compact genetic algorithm

The compact genetic algorithm (cGA) [10] was proposed as a way to get around the memory requirement of the original GA. The problem encoding string is now not any general string. Instead, it is a bit string consisting of only 0 and 1. Instead of maintaining the whole population string, a *probability vector* (p) of the same length as the chromosome string is used. At index i of p (p_i) stores a real number between 0 and 1 which represents the ratio of the current population's chromosome at the same index being 1 to the whole population. In other words, the probability that the chromosome at index i being 1 if we were to randomly select an individual from the population. This approach changes the memory requirement from scaling with population size to only the length of the problem encoding bit string and allows for easier hardware-based acceleration [37]. The evolution process for cGA is usually done by sampling the probability vector p to get two individuals, evaluating their fitness, and adjusting the probability vector closer to the one with higher fitness if the chromosome bit in that position differs. The performance of cGA is comparable to that of SGA with uniform crossover and low underlying problem structure. Since cGA can be viewed as a random walk on each bit of the chromosome, each bit is independent of the others. This can lead to cGA getting stuck on the local optima and not reaching an optimal solution when no mutation is considered.

2.3 Compact genetic algorithm with an elite

Many techniques have been proposed to circumvent the problem of cGA getting stuck in local optima due to each bit being independent of the others. One such approach is to include *elitism* into the cGA process. For GA, the selection process can play a huge role in the potency of the algorithm [38]. By maintaining an "elite set" comprised of top n best individuals from the population as the only candidates for the selection of parents, the problem can be alleviated a bit by always guaranteeing that once a good candidate has been found, it will be used to drive the population so that the newer generations will not become worse [39], [40]. This is especially useful for cGA since it suffers heavily from this issue when the problem has a high building block structure.

We define the cGA with elitism (cGA*) where we only keep one best candidate which we will use as the baseline for comparison in Algorithm 1. In the selection procedure, the first individual is sampled from the probability vector. The second individual is also sampled in the same way, but it will be replaced by the "current best candidate" if its fitness is lower. The current best candidate is kept and always updated when an individual with higher fitness is found.

Algorithm 1: The cGA*

```
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 $winner[i] = 1$ 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.
```

3. QUANTUM SEARCH ALGORITHM

Let us briefly review the difference between quantum states and classical states. Unlike a classical bit where a bit can only hold two values, 0 or 1, a quantum bit or qubit can represent a superposition of $|0\rangle$ and $|1\rangle$ as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers referred to as amplitudes with the constraint that $|\alpha|^2 + |\beta|^2 = 1$. And in contrast to the classical case where n classical bits can represent 2^n different states, in general, the quantum state of n qubits has 2^n basis states. It can represent a linear combination of all these basis states as

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i|i\rangle, \tag{2}$$

where $N = 2^n$ the normalization condition forces the amplitudes to follow $\sum |\alpha_i|^2 = 1$. This means that a quantum state of n qubits needs 2^n numbers to describe. The modulus squared of each basis state's amplitude refers to the probability of observing each basis state when the state is measured. After a quantum state is measured, the superposition is destroyed and the state will stay in the measured state with unit probability. For full details of quantum information processing, we refer the readers to [41].

Grover's search algorithm is a quantum unstructured search analogous to the classical exhaustive search which exhibits speedups that no classical algorithm can achieve [15]. The key ingredient of Grover's algorithm is the ability to utilize the superposition of all possible solution states and amplify the probability to observe the solution while lowering the probability to observe a non-solution state. Grover's algorithm can achieve this task with a query complexity of $\mathcal{O}(\sqrt{\mathcal{N}})$, where N is the number of possible solutions. In contrast, the same task is bounded by $\mathcal{O}(\mathcal{N})$ for the classical case. Grover's algorithm is a special case of a more general technique called *amplitude amplification* [42] which is known to be optimal when the problem does not have any structure we can exploit [43], [44].

Grover's algorithm consists of three main components: (1) quantum state preparation, (2) oracle application, and (3) diffusion. Grover's algorithm is an iterative process. After initializing the states, the oracle operator is applied, followed by the diffusion operator. This application of the two operators is denoted as one iteration of the algorithm.

The first component, quantum state preparation, is usually done by preparing a uniform superposition of all states via the application of the Hadamard gate to all qubits. If only a certain part of the search space is required, one can also prepare the state in such a way that it preserves the search space throughout the algorithm as well [45].

The second component is the oracle operator. The oracle is an unitary operator O which differentiates basis states into two sets, the solution set and the non-solution set. The action of oracle operator O can be described mathematically as

$$|x\rangle \xrightarrow{O} (-1)^{f(x)}|x\rangle = \begin{cases} |x\rangle & x \text{ is not a solution} \\ -|x\rangle & x \text{ is a solution,} \end{cases} \tag{3}$$

where $f(x)$ is a binary function which outputs 1 when the state is in the solution set and 0 otherwise.

The final component is the diffusion operator. As we can see from the oracle operator, what it does is that it changes the phase or the states while the magnitude of the amplitude stays unchanged. The oracle operator alone does not change the probability of measuring the solution states. The diffusion operator, or the inversion around the average operator when applied, changes the magnitude of each basis state. That results in the amplification of the solution states. The changes in the amplitude of each state can be explained as

$$\alpha_i \xrightarrow{Diffusion} 2\bar{\alpha} - \alpha_i, \tag{4}$$

where $\bar{\alpha}$ is the average of all amplitudes $\bar{\alpha} = 1/N \sum \alpha_i$. It should be noted that Grover's algorithm suffers from a problem called the "souffle problem" where too many or too few iterations do not give good results. Only the right number of iterations should be performed. The optimal number of iterations for Grover's algorithm is shown to be $(\frac{\pi}{4})\sqrt{\frac{N}{T}}$, where N is the number of states and T is the number of target solutions [43].

