Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Highlights

Size-efficient sparse population for strictly structured quantum genetic algorithm

Future Generation Computer Systems xxx (xxxx) xxx

Jun Suk Kim, Chang Wook Ahn*

- This paper introduces a semi-classical quantum algorithm to achieve optimization.
- This paper then proposes a method of improvement with a modified population setup.
- The experiment results show that the proposed method is scalable and practicable.

Please cite this article as: J.S. Kim and C.W. Ahn, Size-efficient sparse population for strictly structured quantum genetic algorithm, Future Generation Computer Systems (2022), https://doi.org/10.1016/j.future.2022.04.030.

Future Generation Computer Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Size-efficient sparse population for strictly structured quantum genetic algorithm

Jun Suk Kim, Chang Wook Ahn *

Gwangju Institute of Science and Technology, Gwangju, Republic of Korea

ARTICLE INFO

Article history:
Received 13 August 2021
Received in revised form 15 April 2022
Accepted 25 April 2022
Available online xxxx

Keywords:
Quantum genetic algorithm
Evolutionary computation
Optimization
Metaheuristics
Grover's algorithm
Population setup

ABSTRACT

Quantum genetic algorithm is a field of research to discover a potential structure to realize an effective heuristic, evolutionary optimization technique powered by quantum computation. Apart from contemporary efforts to look for a novel quantum evolutionary design, some studies suggest the idea of *strictly structured* quantum genetic algorithm, which rigorously imitates the classical practice via a sparse population to sample a few individuals from the given problem's domain, as opposed to the conventional quantum practice that transforms the entire domain into a population itself. Albeit having its own advantages, the algorithm requires to periodically measure and reinitialize the quantum system over generations. It therefore leads to several computational inefficiency issues including the wasteful population initialization, during which individuals that rather marginally contribute to the overall optimization are iteratively generated. This paper proposes that the algorithmic inefficiency from the mentioned problem can be alleviated by preemptively eliminating a large portion of undesirable individuals and still maintain the tasked optimization result to an acceptable degree. The main idea is to deliberately reduce the amount of randomness among the quantum population, thereby discarding individuals that do not contain any fitted genes. A number of tests were conducted on continuous optimization problems to validate the theory of the proposed method, resulting that it can effectively reduce the size of the involved population while minimizing the loss in the original algorithm's performance.

1. Introduction

Quantum computing has already become a crucial topic in computer science thanks to the recent events that occurred in the span of two decades, including the successful construction of real quantum processor prototypes, advancement of the sophisticated quantum algorithms, and overloaded needs from industries for supremely efficient optimization technologies [1]. Though imperative, however, effective quantum algorithms are difficult to invent mainly because quantum computing's counterintuitive nature often hinders much of efforts to grasp a solid picture of its peculiar mechanism [2]. It therefore became one of the most popular strategies so far to create a quantum version of an existing classical algorithm with its overall structure inspired by the original design. Early attempts on this strategy made an important progress, resulting in the birth of quantum machine learning and quantum genetic algorithm [3].

Quantum genetic algorithm (QGA), like its contemporary classical or conventional genetic algorithm (CGA) [4], mimics the natural evolutionary processes to accomplish an optimization task. One major flaw of CGA is that it is computationally expensive [5] due to the repetitive access to every bit of input data, and it is a widespread expectation that the quantum-enabled parallel computation can help lift such a burden to a considerable degree. In spite of several expectations that organizing a series of genetic operators in a similar form to those in the classical version could result an intermediately efficient quantum algorithm, some intrinsic differences between quantum and classical computing systems seem to require an introduction of unprecedented approaches in order to achieve further improvements [6]. For example, a process of redistributing the small number of components in each individual data fragment, or chromosome, is deemed an essential step that diversifies candidate solutions in a conventional genetic algorithm, [7] and it inevitably requires specifying the address of each target component. This could be a tricky part to implement in a QGA, where any interaction between a quantum system and its surroundings can cause a physical phenomenon called quantum wave function collapse, which destroys nearly every sort of information stored within it [8].

Continuous efforts to overcome the quantum-oriented discrepancies under the conventional evolutionary structure have

* Corresponding author.
E-mail address: cwan@gist.ac.kr (C.W. Ahn).

https://doi.org/10.1016/j.future.2022.04.030
0167-739X/

led to somewhat loose establishment of *reduced quantum genetic algorithm* (RQGA) [9], although that specific term has been referred to by only a few. Concisely speaking, RQGA avoids using the classical genetic operators by adopting single modified selection operator, and such simplification is compensated by exploiting a quantum selection algorithm that can specify target elements about quadratically faster than any known classical selection algorithms. In spite of a number of arguments in details, RQGA provides a theoretical basis that many current researches in QGA begin with.

In 2014, an unique approach to implement a QGA was proposed [10]. Instead of introducing a comprehensive quantum selection operator, the authors went back to the older idea of reproducing the classical genetic operators in a form that suits the quantum rules. The most noteworthy part is the cleverly configured population generator, which prepares 2^c individuals in a length of n, conditioned $c \ll n$. This deliberately replicates the population sparsity of CGA, which extracts only a portion of the target problem's domain to use as its initial population. This work independently labels their work *strictly structured quantum genetic algorithm* (SS-QGA) with regards to RQGA, since, we would argue, it rather strictly resembles the CGA's algorithmic procedure.

SS-QGA manifests its own advantages by counting fewer individuals during selection, as opposed to RQGA that considers nearly every possible combination of its chromosomes. In order to enhance SS-QGA's computational efficiency regarding population sparsity even further, a feasible improvement is introduced here to reduce the algorithm's cost by eliminating the redundant part of its population that barely contributes positively to the overall optimization convergence. In the next section, the theories of RQGA and SS-QGA with emphasis on their differences are reviewed along with the related work. The third section mathematically shows that the initial population size for the original SS-QGA can be contracted dramatically, while its optimization performance remains preserved. In the fourth section, the results from the experiments, programmed and conducted via IBM Qiskit [11] and Python, to validate the proposed method on the problems with continuous domains are introduced. The last sections conclude the study with analyses of the result in depth.

2. Quantum genetic algorithm: Overview

In this section, we briefly introduce the basics of quantum computing and genetic algorithm and then discuss the main features of the strictly structured quantum genetic algorithm. The abstract ideas of reduced quantum genetic algorithm are then explained for comparison and referencing purposes, followed by the introduction of the related work of the field. For readers who want a comprehensive look, it is recommended to see [9, 10,12] that thoroughly describe them in details. The quantum background equations present in this section are borrowed and reorganized from [2,13].

2.1. Background

Quantum computers use *qubit* as a basic computation unit. It is a quantum analog to bit in a conventional computer, but its unique, distinguished features allow quantum machines to carry out tasks in a different manner. Quantum computing inherits several laws from quantum mechanics to shape its main characteristics, three of which come to a primary concern when a quantum algorithm is designed. First, any computation inside a quantum system can be written with notions of linear algebra, involving complex numbers, under a Hilbert space [2]. Second, quantum computation must be performed in a physically isolated, uninteractive system in order to pursue a desired result, which can only be observed via an irreversible operation of *measurement*. Third, any quantum operation except for measurement must be reversible, i.e. the input of an algorithm must be tractable from the output and assessed operators [13].

n qubits can be represented as their quantum state $|\Psi\rangle$, which is written in the form of a 2^n-dimensional vector. A single-qubit state, for example, can be represented as follows:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{bmatrix}1\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix} \tag{1}$$

A *ket* $|v\rangle$ is a mathematical notation of a column vector in a vector space, and in quantum computing each ket itself is a quantum state. In other words, $|\psi\rangle$ in (1) is a probabilistically combined state of the states $|0\rangle$ and $|1\rangle$, with α and β being the respective state's probability density, which dictates each state's probability to be observed after measurement. Since the sum of probabilities of all states within a certain system must be 1, α and β obey $\sqrt{\alpha}+\sqrt{\beta} = 1$ and can be complex numbers. The fact that quantum computation is performed in a Hilbert space demands the ket vectors to be orthonormal to each other [2]. Eq. (1) is written based on the $Z-basis$, or 0–1 computational basis, which is a conventional choice to write a quantum state with due to its similarity to classical bits in manifesting the states under the binary notation of 0 and 1. Under the computational basis, $|0\rangle$ represents a vector $(1, 0)^T$, and $|1\rangle$ represents a vector $(0, 1)^T$. In addition, it is a common practice to initialize every qubit as $|0\rangle$ at the beginning of computation.