4. QUANTUM-ASSISTED COMPACT GENETIC ALGORITHM

In this section, we will explain how to utilize Grover's search algorithm as a subroutine for the individual selection process to generate feasible candidates with high probability. This quantum subroutine not only serves as an improvement to the selection process, but can also be viewed as a mutation operator in the sense that the sampling probability at the end of the amplitude amplification process deviates from the probability vector. After the outline of the algorithm, we will show how to implement it using a quantum circuit-based model. Although we picked the Traveling Salesman Problem (TSP) as our example to show the implementation, it should be noted that the idea of encoding the problem via the Ising model works for most NP-hard problems [32].

4.1 Algorithm outline

Having established the idea behind the use of the quantum subroutine, we now present our algorithm as outlined in Algorithm 2. It has a similar structure to that of the cGA* outlined in Algorithm 1. There used cGA with elitism as the base, but the selection process to generate the second individual is not sampled from the probability vector directly. Instead, it is sampled from the quantum states prepared by the amplitude amplification process which the oracle operator defines by using a feasibility check function. This subroutine is not meant to be a full-scale Grover search. Its purpose is to broaden the reach of individuals generated from the probability vector.

4.2 Problem encoding

To encode the TSP to be addressed on the quantum processor, we mapped the TSP to the Hamiltonian cycle problem. That reduces the problem to the decision form of an Ising model [32]. It requires $(n-1)^2$ qubits to represent a solution, where n is the number of cities. We designate city 1 to always appear first in the Hamiltonian cycle. A route is represented by the matrix $(n-1)\times(n-1)$. Each cell of

Algorithm 2: Grover-assisted cGA*

```
1) Classical: initialize probability vector:
for $i \leftarrow 1$ to $l$ do
    $p[i] \leftarrow 0.5$;
end
2) Classical: initialize the current best individual and
   Grover iteraton:
$curBestIdv \leftarrow 000..00$, $t \leftarrow numGroverIteration$;
3) Quantum: initialize quantum register, classical
   register, circuit:
$circuit \leftarrow QuantumCircuit(qr, cr)$;
4) Quantum: generate the first individual using qubit
   rotation based on the probabilities:
$a \leftarrow generateFirstIdv(p)$;
5) Quantum: generate the second individual using the
   adjusted Grover's algorithm with oracle of the
   objective function:
$circuit \leftarrow initialize(p)$;
for $i \leftarrow 1$ to $t$ do
    $circuit \leftarrow feasibleSolution(circuit)$;
    $circuit \leftarrow inverseFeasibleSol(circuit)$;
    $circuit \leftarrow diffuser(circuit)$;
end
$b \leftarrow measure(circuit)$;
6) Classical: evaluate the second individual's fitness:
if $curBestIdv.fitness > b.fitness$ then
    $b \leftarrow curBestIdv$;
end
7) Classical: let them compete:
$winner, loser \leftarrow compete(a, b)$;
8) Classical: update probability vector towards winner:
for $i \leftarrow 1$ to $l$ do
    if $winner[i] \neq loser[i]$ then
        if $winner[i] = 1$ then
            $p[i] \leftarrow p[i] + 1/n$;
        else
            $p[i] \leftarrow p[i] - 1/n$;
        end
    end
end
9) Classical: update the current best individual:
if $curBestIdv.fitness < winner.fitness$ then
    $curBestIdv \leftarrow winner$;
end
10) Classical: 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, $n$ is a population size,
and $t$ is a number of Grover iterations.
```

the matrix is a variable $X_{i,p}$ where i represents the node and p represents its order in a prospective cycle, so $X_{i,p} = 1$ if city i is visited at order p, and 0 otherwise. For example, a TSP 4 cities with a route $1 \to 2 \to 3 \to 4 \to 1$ is represented by Fig. 1 or as a flatted vector $X = [1, 0, 0, 0, 1, 0, 0, 0, 1]$.

	1	2	3
1	1	0	0
2	0	1	0
3	0	0	1

Fig.1: *Matrix represents the route.*

4.3 Fitness function definition

The objective of TSP is to find the shortest route such that a salesman visits every city exactly once and returns to the starting point. We require that every node in the cycle appear exactly once, and that at each time step a node must be visited. This creates two constraints as shown below:

$$\sum_i X_{i,p} = 1; \quad \forall p \,. \tag{5}$$

$$\sum_p X_{i,p} = 1; \quad \forall i \,. \tag{6}$$

For the nodes in our prospective ordering, if $X_{i,p}$ and $X_{j,p+1}$ are both 1, then there should be an energy penalty for i and j that are not connected in the cycle. However, in this paper, it will be presumed that the graph is fully connected. As a result, this penalty term will not be included. The distance that needs to be minimized is:

$$\text{Energy} = \sum_{i,j} w_{i,j} \sum_p X_{i,p} X_{j,p+1}. \tag{7}$$

Consequently, our energy will have three components in a single objective function to be minimized, and we get the following:

$$\begin{aligned}\text{Energy} = &\sum_{i,j} w_{i,j} \sum_p X_{i,p} X_{j,p+1} \\ &+ B \sum_p \left(1 - \sum_i X_{i,p}\right)^2 \\ &+ B \sum_i \left(1 - \sum_p X_{i,p}\right)^2,\end{aligned} \tag{8}$$

where B is the weight of the penalty term and $B > \max(W_{i,j})$ to avoid an infeasible solution. The terms in underbraces are squared so that the lowest possible minimum value is zero. A ground state of this system must have zero for penalty terms *if and only if* we have an ordering of vertices where each city is only visited once, and for each time a node has to occur. Therefore, the fitness function of TSP is represented by the energy of the Ising formulation. We then need to minimize the energy of the Ising formulation to find the shortest route possible.

4.4 Quantum state preparation from probability vector

Recall that quantum states must obey the normalization condition to preserve the probability, which must be 1. We can rewrite Eq. (2) using this fact to

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, \tag{9}$$

where θ, and ϕ are real numbers with natural ranges of $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. This way of writing the qubit representation can be used to visualized the single qubit state inside a Bloch sphere. A Bloch sphere is a three-dimensional sphere where states of a qubit are points on the surface of this unit sphere. An example of $|+\rangle$, a single qubit state with equal superposition of $|0\rangle$ and $|1\rangle$, is depicted in Fig. 3.

Using the Bloch sphere representation, it is easy to see that we can use the Pauli-Y rotation operator (R_Y) to initialize each qubit to have the same probability of observing $|1\rangle$ as the probability vector where the matrix of R_Y can be defined as

$$R_Y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}. \tag{10}$$

We set the rotation angle θ to be

$$\theta = 2 \times \arccos\sqrt{1-p}, \tag{11}$$

where p is probability of observing $|1\rangle$ according to our probability vector. The quantum circuit for state initialization is depicted in Fig. 2.

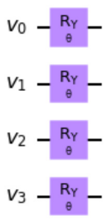


Fig.2: *A circuit to initial the quantum state of the system based on the probability distribution which encoding problem with 4 qubits.*

4.5 Oracle construction

The role of the oracle in Grover's search is to distinguish the solution states from the non-solution states in the search problem. The key idea for our subroutine here is to define a function $f(x) = 1$ if x

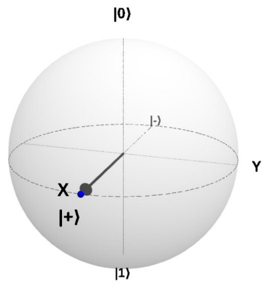


Fig.3: *Bloch sphere representation of a qubit in the state $|+\rangle$. In this case, $\theta = \frac{\pi}{2}$ and $\phi = 0$.*

is a feasible string obeying the constraints in Eqs. (5), (6), and $f(x) = 0$ otherwise.

To check a feasible solution on the quantum state, we simply need to create a function on a quantum circuit to check along both columns and rows with "1" appearing only once in each axis. This is because every city can only appear once in the cycle, and for each time a city has to occur. We compiled this set of comparisons into a list of clauses and check these clauses computationally using the XOR gate. In quantum computers, the job of the XOR gate is done by the controlled-NOT gate or CNOT. In circuit diagrams, it is drawn as in Fig. 4.

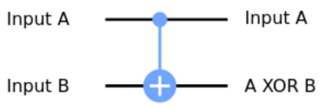


Fig.4: *A circuit of the controlled-NOT (CNOT) gate.*

The CNOT gate is applied to a pair of qubits. The one with the little dot acts as the control qubit and the other acts as the target qubit. The target becomes 1 if they are different, and 0 if they are the same. For instance, the oracle to verify a solution of TSP 3 cities is shown in Fig. 5. The TSP 3 cities is encoded with 4 qubits. Each qubit represents $X_{i,p}$, where i represents the city and p represents its order in a cycle. We consider from the second city and the second order onwards because we always start in the first city. There are $X_{2,2}$, $X_{2,3}$, $X_{3,2}$, and $X_{3,3}$. As part of computing clauses, we complete a checking circuit to provide a single qubit to be "1" of each clause, i.e. ($X_{2,2}$, $X_{2,3}$), ($X_{2,2}$, $X_{3,2}$), ($X_{2,3}$, $X_{3,3}$), and ($X_{3,2}$, $X_{3,3}$), to check that each city is only visited once, and for each time a city has to occur. 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 clause qubits are reset by repeating the part of the circuit that computes the clauses.

4.6 Diffusion operator construction

The conventional Grover diffusion operator (the inverse around the average operator) was made that way due to the initial state being uniform superposition. Since we prepare our initial state from the probability vector, we adopt the geometry preserving diffusion operator as mentioned in [43]. This way we can make a reflection around the initial state instead and that allows us to enforce the feasibility to quantum state even when the probability vector is near its convergence point. As a result, the diffusion operator we used can be written as $2|w\rangle\ \langle w| - I$ or $w[|0^n\rangle\langle 0^n| - I]w^\dagger$, where $|w\rangle$ is our initial state prepared from the probability vector. A multi-controlled phase gate can be used to build a reflection for the specific state in the circuit model. The diffusion circuit is depicted in Fig. 6.

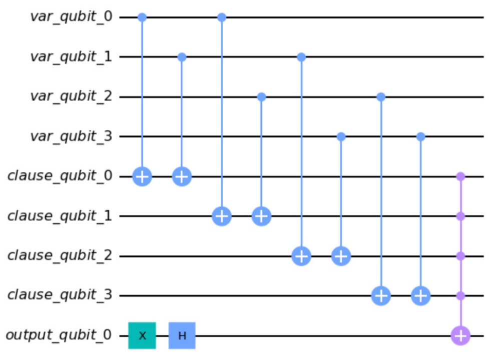


Fig.5: *An feasibility check oracle circuit for TSP 3 cities.*

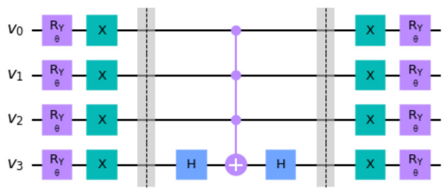


Fig.6: *A diffusion circuit which encoded a problem with 4 qubits.*

5. TESTING PROBLEMS AND EXPERIMENTAL SETUP

In the previous section, we illustrated how to encode the TSP as a quantum state and how the proposed algorithm works. It can be seen that the proposed algorithm uses a binary encoding as an Ising formulation that requires $(n-1)^2$ qubits. On the other hand, we use a different method to encode TSP to our baseline cGA* which we will compare the quantum-assisted version against.