A quantum operator, represented as a quantum gate, applicable to states made from n qubits is represented by a $2^n \times 2^n$ matrix. Due to their reversibility, all the quantum gates are unitary, and many of them are also Hermitian [13]. One of the most important and frequently used quantum gates is the *Hadamard* operator H, which places a quantum state into uniform *superposition*. For example, applying the Hadamard operator to a quantum state $|0\rangle$ results

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \tag{2}$$

Note that now both $|0\rangle$ and $|1\rangle$ share a probability of $(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ i.e. 50% chance to be measured. Under superposition, multiple different states have their corresponding probabilities to exist, and applying quantum operators to such states will achieve parallel computation [13], i.e. the identical operation is applied to every embedded state simultaneously. It then becomes a matter of measuring them as effectively as possible, since measuring a quantum state causes *wave function collapse*, which reduces the superposed quantum states to only one of them accordingly with their probabilities.

One other notable exploitation of quantum mechanics in quantum computing is *entanglement*, which mutually ties two (or more) states such that measurement upon one state directly and immediately affects the other. A theoretical example of entanglement is the Bell states, one of which writes

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{3}$$

and can be implemented by applying a Controlled-NOT (*CNOT*) gate to a 2-qubit superposed state [9]. Note that the state only consists of two probabilistically measurable states, instead of four. In terms of measurement, the equation indicates that if the first qubit has been measured as 0, then the second qubit is destined to be measured as 0 as well, and the same goes for the case of 1. Such a state is said to be entangled, and the technique of entanglement is frequently utilized along with superposition to enhance the effectiveness of quantum algorithms [14].

The potential of the quantum parallel computation discussed above has raised the anticipation of vast applications to diverse fields of optimization strategy, including genetic algorithms. Genetic algorithm (GA), also referred to as CGA in this paper, is one of the most prominent meta-heuristic approaches to solving large-scale optimization problems, by implementing the abstract replication of the evolutionary steps inspired by the actual phenomenon in nature [7]. It starts with creating a set of population, mapping a fragment of the given problem's search space (domain), that consists of multiple individuals, which are then processed with genetic operators, such as crossover and mutation, to perform the necessary perturbation of genetic values stored within them. Throughout multiple generations, individuals with sufficiently high fitness values are repetitively selected, resulting in the heuristic discovery of the optimal or nearly optimal solutions at the end of the algorithm.

The field of quantum genetic algorithm, along with its variant quantum-*inspired* genetic algorithm, started with a notion that superposition of multiple quantum states can be exploited for producing a quantum version of genetic population. Suppose that for an arbitrary optimization problem, its domain has been encoded with n-bit binary chromosomes. For a quantum processor, the algorithm begins with initializing n qubits in 0:

$$|\Psi_1\rangle = |0\rangle_1 \otimes |0\rangle_2 \otimes |0\rangle_3 \otimes \cdots \otimes |0\rangle_n = |0_1 0_2 0_3 ... 0_n\rangle \tag{4}$$

Multiple qubits can be represented as single state with tensor products $\otimes$ such that

$$|0\rangle \otimes |1\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}0\\1\end{bmatrix}\\0\begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \tag{5}$$

Applying a Hadamard gate to the state in (4) results

$$|\Psi_2\rangle = H|000...0\rangle = \frac{1}{\sqrt{2^n}} \sum_{t=1}^{2^n} |x_0 x_1 ... x_{n-2} x_{n-1}\rangle, x \in \{0, 1\} \tag{6}$$

which is equal to

$$|\Psi_2\rangle = \frac{1}{\sqrt{2^n}} \begin{bmatrix}0\\0\\...\\0\end{bmatrix}_n + \frac{1}{\sqrt{2^n}} \begin{bmatrix}1\\0\\...\\0\end{bmatrix}_n + \cdots + \frac{1}{\sqrt{2^n}} \begin{bmatrix}1\\1\\...\\1\end{bmatrix}_n \tag{7}$$

There are now 2^n states in superposition configured from n qubits, each having a different combination of 0s and 1s. This is essentially a quantum binary population with 2^n chromosomes, and the task now is to find a state with the highest fitness value with respect to the given problem function. Most quantum genetic algorithm studies choose to use *Grover's search algorithm* [15] and its variant *Grover-BBHT* [16] to reproduce a selection operator. Effective ways to implement the quantum version of crossover and mutation operators remain disputed, although a few studies claim their robustness over others.

2.2. Reduced quantum genetic algorithm

Reduced quantum genetic algorithm (RQGA), as its name suggests, pursues a quantum counterpart for the classical evolutionary process in a simplified configuration. The idea is driven by two main causes. First, researchers highly anticipate that the exotic quantum computing is capable of providing novel means to achieve efficient heuristic optimization [17]. Second, allegedly, effective quantum analogs for some classical genetic operators, such as crossover, have not been developed yet [18]. RQGA is meant to overcome those problems by exploiting quantum algorithms that can work as a proxy for the classical operators. Algorithm 1 below shows the overall structure adopted by most studies on RQGA.

One of the important advantages of RQGA is that it generates the entire population possible, as a set of quantum states, out of single chromosome. Putting a n-dimensional state into superposition, it creates a population of 2^n superposed individuals, all sharing the same length n and initial probability amplitude, as described in the Eqs. (6) and (7). Desired solutions should already exist amongst the generated population, so the whole problem is reduced to specifying them via an effective selection method.

The selection operator in RQGA is entirely composed from a variant of Grover's selection algorithm [15], Grover-BBHT [16]. It marks the desired solution states with the Grover's *oracle* and amplifies their probability amplitudes to a degree at which they can be observed with a sufficiently high probability. Each step of marking and amplification is done by the rotation of the Grover's statevector, often referred to as the Grover's *diffuser* [11]. Grover-BBHT bypasses one of the original algorithm's prerequisites to know the number of solutions beforehand, by heuristically searching for them with a hyperparameter threshold. Although it requires measurement at the end of each diffuser rotation, Grover-BBHT sustains the same time complexity with its original design; given an unsorted table $T[0...N-1]$ with t solutions, the algorithm is expected to find them in queries of $O(\sqrt{\frac{N}{t}})$, which is regarded quadratically faster than any known classical search algorithms that would take $O(\frac{N}{t})$ in average [19]. The fact that Grover-BBHT can safely find solutions, even if their quantity is unknown, makes it a successful applicator to minimum or maximum search problems, where searching process is frequently done by iteratively raising a threshold bar above which candidate solutions in an unpredicted number are to be specified. Undoubtedly, its discovery was quickly followed by the introduction of the quantum minimum [20] and maximum [21] search algorithms that majorly exploit it. The procedure of Grover-BBHT is briefly described in Algorithm 2.

Algorithm 1 Reduced Quantum Genetic Algorithm

```
Initialize a population by superposition
Evaluate the fitness value for each chromosome
for iteration = 1, 2, ..., n do
    Make a query to the Grover-BBHT oracle O
    Rotate the Grover-BBHT diffuser G
end for
Measure the state
```

Algorithm 2 Grover-BBHT Algorithm

```
Set m = 1 and λ such that 1 < λ < 4/3
for iteration = 1, 2, ... do
    Randomly choose j such that 0 < j < m
    Initialize the items into equal superposition
    Apply Grover's algorithm for j times
    Measure the state
    if the outcome matches the desired feature then
        Exit the loop and terminate
    else
        Set m ← min(λm, √N)
    end if
end for
```

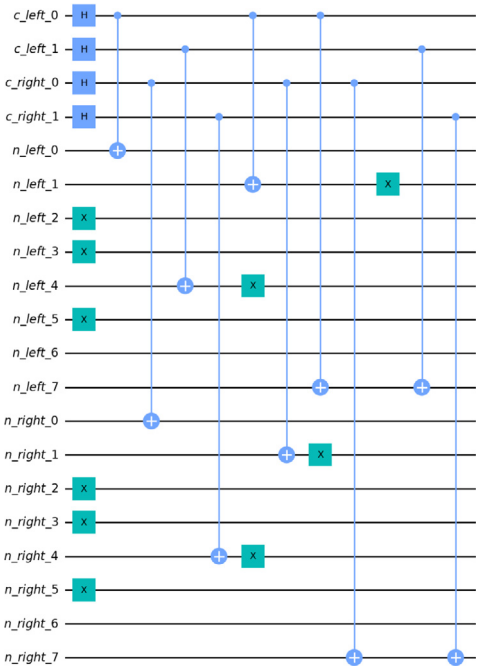

Fig. 1. An example of a SS-QGA pseudo-randomizer circuit with $c = 2$ and $n = 8$. Drawn via Qiskit.

2.3. Strictly structured quantum genetic algorithm

Strictly structured quantum genetic algorithm (SS-QGA) is a semi-classical quantum genetic algorithm, which keeps the structure of CGA but adopts quantum parallelism to alleviate the heavy cost from massive evolutionary computations. It is distinguished from quantum–classical hybrid algorithms because every sequence of data processing, once its encoding is done, is conducted under a quantum system. SS-QGA creates a sparse population that represents only a portion of the given problem's domain, unlike RQGA that takes the whole of it. The individuals are then rearranged via crossover and mutation until their fitness values are measured at the selection step, as summarized in Algorithm 3.

SS-QGA starts with initializing two copies of a population $|\Psi\rangle$ of 2^c individuals each comprising n genes.

$$|\Psi\rangle = \frac{1}{\sqrt{2^c}} \sum_{n=1}^{2^c} |x_0 \ldots x_{c-1}\rangle |x_0 \ldots x_{n-1}\rangle, x \in \{0, 1\} \tag{8}$$

This procedure is done by preparing two registers containing $2c$ qubits and $2n$ qubits respectively, constructing a pseudo-randomizer circuit (Fig. 1) which randomly connects the c and n qubits via *CNOT* gates after superposition, and running the randomizer twice. Under a condition $c \ll n$, the overall size of the population is much smaller than one in RQGA, which would be 2^n. This necessitates multiple iterations of the selection operator yet dramatically reduces the required number of Grover iterations within each operation [10]. Notice that although the population is described conceptually as *two* distinct sets, they are actually generated by superposition in single circuit. Therefore, a state of the whole circuit can be written as a probabilistic sum of 2^{2c} sub-states, each having $2c + 2n$ digits. This automatically completes crossover, since each individual state in one set is conjoined with every individual state in another due to the applied superposition [13]. One can relabel these states for convenience, but it is an extra step after all.

Algorithm 3 Strictly Structured Quantum Genetic Algorithm

Set integers c and n such that $c \ll n$
Set a sufficiently large fitness threshold T with respect to the target function f
for *generation* $= 1, 2, \ldots$ **do**
  if *generation* $= 1$ **then**
    set $z = 0$
  end if
  Build a *pseudo-randomizer* circuit R accordingly with the genetic configuration of z
  Call R twice to generate two population sets of random 2^c individual states each in a length of n
  Perform crossover, which is basically relabeling the states previously created
  Perform mutation
  Apply Grover-BBHT algorithm upon the states and measure the highest fitted individual u
  if $f(u) > f(z)$ **then**
    Set $u = z$
  end if
  if $f(z) > T$ **then**
    break
  end if
end for

SS-QGA runs the same selection process U_{sel} as RQGA does, except for two changes. First, SS-QGA selects candidate solutions from a tailored population with a size of 2^{2c}, much smaller than 2^{2n}. Second, unlike in RQGA that requires only superposition, in SS-QGA the population initializer U_{init} (and the mutation operator U_{mut}) must be initiated once for each iteration of Grover-BBHT, i.e.

$$U_{sel} = U_{mut} U_{init} |\Psi'\rangle U_{init}^{\dagger} U_{mut}^{\dagger} \tag{9}$$

where $|\Psi'\rangle$ indicates the population after applying the Grover's oracle. Note that each operator is applied again as its conjugate transpose in order to deconstruct, or decouple, the interrelation among the qubits so that proper measurement can be made [10]. Because Grover-BBHT ends every iteration with measurement that breaks the whole population into single individual state, repetitively applying the operators is mandatory so that the population can be reinstated for subsequent runs. This slightly complexifies the circuit in terms of its size because every Grover iteration now includes U_{init} and U_{mut}.

A population in SS-QGA consists of nonconsecutive, sparse binary string chromosomes, which raise the issue of ambiguous indexing. Suppose that out of N possible individuals in total, $\tilde{N}$ individuals form a population for one arbitrary generation such that $\tilde{N} \cong N$. It is likely here that any index can well approximate the individual it corresponds to, since $\tilde{N}$ and N are nearly equal in size. Therefore, each individual state $|x\rangle$ can approximately represent its own index x, so any extra steps to convert the state to its index are not needed during the process of fitness evaluation. In the case of SS-QGA, on the other hand, it is hardly convincing that individuals can represent their indices because, now with $\tilde{N} \ll N$, some individuals can only be represented with numbers that are larger than the maximum available index. The algorithm then needs either an extra step to convert each index to its individual or somehow making each individual represent its own index. SS-QGA chooses the second path and resolves the issue by creating the pseudo-randomizer.

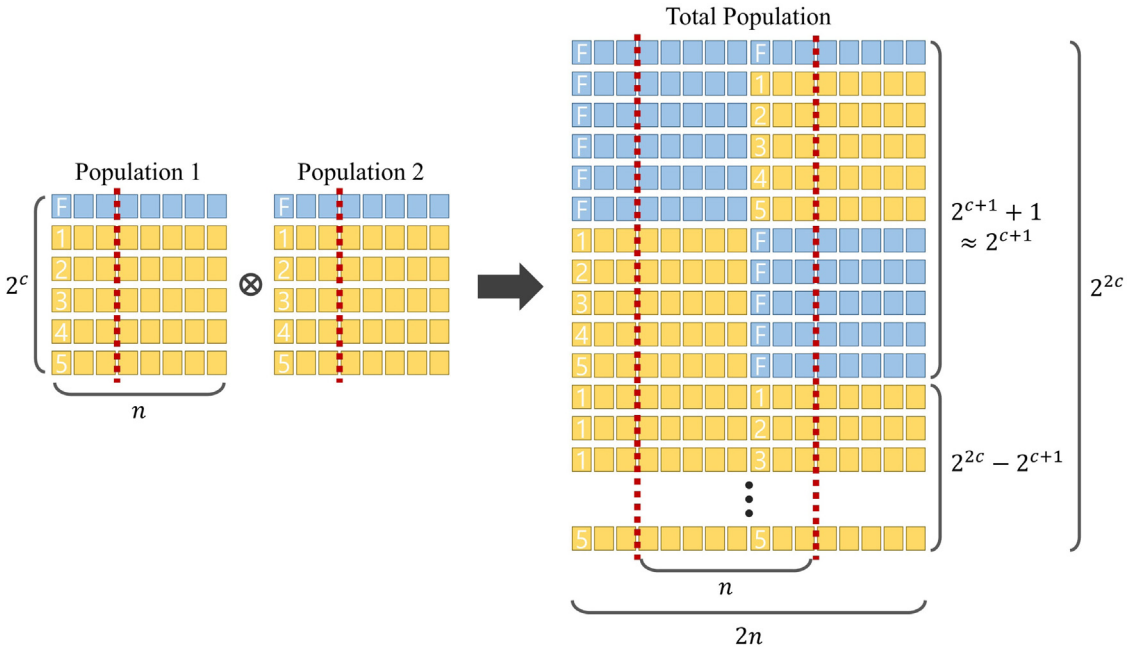

Fig. 2. A total population formed with two identical population sets. The blue chromosomes marked with the letter *F* are the inherited, high fitness individuals, and the yellow chromosomes are the randomly generated individuals. The red dotted bar lines mark the location of single crossover points.

2.4. Related work

GA is still an active field of research and discussions, and its applications are extensively made in various areas of computer science. Recently, for example, Akimoto proposed a GA technique that can effectively replace generative adversarial network (GAN) in solving the problem of min–max optimization by exploiting a minimization oracle [22]. In addition, Chiesa et al. introduced the utilization of GA for identifying and selecting features from high-dimensional datasets, which are a crucial step in general machine learning [23]. Overall, there are many reasons to make use of GA in various areas of research, as Katoch et al. in their review paper conclude [24].

In terms of the amount of published studies, RQGA has dominated the field of QGA. In fact, the first study to attentively analyze the concept of QGA by Han et al. in 2002 describes the potential algorithmic structure to resemble RQGA [3]. The idea was expended by Udrescu et al. who provided a more specific procedural description of implementing RQGA via Grover's search algorithm [12], which was soon polished further by Malossini et al. in 2008 [6]. The history of SS-QGA is relatively short. While its idea has existed, though rather obscurely, since the beginning stage of QGA, it was in 2014 when Saitoh et al. first proposed an algorithmic structure that is rigorous enough to provide the detailed procedure and to enable its simulation [10].

Despite its short history, the field of QGA overall has also seen a couple of review studies for it. For example, Laboudi et al. published a study that thoroughly compares CGA and QGA in terms of optimization [5]. In 2016, Lahoz-Beltra published a general introduction to QGA in the perspective of computer science and information technology [9].

3. Proposed improvement

In this section, we first point out the lurking problem of the current SS-QGA and then suggest a theory to solve it and to improve the overall performance of the algorithm.