5.1 Encoding traveling salesman problem on a classical computer

By encoding the TSP to be addressed on a classical computer, we 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, the total number of feasible edges between cities is defined as:

$$\text{Number of feasible edges} = \frac{(n-1)}{2} \times n. \qquad (12)$$

The total number of feasible edges is a set of l-bit binary strings, where l in the compact representation is expressed by $\lceil \log_2(n) \rceil$ bits. This means that an order $\mathcal{O}(n \log n)$ bits is required to encode n cities in the TSP, compared to the Ising representation of order $\mathcal{O}(n^2)$. The probability value of each bit $(P_{i,j})$ represents the probability value of an edge between city i and city j. In this experiment, we illustrated examples of the traveling salesman problem by taking 3 and 4 cities, which are shown in Fig. 7 and Fig. 8, respectively.

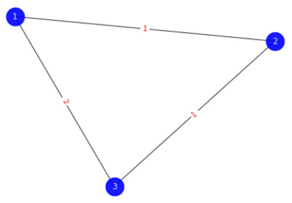


Fig.7: *TSP 3 cities.*

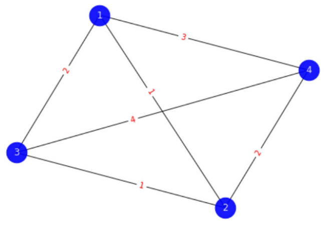


Fig.8: *TSP 4 cities.*

5.2 Experimental setup

We employed TSP 3 and 4 cities as test problems in this study, as shown in Fig. 7 and Fig. 8, respectively. The quantum computation phases were run on the IBMQ QASM simulator. Since the number of qubits supported by the simulator is limited, we can only demonstrate a small-sized TSP. The cGA* and quantum-assisted cGA* were compared in terms of solution quality and the number of function evaluations that were run. We used a population size of 50 for TSP 3 cities and a population size of 100 for TSP 4 cities with increments of 2. The number of function evaluations of each generation of the proposed method can be estimated by multiplying *Number of shots* by *Number of Grover iterations*, plus *one* for evaluation of its fitness. The number of shots is the number of executions of a quantum circuit to sample from quantum states. The number of Grover iterations is the number of oracle operator applications in each run of the quantum subroutine.

The data was averaged over 25 runs and we used tournament selection with $s = 2$. The quantum computation phases were run with the various numbers of shots and Grover iterations. All runs end when the vector fully converges. We used 1 shot for TSP 3 cities and used 1, 10, and 20 shots for TSP 4 cities. The required number of Grover iterations for TSP 3 cities (encoded with 4 qubits) is approximately 2 iterations at maximum because there is a total size of 2^4 population and 2 target solutions that can provide a feasible route. TSP 4 cities (encoded with 9 qubits) requires approximately 7 Grover iterations at maximum since there is a total size of 2^9 population and 6 target solutions. In this experiment, various numbers of Grover iterations and shots were used to see how they affected solution quality and performance.

6. RESULTS

Experiments on TSP 3 cities are shown in Fig. 9 and Fig. 10. The results obtained show that the two algorithms are equivalent in terms of solution quality, but the classical cGA* uses a smaller number of function evaluations than the quantum-assisted cGA*. Because there are only 4 qubits that represent all 16 quantum states, one Grover iteration is sufficient for solving TSP 3 cities with 100% success. Fig. 11and Fig. 12 show the experimental results on the TSP 4 cities. 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 quantum-assisted cGA*, with 7 Grover iterations and 20 shots, gives a solution quality close to the classic but it uses more function evaluations. From Fig. 11, 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). 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. This is because applying adequate Grover iterations results in a higher probability of finding the target solution. In addition, increasing the number of shots is the same as increasing the tournament selection size. As a result, the chance of finding better individuals increases in each generation. However, the number of function evaluations taken to converge, as shown in Fig. 12 has quadratic growth according to the number of shots and Grover iterations. On a classical cGA*, the TSP 4 cities uses 6 bits to represent all 2^6 solutions. On a quantum computer, the TSP 4 cities 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-sized problems.

7. QUANTUM COMPLEXITY ANALYSIS

For our proposed quantum-assisted cGA*, the quantum complexity analysis can be divided into two parts: analyzing the number of qubits and the number of ancilla qubits required, and the cost defined by CNOT count and circuit depth. The number of qubits required depends on the number of cities. The CNOT cout and circuit depth depend on Grover iteration used in Grover's search algorithm. In this work, we only considered the number of CNOT gates used because the CNOT gate is the most costly gate. It requires a longer time to execute and is more error-prone than single qubit gates. In the literature, CNOT count is also commonly used to compare good circuit design for any quantum computers in general [46]-[48]. In addition, we considered the circuit cost from circuit depth complexity as well. This metric matters because qubits have finite decoherence time. This decoherence time restricts the depth of the circuit (the number of operations that may be performed before qubits lose their quantum mechanical properties), making long execution time algorithms impossible to implement on existing NISQ devices. Finally, we provided an analysis of the total quantum complexity of the algorithm.