3.1. Excess in population

In the previous section it is mentioned that SS-QGA prepares two copies of an identical population $|P\rangle$ as $|P_1\rangle$ and $|P_2\rangle$, mainly for the purpose of performing crossover. The whole system's state $|\Psi\rangle$ can be written as

$$\begin{aligned}|\Psi\rangle &= |P_1\rangle \otimes |P_2\rangle \\ &= \frac{1}{2^c}\sum_{n=1}^{2^c}\sum_{n=1}^{2^c}|a_1a_2\rangle|x_1x_2\rangle\end{aligned} \tag{10}$$

where a is the address to its corresponding individual x. SS-QGA performs single-point crossover, in which an individual is replaced with a concatenation of two shorter chromosomes, inherited from different population sets. Recall that the configuration of each individual as a binary string is represented by qubits from the $n-qubit$ registers, but they are connected by CNOT gates with superposed qubits in the $c-qubit$ registers in order to form a population. Therefore, the total number of individuals in the circuit is 2^{2c}, with every individual state formed as a part of an individual from one set added by a part of an individual from another (Fig. 2). To reveal its genetic features more clearly Equation (10) can be rewritten as

$$\frac{1}{2^c}\sum_{n=1}^{2^c}\sum_{n=1}^{2^c}|a_1a_2\rangle|x_1^{left}x_2^{right}\rangle_{ind}|x_2^{left}x_1^{right}\rangle_{rest} \tag{11}$$

The state $|x_1^{left}x_2^{right}\rangle_{ind}$, in the length of n, represents a group of individuals to be processed. With a hyperparameter crossover point placed between the chromosome's indices $h-1$ and h, each individual is formed with a left-hand part of an individual in the length of h and a right-hand part of the other in the length of $n-h$, while the remnant $|x_1x_2\rangle_{rest}$ is disregarded for the later processing. Note that it is only the expression that has changed from Eqs.(10) to (11); the actual quantum state expressed by them remains unaltered.

As described in Algorithm 3, SS-QGA reserves precisely one individual with a considerable fitness value from each generation, due to the required measurement during the Grover-BBHT

selection. At the initializing stage of a subsequent generation, a new population is formed with this single particular individual plus fully random individuals generated by a pseudo-randomizer. Within two identical population sets, some individuals contain partial genetic features from the reserved individual, and as a result these partially fitted individuals are more susceptible of higher fitness values. Individuals purely filled with random genes, however, lose their relative proclivity to higher fitness values along a series of continuing generations as the partially fitted individuals gradually become more sophisticated, i.e. resembling more closely to the potential maximum, which is hard to imitate by trivially generating random genes successively. Alternatively speaking, the degree of contribution by the random individuals to the evolutionary effort to search for the maximum remains highly limited, especially when a given problem is sufficiently complicated and requires a long series of generations to accomplish an acceptable solution.

In a SS-QGA circuit with two sets of a 2^c sized population that contain 2^{2c} individuals in total, the single high fitness individual in each set is superposed and joined with every individual on the other set, forming $2 \times 2^c = 2^{c+1}$ partially fitted individual substates. The rest, $2^{2c} - 2^{c+1}$ individuals are then entirely filled with randomly generated genes. Let $N = 2^c$. It is not hard to find $p(N)$ and $q(N)$ that satisfy

$$p(N) \leq \frac{N^2 - N = 2^{2c} - 2^c}{N = 2^c} \leq q(N) \tag{12}$$

Therefore, the population contains a polynomially larger portion of the random individuals than the partially fitted individuals in terms of N, and such difference is exponentially commensurate with the number of exploited qubits c. Judging by their low adequacy to the optimization task, it is reasonable to argue that the presence of the mentioned randomly generated individuals unnecessarily and severely increases the amount of computational cost needed for the selection process. Removing the rather unimportant chunk of population would be likely to speed up the overall algorithm, with loss in its optimizing capability kept at the minimum.

3.2. Removing trivial individuals

Now that the problem to concern has been explained, it is described here how it can be resolved, with more details on SS-QGA's quantum circuit and its logic. The goal is to remove all the fully randomly generated individuals, which do not share any genetic features with the high fitness individuals inherited from previous generations. The size of a population in SS-QGA is directly and solely related to the number of qubits in the $c-qubit$ register, c, and superposition applied to every qubit in the $c-qubit$ register inevitably generates the chunk of randomly generated individual chromosomes. Therefore, the primary goal should be to properly diminish the size of the population that the circuit exploits while guaranteeing that the high fitness genetic features remain preserved among the survivors.

We have shown in the previous section that the total number of individuals to either partially or entirely share the high fitness genetic features is $2^{c+1}+1 \approx 2^{c+1}$, which is still dependent on the variable c. A modified SS-QGA circuit thus should start with the different number of the $c-register$ qubits, this time being c, not $2c$. With every c qubit under superposition, iterating the circuit twice with different pseudo-randomizer circuits counts up to 2^{c+1} individuals in total into selection. Unlike the original algorithm that randomly chooses among n qubits to connect with c qubits, the circuit now divides the $n-qubit$ register into $preserved$ and $unpreserved$ qubits. Preserved qubits are applied with X gates to adjust their initially all-0 gene values to those of the inherited

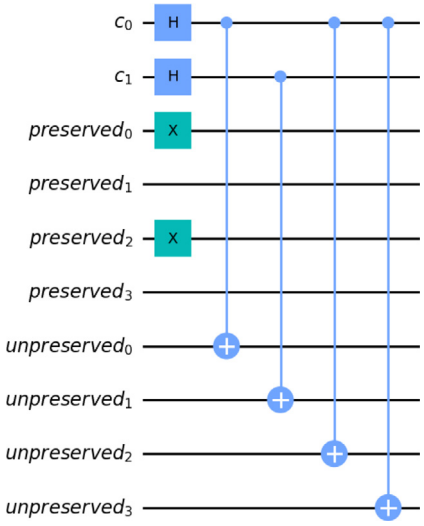

Fig. 3. An example of a modified SS-QGA pseudo-randomizer circuit with $c = 2$ and $n = 8$. Note that the c qubits do not interact with the n qubits marked as *preserved*. Drawn via Qiskit.

individual, representing its preserved genetic features. They are not applied with Hadamard operators or connected with c qubits because their binary gene values must be the same for every state of each population set so that multiple individuals to share the identical partial high fitness genetic features can be created. The rest of the genes are connected with c qubits to be initialized randomly (Fig. 3). Refer to Algorithm 4 below for the modified procedure.

Each $n-qubit$ register now contains a different portion of the preserved genes, indicating that the two registers cannot be processed concurrently for selection. Suppose that, like in the original algorithm, the $n-qubit$ registers are placed in parallel and processed with two Grover iterators independently. One Grover iterator for the first register will amplify the probability amplitude of an individual state in that register with the highest fitness value, but it will also raise the probability amplitudes of the states in the second register that are bound with the selected state in the first register. For example, let two $n-qubit$ registers start selection with a hypothetical state

$$|\psi_{hyp}\rangle = \frac{1}{2}(|0010\rangle + |0111\rangle + |1000\rangle + |0101\rangle). \tag{13}$$

The first $n-qubit$ register owns the first two qubits, and the second the rest. For simplicity, let the oracle of the first register's Grover iterator marks the state $|01\rangle$. The selection process will then also amplify the probability amplitude of the states $|11\rangle$ and $|01\rangle$ from the second register, since they are entangled with $|01\rangle$ from the first register. This amplification will be canceled out, however, if the Grover iterator on the second register happens to choose neither of those two. Therefore, the two $n-qubit$ registers must be processed separately, either with physically distant pseudo-randomizers, or in series.

Fig. 4 shows the overflow graphic of the modified SS-QGA per generation. Note that, in order to secure population diversity in the early stage, the algorithm runs the first generation with the exactly same steps of the original SS-QGA, creating 2^{2c} individuals and preserving single locally highest fitted individual z at the end. From the second generation on, the algorithm then follows the proposed modification, creating 2^{c+1} individuals, none purely random. The selection process is proceeded in two parts in series, each searching for the locally highest fitted individual from either of the two $n-qubit$ registers. At the end of a generation, those

Algorithm 4 Modified Strictly Structured Quantum Genetic Algorithm

Set integers c and n such that $c \ll n$
Set a sufficiently large fitness threshold T with respect to the target function f
for *generation* $= 1, 2, \ldots$ **do**
  if *generation* $= 1$ **then**
    set $z = 0$
    Build a *pseudo-randomizer* circuit R accordingly with the genetic configuration of z
    Call R twice to generate two population sets of random 2^c individual states each in a length of n
    Perform crossover, which is basically relabeling the states previously created
    Perform mutation
    Apply Grover-BBHT algorithm upon the states and measure the highest fitted individual u
    if $f(u) > f(z)$ **then**
      Set $u = z$
    end if
  else
    Build two modified pseudo-randomizer circuits R_1' and R_2', each connecting $c - qubits$ with only a part of $n - qubits$ and applying X gates to the rest, accordingly with the genetic configuration of z
    Call R_1' to generate single population set of partially random, partially fit c individual states each in a length of n
    Perform mutation to R_1'
    Apply Grover-BBHT algorithm upon the states and measure the highest fitted individual u_1
    Call R_2' to generate single population set of partially random, partially fit c individual states each in a length of n
    Perform mutation to R_2'
    Apply Grover-BBHT algorithm upon the states and measure the highest fitted individual u_2
    Compare $f(u_1)$, $f(u_2)$, and $f(z)$, and set the individual with the highest fitness value as z
  end if
  if $f(z) > T$ **then**
    break
  end if
end for

two locally highest fitted individuals and the highest individual from the last generation z are compared, returning a new z that contains the highest fitness value among them. This process alone would not cause any decrease in the best fitness value threshold because z is always accounted for in each generation, although its value could still be hindered by mutation.