7.1 The number of qubits required

We approached the TSP by encoding it with an Ising model with a polynomial number of spins/qubits which scales as $(n-1)^2$, where n is the number of cities. In general, we may require ancilla qubits, which are required to be used during computation in addition to input and output qubits. They speed up the computation process by reducing the total of gates or circuit depth [49]. The number of ancilla qubits used in our straightforward implementation explained in Section IV-E can be counted easily. We need 1 qubit for verifying that 1 appears once in every row and every column of $X_{i,p}$. This means that total number of ancilla qubits is $2(n-1)$, which is shown in Table 1.

7.2 Circuit cost

In this work, we assume the topology of the quantum device to be all-to-all which means that CNOT can be placed directly between any two qubits in the quantum circuit. It should be noted that this is not the case for most real devices. However, it is a good benchmark to use, because after the circuit is tran-

Table 1: *Circuit cost summary classified as number of qubits, number of ancilla qubits, CNOT gate count, and circuit depth on TSP.*

Number of cities	**Number of qubits**	**Number of anc. qubits**	**CNOT count**	**Circuit depth**
3	$(n-1)^2$	$2(n-1)$	$57t$	$2+99t$
4	$(n-1)^2$	$2(n-1)$	$315t$	$1+549t$

n is the number of cities.
t is the number of Grover iterations.

spiled to a real device of any qubit topology, the increase in CNOT is only polynomial. As shown in Table 1, in the first Grover iteration for TSP 3 cities, there are 57 CNOT gates and the depth of the circuit is 101, which is the longest path in the circuit between the data input and the output. For the TSP 3 cities, applying more Grover iterations makes the number of CNOT gates increase by $57t$ and the depth of the circuit increases by $2 + 99t$, where t is the number of Grover iterations. The number of CNOT gates and the depth of the circuit both increase by $315t$ and $1 + 549t$, respectively, for the TSP 4 cities.

Table 1 shows that when the number of cities increases, the number of qubits grows quadratically, while the number of ancilla qubits grows linearly. Furthermore, the circuit cost defined by the CNOT count, and the circuit depth, grows linearly with the number of Grover iterations. However, the entire search space of the problem increases exponentially as the number of qubits increases.

8. CONCLUSION

In this paper, we have explored the prospects of using a quantum search algorithm as a means for implementing a global search operator, a mutation operator, and enforcing a selection procedure to generate feasible candidates with high probability to circumvent the local optima problem of the compact genetic algorithm. Although the idea of using Grover's search to get around the local optima and to speedup the convergence rate is not new, it has not previously been shown directly in the compact genetic algorithm model. We also took a step further than just outlining the theoretical method by showing a working circuit construction, and then simulating it via the IBM Qiskit simulators.

In this work, we assume that our underlying quantum operations are perfect. This would not be the case if we were to realize this algorithm on real quantum computers because current real devices are subject to noise from imperfect quantum operations [50] and also noise from the environment [51]. It is hard to completely isolate the quantum system from the environment. It is well-known that amplitude amplification algorithms are very sensitive to noise, but it would still be interesting to see how that would effect our proposed quantum-assisted algorithm. We do not require a perfect Grover's algorithm execution to see an improvement and only a few iterations would be required for our task. We hope to answer this question in future work.

One possible extension of this work would be to include the fitness function evaluation in the oracle. This would make the quantum subroutine produce higher fitness offspring with high probability. On the other hand, one might require more Grover iterations with this extension. It would be interesting to see how the two approaches fare against one another. Also, in contrast to the usual evolutionary algorithm analysis where reasoning about bounds on convergence time is often difficult, if not impossible, this would make the whole process closer to the adaptive quantum search [52]. That in turn would allow us to give bounds and theoretical framework to compare against other variational algorithms such as QAOA [34].

Finally, we suspect that in order to fully exploit the quantum properties like entanglement in the compact genetic algorithm, one would need to consider the building block theory of genetic algorithms [53] during circuit construction. Although the quantum search subroutine already provides global search capability and feasibility enforcement in the compact genetic algorithm, we should be able to further decrease the problem space to better represent the correlation of the chromosome representation. It could be done with the entangling operator, and could speed up the rate of convergence of the probability vector. We also leave such an extension as future work.

ACKNOWLEDGMENTS

KS and PC acknowledge financial support from the NSRF via the Program Management Unit for Human Resources and Institutional Development, Research and Innovation [Grant Number B05F640051]. NB acknowledges support from MEXT Quantum Leap Flagship Program Grant Number JPMXS0118067285 and JPMXS0120319794.

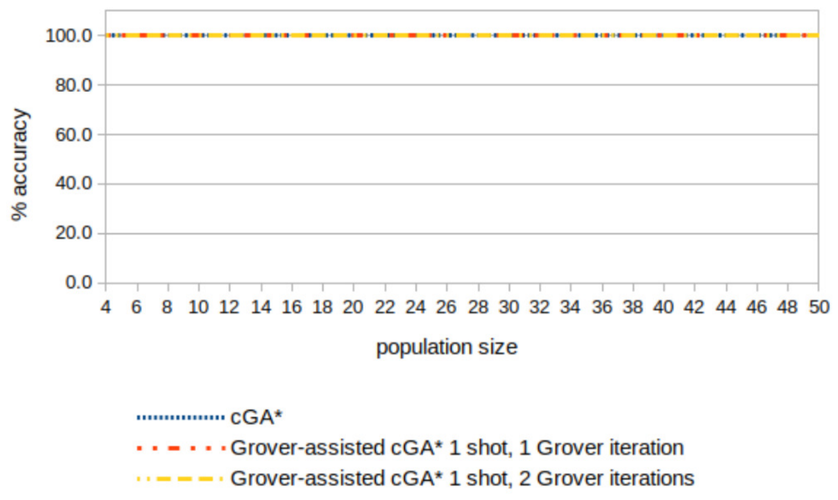


Fig.9: *Comparison of the solution quality (number of correct bits in percentage at the end of the run) achieved by the cGA* and the Grover-assisted cGA* on TSP 3 cities.*