The particular study on the Grover-BBHT-based minimum search algorithm proves mathematically that it at most takes the total time of $\frac{45}{4}\sqrt{N} + \frac{7}{10}\lg^2 N$, given a list of N items, to find the minimum with probability at least $\frac{1}{2}$ [20]. To guarantee that the algorithm almost certainly picks the minimum, the number of Grover iterations G_{iter} can be multiplied with an arbitrary constant integer. For example, the authors of the original SS-QGA claims its required number of iterations to be $G_{iter} = \eta\lceil * \rceil \frac{45}{4}\tilde{N} + \frac{28}{5}(\log_2 \tilde{N})^2$ with a constant integer $\eta \geq 1$. Note that the total number of individuals for each generation in SS-QGA is $2^{2c} = \tilde{N}^2$, so the expression is a polynomial of $\tilde{N}$ and an exponential function of c. The modified SS-QGA we have introduced instead takes 2^{c+1} individuals for each generation. Because each population set is processed for selection in series, the number of required Grover iterations for the proposed modification is $G'_{iter} = \eta\lceil * \rceil \frac{45}{4}(2\sqrt{\tilde{N}}) + \frac{28}{5}(\log_2(2\sqrt{\tilde{N}}))^2$, which is now a polynomial of $\sqrt{\tilde{N}}$ and an exponential function of $\frac{c}{2}$. The classical comparison made at the very end of each generation is negligible, since it only requires a small constant time complexity. We argue that this is quadratically faster than the original SS-QGA in terms of the number of each generation's individuals $\tilde{N}$ to be processed with the Grover selection procedure.

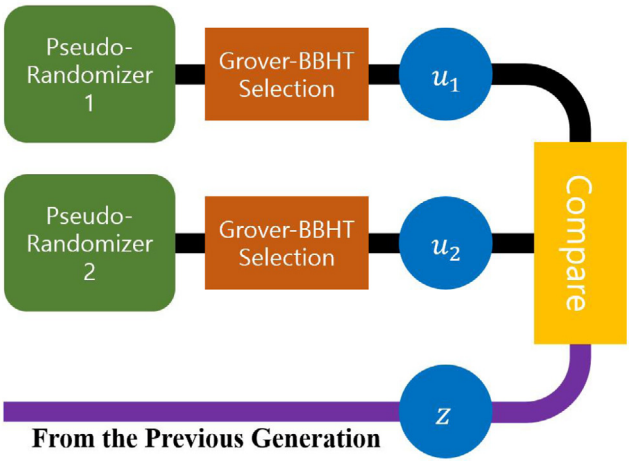

Fig. 4. An abstract procedure of single generation for modified SS-QGA. Two selected individuals from separately constructed population sets, along with z from the previous generation, compete for the current generation's best individual to spare.

4. Experiment

This section explains the experiment conducted via a series of simulations with several optimization tasks to validate the effectiveness of the proposed method as opposed to the original SS-QGA, in terms of the optimizing performance. A part of the experiment was programmed with Qiskit [11], a Python quantum simulator released by IBM, to simulate a quantum system with a classical computer. The experiment was divided into two parts, quantum and classical, for practical reasons. First, the current version of Qiskit does not technically support configuring the Grover's oracle that enables Grover's minimum and maximum search algorithms [11]. It for now only supports the oracles with the static boolean logic, which is unsuitable for the dynamic computations needed to do practical searching. Second, Qiskit's internal *Statevector Simulator* package only replicates up to 24 qubits, which are inadequate in quantity to simulate with for reproducing sufficiently complex problems.

Along with a quantum simulation that constructs a theoretically accurate quantum circuit under a pseudo-quantum environment restricted by the aforementioned limitations, a classical simulation was separately conducted, at the cost of mathematically rigorous quantum processing, in order to create and solve the scalable problems to verify the algorithms' general practicality. For both simulations, the setup prepared a group of continuous optimization problems with the corresponding sizes of domain, through which it is feasible to observe how the modified algorithm competes with the original SS-QGA in the abstract and comparatively real-world environments.

4.1. Quantum and classical simulations

The quantum simulation sets $c = 2$ and $n = 10$, reproducing $2^c = 4$ individuals for each generation. This setting respects SS-QGA's presupposition $c \ll n$ because 2 is quadratically smaller

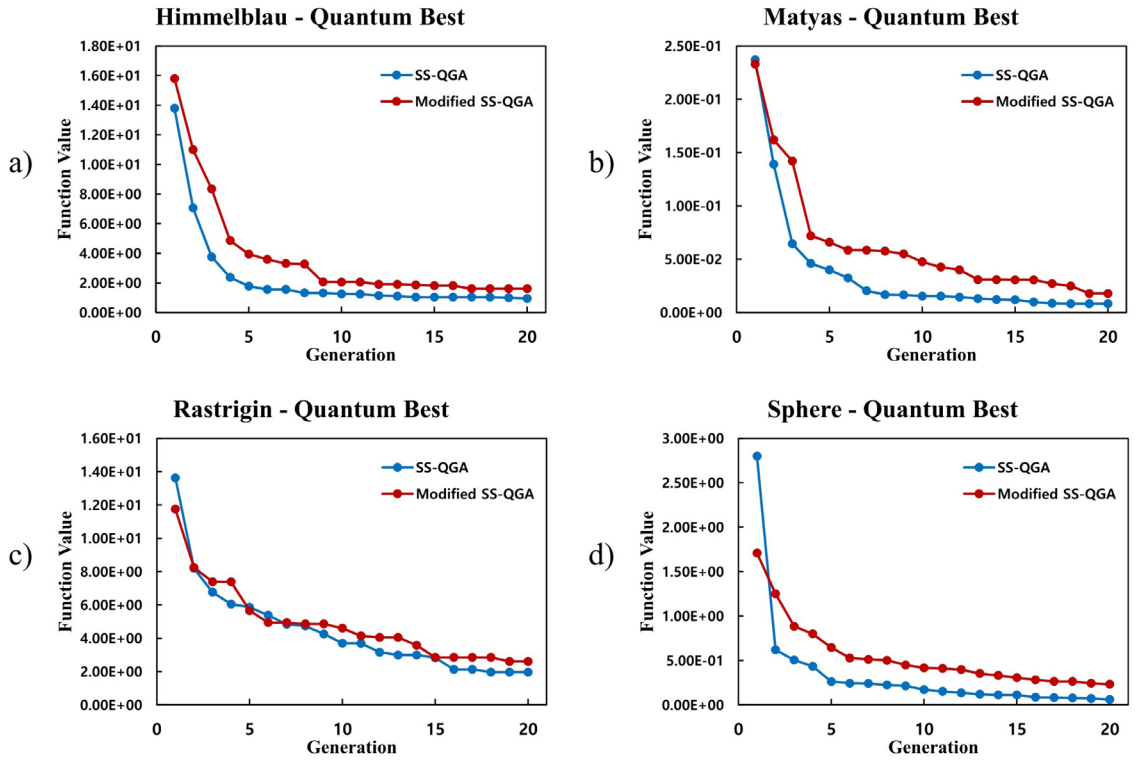

Fig. 5. The best fitness results for the quantum simulation of SS-QGA vs. Modified SS-QGA on the problem functions of (a) Himmelblau, (b) Matyas, (c) Rastrigin, and (d) sphere. The blue curve is the fitness optimization progress of the original SS-QGA, and the black curve is the fitness optimization progress of the modified SS-QGA, over 20 iterations each running 20 generations.

than 10, i.e. $2 < \sqrt{10}$. Simulating 24 qubits in total, it also satisfies Qiskit's 24-qubit limit. Since the number of individuals is highly limited, only simple, continuous toy optimization problems were considered in order to clearly observe changes in fitness values over time. The algorithmic section for population initialization, crossover, and mutation was constructed as a set of appended quantum-simulated circuits via Qiskit, and the selection, measurement, and comparison processes were run classically. For the target optimization problems, the problem functions of Himmelblau, Matyas, Rastrigin, and Sphere were chosen, since they are suitable for short and simple validation [25].

Table 1 shows the important parameters with the allocated values for the quantum simulation. *cr* and *mut* indicate the probabilities for crossover and mutation, and *gen* and *iter* the number of generations and iterations, respectively. Each iteration, having a different starting point to respect the genetic algorithmic stochasticity, is a set of multiple generations and evaluates their degree of optimization in average at each generational step. *cr* is 1, since the algorithm unavoidably performs crossover in every generation, and *mut* is set to 0.05, which is a conventional value many GA-based optimization studies choose to use. Fig. 5 shows the plots of the best function-evaluated fitness values per generation, each averaged via 20 iterations, for the Himmelblau, Matyas, Rastrigin, and Sphere functions, comparing the relative performances of the original and modified SS-QGAs, in the quantum simulation. For simplicity, the fitness and objective functions were set to be equal. Therefore, since all the tested functions are minimization problems, the lowering curves indicate that the intended optimization has been achieved.