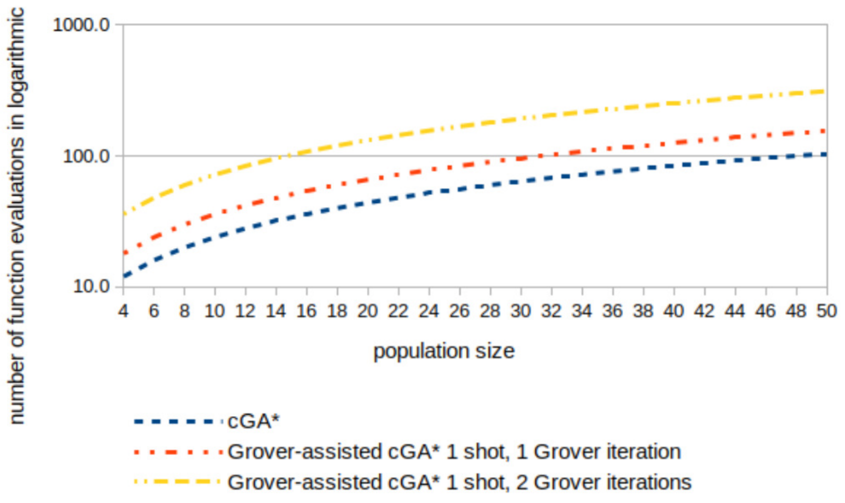


Fig.10: *Comparison of the cGA* and the Grover-assisted cGA* in the number of function evaluations needed to achieve convergence on TSP 3 cities.*

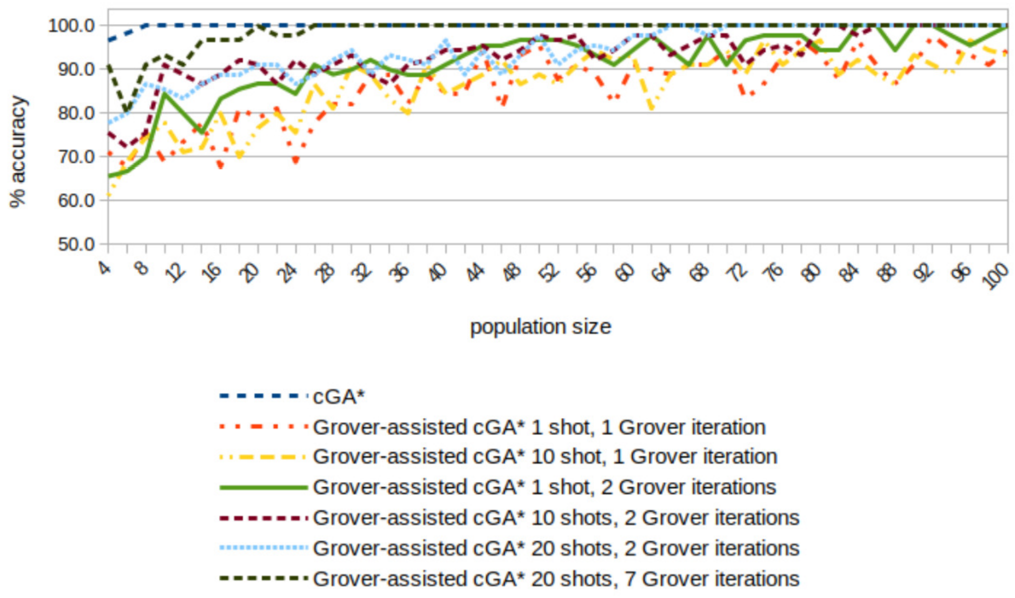


Fig.11: *Comparison of the solution quality (number of correct bits in percentage at the end of the run) achieved by the cGA* and the Grover-assisted cGA* on TSP 4 cities.*

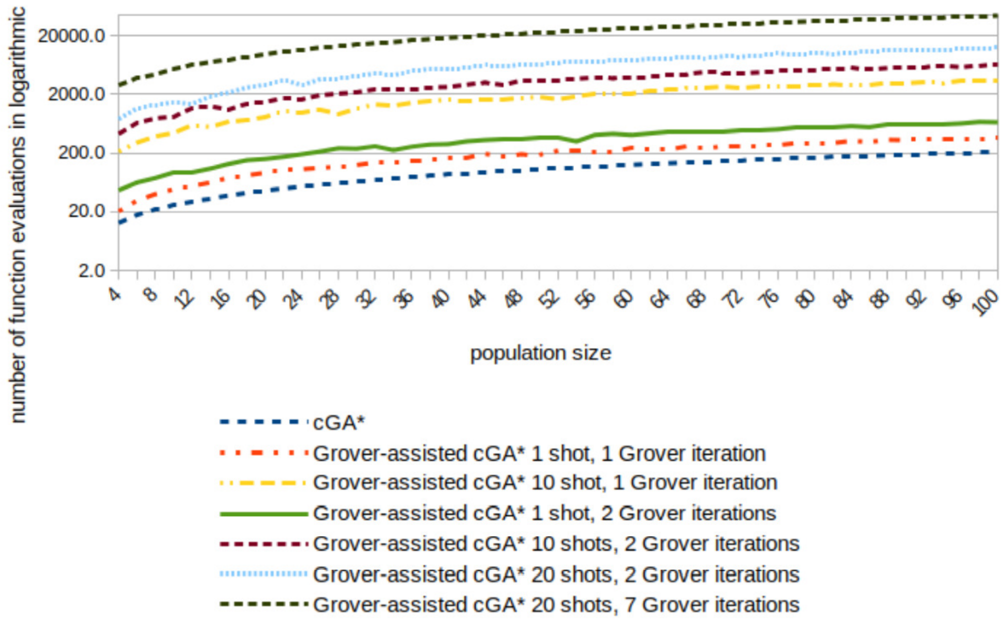


Fig.12: *Comparison of the cGA* and the Grover-assisted cGA* in the number of function evaluations (are plotted in logarithmic scale) needed to achieve convergence on TSP 4 cities.*

References

[1] D. E. Goldberg, D. E. Goldberg, G. Goldberg, David Edward, and V. A. P. o. H. D. E. Goldberg, *Genetic Algorithms in Search*, Optimization, and Machine Learning. Addison-Wesley.

[2] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling," p. 6.

[3] A. Rodionova, K. Antonov, A. Buzdalova, and C. Doerr, "Offspring Population Size Matters when Comparing Evolutionary Algorithms with Self-Adjusting Mutation Rates," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 855–863. [Online]. Available: `http://arxiv.org/abs/1904.08032`

[4] D. Mora-Melià, F. J. Martínez-Solano, P. L. Iglesias-Rey, and J. H. Gutierrez-Bahamondes, "Population Size Influence on the Efficiency of Evolutionary Algorithms to Design Water Networks," vol. 186, pp. 341–348. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1877705817313565`

[5] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. L. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," vol. 7, no. 3, pp. 231–253.

[6] C. W. Ahn, K. P. Kim, and R. S. Ramakrishna, "A memory-efficient elitist genetic algorithm," in *International Conference on Parallel Processing and Applied Mathematics*, Springer, 2003, pp. 552–559.

[7] U. F. Siddiqi, Y. Shiraishi, and S. M. Sait, "Memory-efficient genetic algorithm for path optimization in embedded systems,' *IPSJ Online Transactions*, vol. 6, pp. 28–36, 2013.

[8] C. T. Joseph, K. Chandrasekaran, and R. Cyriac, "Improving the efficiency of genetic algorithm approach to virtual machine allocation," in *2014 International Conference on Computer and Communication Technology (ICCCT). IEEE*, 2014, pp. 111–116.

[9] D. R. da S. Medeiros, M. F. Torquato, and M. A. Fernandes, "Embedded genetic algorithm for low-power, low-cost, and low-size-memory devices," *Engineering Reports*, vol. 2, no. 9, p. e12231, 2020.

[10] G. Harik, F. Lobo, and D. Goldberg, "The compact genetic algorithm," vol. 3, no. 4, pp. 287–297.

[11] 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`

[12] R. P. Feynman, "Simulating physics with computers," vol. 21, no. 6, pp. 467–488. [Online]. Available: `https://doi.org/10.1007/BF02650179`

[13] X. Ma, X. Yuan, Z. Cao, B. Qi, and Z. Zhang, "Quantum random number generation," vol. 2, no. 1, pp. 1–9. [Online]. Available: `https://www.nature.com/articles/npjqi201621`

[14] Z. Brakerski, P. Christiano, U. Mahadev, U. Vazirani, and T. Vidick, "A Cryptographic Test of Quantumness and Certifiable Randomness from a Single Quantum Device," [Online]. Available: urlhttp://arxiv.org/abs/1804.00640

[15] 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. Association for Computing Machinery*, pp. 212–219. [Online]. Available: `https://doi.org/10.1145/237814.237866`

[16] W. P. Baritompa, D. W. Bulger, and G. R. Wood, "Grover's quantum algorithm applied to global optimization," *SIAM Journal on Optimization*, vol. 15, no. 4, pp. 1170–1184, 2005.

[17] M. Furer, "Solving np-complete problems with quantum search," in *Latin American Symposium on Theoretical Informatics*, Springer, 2008, pp. 784–792.

[18] G. Sun, S. Su, and M. Xu, "Quantum algorithm for polynomial root finding problem," in *2014 Tenth International Conference on Computational Intelligence and Security*, IEEE, 2014, pp. 469–473.

[19] A. Kirke, "Application of grover's algorithm on the ibmqx4 quantum computer to rule-based algorithmic music composition," *CoRR*, 2019.

[20] R. Preston, "Applying grover's algorithm to hash functions: A software perspective," *arXiv preprint arXiv:2202.10982*, 2022.

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

[22] 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.

[23] 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`

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

[25] 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

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

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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

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

[34] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," [Online]. Available: http://arxiv.org/abs/1411.4028

[35] A. N. Sloss and S. Gustafson, "2019 Evolutionary Algorithms Review," in *Genetic Programming Theory and Practice XVII*, ser. Genetic and Evolutionary Computation, W. Banzhaf, E. Goodman, L. Sheneman, L. Trujillo, and B. Worzel, Eds. Springer International Publishing, pp. 307–344. [Online]. Available: https://doi.org/10.1007/978-3-030-39958-016

[36] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms,* USA:Oxford University Press, Inc., 1996.

[37] 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.

[38] G. Huang and L. Wang, "Hybrid Neural Network Models for Hydrologic Time Series Forecasting Based on Genetic Algorithm," in *2011 Fourth International Joint Conference on Computational Sciences and Optimization*, 2011, pp. 1347–1350.