For the classical simulation, use of Qiskit was abandoned while the experiment was classically configured by the regular Python programming. Free from the 24 qubit limit, the experiment was set up with $c = 7$ and $n = 60$ as listed in Table 2, generating 128 individuals each with 60 genes. Note that c and n here still

Table 1
Important parameter values for the quantum simulation.

Parameter	Value
c	2
n	10
cr	1
mut	0.05
gen	20
iter	20

satisfy the presupposition $c \ll n$. Now with the binarily longer individuals due to the higher n, the decimal value of each individual's degree of fitness comprises more digits, making the search space larger. Fig. 6 shows the results of the classical simulation on the Himmelblau, Matyas, Rastrigin, and Sphere, comparing the original and modified SS-QGAs. Again, plotted are the best fitness values in each generation, averaged via 20 iterations. Table 3 lays out the mean difference between the best fitness values averaged via 20 iterations with the corresponding standard deviations, rounded to two decimal places, of the original and modified SS-QGAs along 20 generations for the quantum and classical simulations upon each tested function. In addition to plotting the best fitness values, we also tracked the optimization flow of the average fitness values per generation for the classical simulation, as shown in Fig. 7, and their mean value over the generations for each target function is listed in Table 4.

In order to help acknowledge the fitness differences observed in the plots of Figs. 5 and 6 further, Fig. 8 is prepared to show the plots of the absolute gaps of the best fitness differences per generation, averaged via 20 iterations, in the quantum and classical simulations for the target functions. Their mean and standard deviation values are presented in Table 5, in the same manner as Tables 3 and 4. The fitness differences in Table 3 are

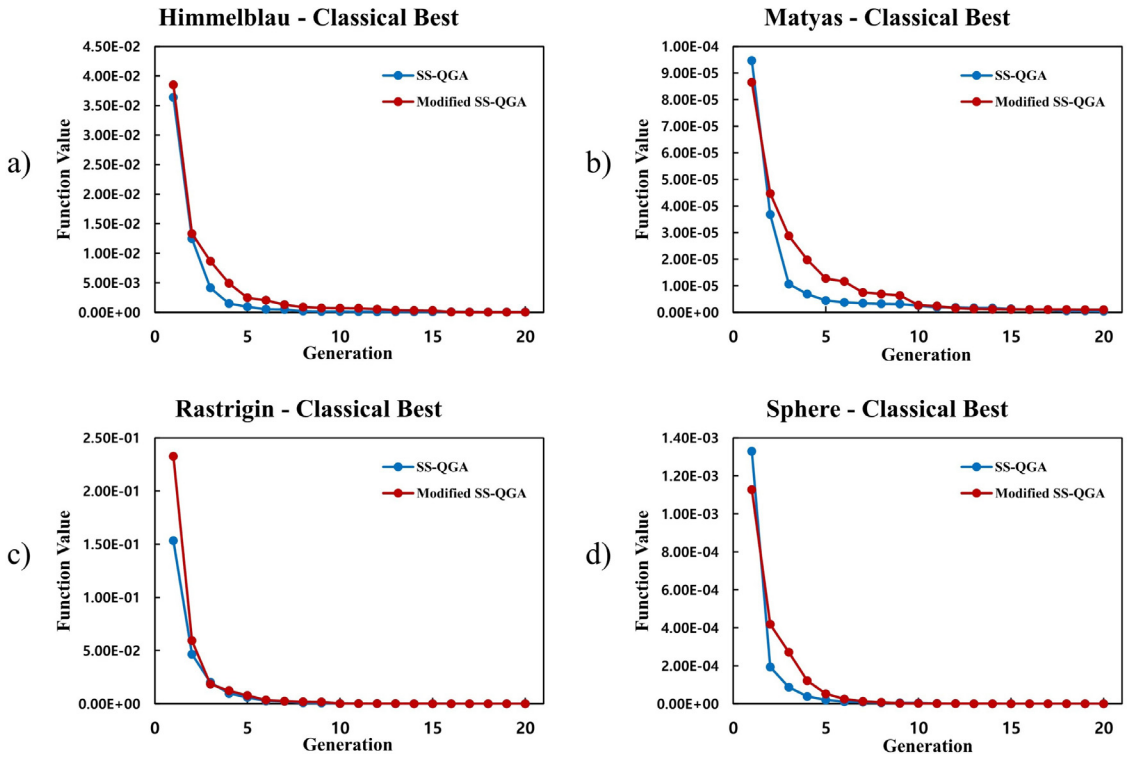

Fig. 6. The best fitness results for the classical simulation of SS-QGA vs. Modified SS-QGA on the problem functions of (a) Himmelblau, (b) Matyas, (c) Rastrigin, and (d) sphere. The blue curve is the fitness optimization progress of the original SS-QGA, and the black curve is the fitness optimization progress of the modified SS-QGA, over 20 iterations each running 20 generations.

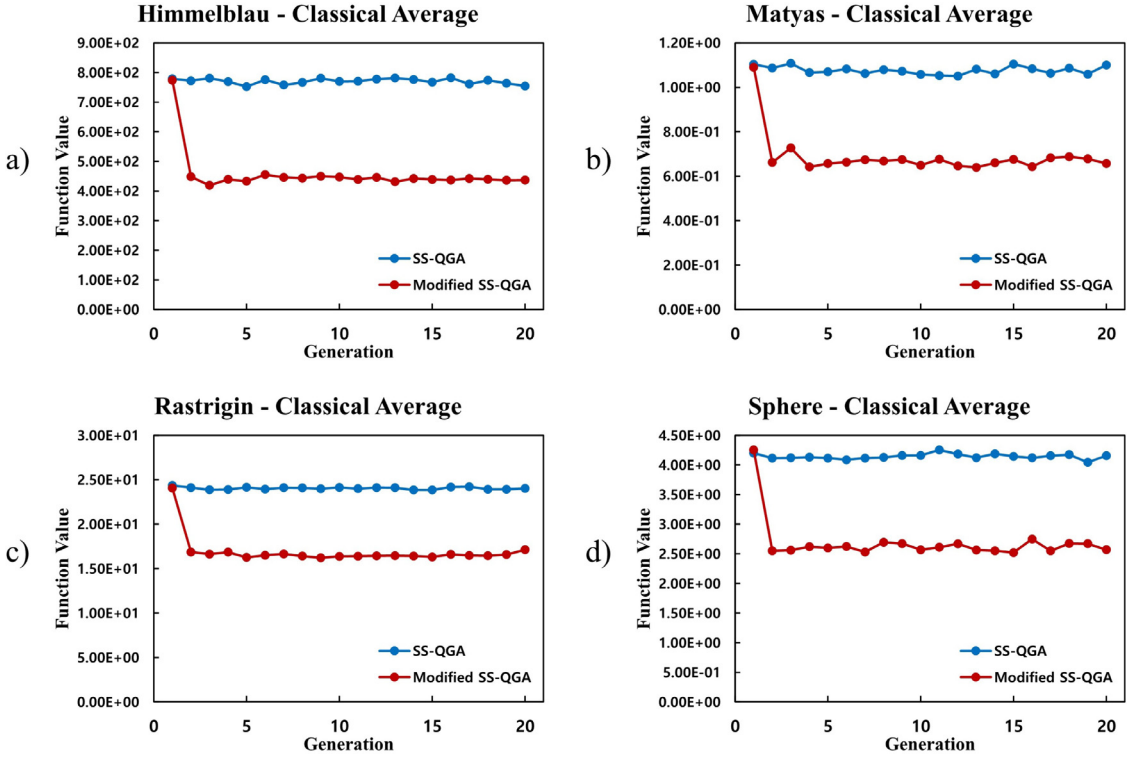

Fig. 7. The average fitness results for the classical simulation of SS-QGA vs. Modified SS-QGA on the problem functions of (a) Himmelblau, (b) Matyas, (c) Rastrigin, and (d) sphere. The blue curve is the fitness optimization progress of the original SS-QGA, and the black curve is the fitness optimization progress of the modified SS-QGA, over 20 iterations each running 20 generations.

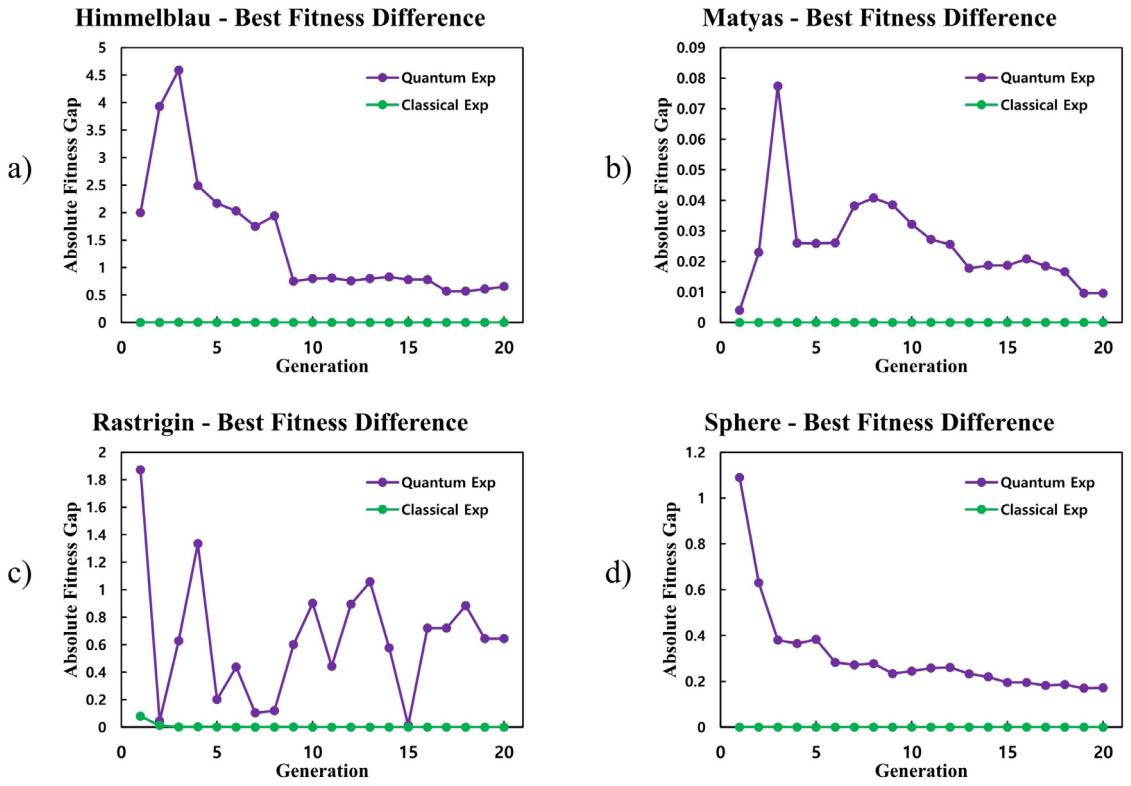

Fig. 8. The absolute gaps of the best fitness differences per generation between the original and modified SS-QGAs in the quantum (the purple curve) and classical (the green curve) simulations on the problem functions of (a) Himmelblau, (b) Matyas, (c) Rastrigin, and (d) sphere.

Table 2
Important parameter values for the classical simulation.

Parameter	Value
c	7
n	60
cr	1
mut	0.05
gen	20
iter	20

Table 3
Compared mean difference between the original and modified SS-QGAs' best fitness values throughout 20 generations for each problem function, quantum vs. classical.

Function	Quantum	Classical
Himmelblau	-1.48 ± 1.15	$-9.09 \times 10^{-4} \pm 1.21 \times 10^{-3}$
Matyas	$-2.54 \times 10^{-2} \pm 1.62 \times 10^{-2}$	$-2.93 \times 10^{-6} \pm 5.72 \times 10^{-6}$
Rastrigin	$-3.94 \times 10^{-1} \pm 6.81 \times 10^{-1}$	$-4.92 \times 10^{-3} \pm 1.79 \times 10^{-2}$
Sphere	$-2.03 \times 10^{-1} \pm 3.22 \times 10^{-1}$	$-1.69 \times 10^{-5} \pm 8.16 \times 10^{-5}$

Table 4
The means of the original and modified SS-QGAs' average fitness values throughout 20 generations for each problem function in the classical simulation.

Function	Original Average Fitness	Modified Average Fitness
Himmelblau	$7.71 \times 10^{2} \pm 9.05$	$4.57 \times 10^{2} \pm 7.49 \times 10$
Matyas	$1.08 \pm 1.87 \times 10^{-2}$	$6.88 \times 10^{-1} \pm 9.67 \times 10^{-2}$
Rastrigin	$2.40 \times 10 \pm 1.32 \times 10^{-1}$	$1.69 \times 10 \pm 1.71$
Sphere	$4.14 \pm 4.45 \times 10^{-2}$	$2.69 \pm 3.73 \times 10^{-1}$

Table 5
The absolute gap of the mean difference between the original and modified SS-QGAs' best fitness values throughout 20 generations for each problem function, quantum vs. classical.

Function	Quantum vs. Classical Gap
Himmelblau	1.48 ± 1.14
Matyas	$2.58 \times 10^{-2} \pm 1.55 \times 10^{-2}$
Rastrigin	$6.37 \times 10^{-1} \pm 4.46 \times 10^{-1}$
Sphere	$3.12 \times 10^{-1} \pm 2.12 \times 10^{-1}$

gained by subtracting the fitness values of the original from those of the modified. Since every tested function is of a minimization problem, minus signs indicate that the original has achieved better optimization. On the other hand, the gaps in Table 5 merely demonstrate the numerical differences in the fitness.

4.2. Comparison on amount of computation

In order to support another claim that the modified SS-QGA is more time-efficient than the original SS-QGA, the number of fitness evaluations per generation for each algorithm was counted and plotted cumulatively, as seen in Fig. 9. Table 6 lists the number of total fitness evaluations made in the original and modified SS-QGAS in the quantum classical simulations, along with the corresponding ratios between the two algorithms. The hyperparameter η, introduced in Section 3.2, is set to 1, with which the probability that the chosen individual is the fittest is 1/2. Setting $\eta > 1$ could be a plausible choice in terms of ensuring a higher probability to secure the fittest individuals, but since η is multiplied as a constant for both algorithms, the overall outcome would not lead to drawing a different conclusion. Note that the Qiskit's quantum simulation assumes an ideal quantum machine, in which quantum-oriented errors such as decoherence do not occur [11]. In a real-life experiment, the number of fitness evaluations performed by the Grover's diffuser could fluctuate due to the machine's physical incompleteness.

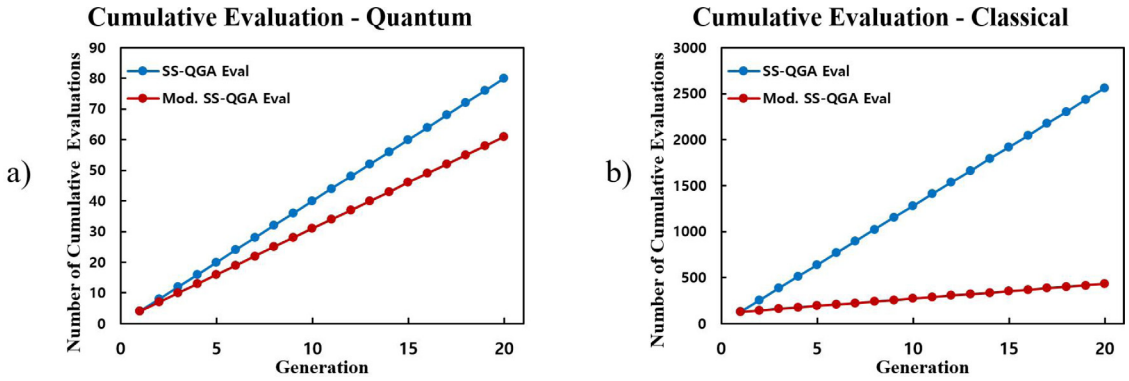

Fig. 9. The flows of cumulative evaluations for the original (the blue curve) and modified (the red curve) SS-QGAs, in (a) the quantum and (b) the classical simulations. Since the number of conducted evaluations is irrelevant to the types of the target problem functions but to the number of the involved individuals, all the problem functions in each simulation share the same result.

Table 6
The numbers of total evaluations in the original and modified SS-QGAs for the quantum and classical simulations, along with their ratios between the two algorithms.

Experiment (c, n)	Original	Modified	Ratio
Quantum (2, 10)	80	61	1.31
Classical (7, 60)	2560	432	5.93

5. Analysis

Figs. 5 and 6, side by side, manifest the comparative optimization performances of the original and modified SS-QGAs. In Fig. 5, note that the modified algorithm generally performs worse than the original in all cases. Particularly, the modified generally failed to reach the degree of optimization that the original achieved at the end. Such relatively poor performance has been expected, since the original algorithm can secure a large pool of random individuals, while any possible configuration for chromosomes remains extremely simple under the quantum simulated environment, forming a narrow search space. In other words, due to the short length of chromosomes, the number of possible combinations of genes is highly limited, and the original SS-QGA can thus relatively frequently discover the highly fitted individual from the pool of the randomly created individuals via a series of stochastic tryouts.