[39] M. Musnjak and M. Golub, "Using a set of elite individuals in a genetic algorithm," in *26th International Conference on Information Technology Interfaces 2004*, vol. 1, pp. 531–535, 2004.

[40] L. Gaxiola, J. Tapia, and V. Diaz-Ramirez, "Parallel Genetic Algorithms on a GPU to Solve the Travelling Salesman Problem," 2014.

[41] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. Cambridge University Press.

[42] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum Amplitude Amplification and Estimation," [Online]. Available: http://arxiv.org/abs/quant-ph/0005055

[43] M. Boyer, G. Brassard, P. Hoeyer, and A. Tapp, "Tight bounds on quantum searching," vol. 46, no. 4-5, pp. 493–505. [Online]. Available: http://arxiv.org/abs/quant-ph/9605034

[44] S. Aaronson and A. Ambainis, "The Need for Structure in Quantum Speedups," [Online]. Available: http://arxiv.org/abs/0911.0996

[45] A. Bärtschi and S. Eidenbenz, "Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 72–82.

[46] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, "Automated optimization of large quantum circuits with continuous parameters," *npj Quantum Information*, vol. 4, no. 1, p. 23, May 2018. [Online]. Available: https://doi.org/10.1038/s41534-018-0072-4

[47] E. Muñoz-Coreas and H. Thapliyal, "T-count and Qubit Optimized Quantum Circuit Design of the Non-Restoring Square Root Algorithm," dec 2017. [Online]. Available: http://arxiv.org/abs/1712.08254

[48] C. S. Cheng, A. K. Singh, and L. Gopal, "Efficient Three Variables Reversible Logic Synthesis Using Mixed-polarity Toffoli Gate," *Procedia Computer Science*, vol. 70, pp. 362–368, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915031993

[49] T. Morimae, K. Fujii, and H. Nishimura, "Power of one non-clean qubit," oct 2016. [Online]. Available: https://arxiv.org/abs/1610.07244

[50] J. Preskill, "Quantum Computing in the NISQ era and beyond," jan 2018. [Online]. Available: http://arxiv.org/abs/1801.00862http://dx.doi.org/10.22331/q-2018-08-06-79

[51] M. McEwen, L. Faoro, K. Arya, A. Dunsworth, T. Huang, S. Kim, B. Burkett, A. Fowler, F.

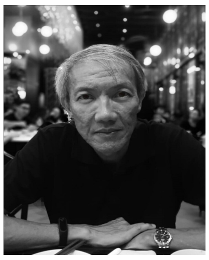
Arute, J. C. Bardin, A. Bengtsson, A. Bilmes, B. B. Buckley, N. Bushnell, Z. Chen, R. Collins, S. Demura, A. R. Derk, C. Erickson, M. Giustina, S. D. Harrington, S. Hong, E. Jeffrey, J. Kelly, P. V. Klimov, F. Kostritsa, P. Laptev, A. Locharla, X. Mi, K. C. Miao, S. Montazeri, J. Mutus, O. Naaman, M. Neeley, C. Neill, A. Opremcak, C. Quintana, N. Redd, P. Roushan, D. Sank, K. J. Satzinger, V. Shvarts, T. White, Z. J. Yao, P. Yeh, J. Yoo, Y. Chen, V. Smelyanskiy, J. M. Martinis, H. Neven, A. Megrant, L. Ioffe, and R. Barends, "Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits," vol. 18, no. 1, pp. 107–111. [Online]. Available: https://www.nature.com/articles/s41567-021-01432-8

[52] C. Durr and P. Hoyer, "A Quantum Algorithm for Finding the Minimum," [Online]. Available: http://arxiv.org/abs/quant-ph/9607014

[53] S. Forrest and M. Mitchell, "Relative Building-Block Fitness and the Building-Block Hypothesis," in *Foundations of Genetic Algorithms*, ser. Foundations of Genetic Algorithms, L. D. Whitley, Ed. Elsevier, vol. 2, pp. 109–126. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080948324500131

Kamonluk Suksen received her bachelor's degree, Master's Degree, and Ph. D. in Computer Engineering from Chulalongkorn University in 2013, 2014, and 2022, respectively. After graduation, she had been employed in the position of Head of Software & Information Technology by Yannix Co., Ltd. until 2020. Currently, she is pursuing a research assistant in Computer Engineering at Chulalongkorn University. Her research is heavily focused on quantum computing and quantum algorithms for optimization problems.

Naphan Benchasattabuse is a Ph.D. candidate in Graduate School of Media and Governance at Keio University. He received M.Eng. from Chulalongkorn University with his thesis on optimizing circuit-based quantum programs for arithmetic circuits. His experience in optimizing quantum circuits got him first place twice in IBM Quantum Challenge for 2019 and Fall 2021. His research interests include protocol design and resource management inside quantum repeater networks, the compilation of quantum programs, and quantum algorithms for optimization problems.



Prabhas Chongstitvatana is a professor in the department of computer engineering, Chulalongkorn University. He earned B.Eng. in Electrical Engineering from Kasetsart University, Thailand in 1980 and Ph.D. from the department of artificial intelligence, Edinburgh University, U.K. in 1992. His research involves robotics, evolutionary computation, computer architecture, bioinformatics and quantum computing. He is a lifetime member of Thailand Engineering Institute, senior member of Thai Academy of Science and Technology, senior advisor of Thai Robotics Society, founding member of ECTI Association and founding member of Thai Embedded System Association. He has been awarded "National Distinguished Researcher" by National Research Council of Thailand in 2009. His current interest is in building a Quantum computer.