It was anticipated that the aforementioned positive effect of the random individuals would be minimized if the number and the length of chromosomes increased accordingly with the complexity of a target problem, which was the very case for the classical simulation. Refer to Fig. 6. Although the modified SS-QGA still performs slightly worse than the original SS-QGA, the degrees of discrepancy are noticeably minimized with respect to the cases of the quantum simulation. For every target function, the modified achieved the same degree of convergence to the original roughly at the midpoint of the generation series. While the classical simulation maintains the same target functions to solve, it maps a larger search space due to the increased number of genes within each chromosome, implying that a dramatically higher number of possible combinations of the gene values exists. The enlarged search space reduces the probability of acquiring desirable solutions by blind exploration with random individuals, thus invalidating the advantage of stochasticity for the original SS-QGA that prevailed in the quantum simulation.

It is worth noticing that the absolute value of the 'Quantum' results in Table 3 and the gaps in Table 5 are roughly similar. For example, the absolute fitness difference of the quantum Himmelblau, 1.48 ± 1.15, is nearly identical to the function's quantum vs. classical gap, 1.48 ± 1.14. Such outcome implies that the fitness differences in the classical simulation is virtually negligible compared to the differences in the quantum simulation, supporting the presumption of this paper that the performance gap between the two algorithms would shrink with respect to the increase in the dimension and complexity of the target problem function's search space and domain. The graphical comparison in Fig. 8 particularly displays the relative significance of the fitness differences generated in the quantum simulation as opposed to the differences in the classical simulation, which can be said to be barely existent in comparison.

In Fig. 7, both quantum and classical average fitness flows stay moderately horizontal until the end of the generational series, reflecting the low ratio of the highly fitted individuals to the partially or fully random individuals in the population. It can also be observed that the classical flows drop considerably at the second generation, where the proposed method of the population configuration is applied for the first time. Such a discrepancy displayed in all the cases suggests that getting rid of randomly generated individuals positively contributes to the overall population fitness. In other words, in spite of the removal of the significant portion of the population, the modified SS-QGA performs better than the original in terms of the average fitness of the overall population and only slightly worse in terms of the best fitness individuals, implying that the eliminated individuals indeed hardly help the optimization task. The differences in the average fitness values of the two algorithms can be seen more clearly in Table 4 where the values in the third column are substantially lower than those in the second column. Correlating such results with those represented in Tables 3 and 5 and the plots in Figs. 5, 6, and 8, we claim that the modified SS-QGA is identical to the original SS-QGA in terms of the optimization performance when they are utilized to handle sufficiently large and complex problems, which are mostly the case in practical usage.

Refer to Table 6 for the numerical details about the difference in time complexity between the original and modified SS-QGAs. For the classical simulation especially, the number of evaluations made in the original is nearly six times higher than the number of evaluations made in the modified. Such a result well suits our anticipation, since the increase step of evaluation per generation in the modified is quadratically less than that of the original in terms of $\tilde{N} = 2^c$, i.e. each step of the modified costs $O(\sqrt{2^{c+1}})$ as opposed to $O(2^c)$ evaluations in the original. The difference is significantly more obvious in the classical simulation where c is higher. It must be noted that the number of generations set for the experiment in this study is remarkably low due to the simplicity of the target functions and the limited availability of

qubits for the quantum simulation. The number of generations set for practical or real-world applications of genetic algorithms often range from 100 to 10000, for which the difference in the total number of evaluations between the original and modified SS-QGAs would vary much more severely than the result presented here.

As discussed in Section 3, the modified SS-QGA maintains the overall circuit configuration of the original SS-QGA and thus shares the same space complexity. The original requires $poly(\log N)$ qubits and $O(\log N)$ classical bits, with $N = 2^n$ and n as the number of qubits in the $n - qubit$ register [10]. The classical bits are needed merely in order to store the measured individual after the quantum computation. Since the same process from the original takes place in the first generation, the same number of qubits is needed, resulting the same requirement of $poly(\log N)$ qubits. Because in any case $c \ll n$, with c as the number of qubits in the $c - qubit$ register, the reduced size of c for the modified does not affect its space complexity analysis.

Note that the labels 'quantum' and 'classical' are merely for convenient distinction; both experiments are intended to simulate an algorithm to be run on a strict quantum device. The quantum simulation here, strictly speaking, is not fully quantum-processed. Due to the lack of the type of the oracle needed for Grover's minimum and maximum search algorithms in the current version of Qiskit, the selection process was run classically. Preparing and initializing the populations are done with the logic under quantum circuits, however, and with regards to this paper's main focus on changing the population layout and size, the overall experiment design well serves the purpose of this study. Because of the limited number of simulated qubits, every problem function was chosen to be continuous, for which, with only the short chromosomes available, the performance difference is set to be more clearly observable.

The classical experiment, as previously stated, has been organized with classical programming from the beginning to the very end. It is coded to generate and initialize population sets in the same way as a quantum circuit would, but it is programmed entirely with the classical Python functions in order to break free from the computational restrictions of Qiskit. The aim was to show the validity of the proposed method, though classically, via problems with sufficiently large search domains. As mentioned earlier, both algorithms perform more similarly to each other as the problems become larger and more complicated. The conclusive analysis is that the original and modified SS-QGAs will mark nearly the same level of performance in optimization for most if not all large, practical problems.

6. Conclusion

The main purpose of this paper, in summary, is to prove that the suggested size of each generation's population for the original SS-QGA can be effectively reduced to the degree where the optimizing performance still remains nearly undamaged. Two sets of experiments, quantum and classical, were prepared to evidently show the comparative performance improvement of the proposed method with regards to the original SS-QGA. The overall outcome reveals the tendency that the gap between the two algorithms' performance levels shrinks as more complicated and domain-wise larger problems are targeted to solve; it is an expected result, since individuals with strong randomness in their configuration possess little chances to imitate the delicate genetic configuration of highly fitted individuals in a dimensionally and spatially complex environment [24]. The main contribution of this work is to have proved the validity of considering a smaller size of population when one implements QGA.

Strictly Structured Quantum Genetic Algorithm, as mainly discussed in this paper, is one sort of semi-classical quantum genetic algorithms that follow the general procedure of conventional, classical genetic algorithms. Although they are seemingly more practical, it is a common belief among researchers that a fully quantum-oriented genetic algorithm, if there is any, would be likely to outperform them in terms of handling multidimensional optimization problems [26]. It is still seemingly desirable, however, to continue developing new ideas on semi-classical quantum genetic algorithms, since doing so can hopefully lead to discovering a crucial cornerstone of a rigid structure for a fully quantum-oriented algorithm, which has not been solidly defined by any related literature.

Quantum genetic algorithm as an optimization theory, in the current stage, is by all means in its infancy. One can validate its expected performances via simulated environments, but how it would actually perform under a real quantum environment remains questionable. It will be only after a fully functional quantum machine has been developed that an actual validation of quantum genetic algorithm as a practical and reliable optimization tool becomes available. The main objective of researching quantum genetic algorithm should then be to have its well-defined procedure ready to be committed just as a quantum machine comes in our hands. Hopefully the recent rapid advancements of physical quantum machines can help let those days come faster. Only time will certainly tell.

7. Future work

More advanced methods to process the final comparison step can be introduced in future studies. For the modification introduced in this work, the comparison between the best individuals from the two population subsets must be done in series in order to avoid nullifying the magnified probability amplitudes of the individual states. While such a modification does not severely hinder the overall computational effectiveness of the algorithm, it is anticipated that there should be a more *elegant* way to fulfill this process, perhaps in parallel. Such design might also involve a simpler circuit configuration, which is clearly an advantage in terms of building an effective quantum algorithm.

With regards to the fact that there are only few quantum genetic algorithms or quantum meta-heuristic methodologies that have been developed so far, one would be encouraged to make a comparison between his/her own proposed quantum genetic algorithm with existing, state-of-the-art classical genetic algorithms in order to observe in which aspects the adaptation of quantum computing could affect the optimization progress. Such an effort should be an important step, since there is a series of predictions that the NP-equivalent problems exist in quantum computing [2], implying that the meta-heuristic approach could still be a valuable process to achieve optimality within quantum computers.

Future work on this topic can also contribute to discovering possible applications of the proposed method. The intrinsic purpose of SS-QGA is to replicate the process of classical genetic algorithms for quantum processors with the enhanced performance in terms of time and space complexities, and the proposed modification in this paper further improves it with the reduced size of population. As such, we expect that the original and modified SS-QGAs are applicable to virtually every optimization task that exploits a genetic algorithmic approach.

The toy optimization problems examined here aside, there are many real-world tasks, either single or multi-objective, where the proposed improvement can be effectively applied. For example, the traveling salesman is a type of NP-hard optimization problems, which is frequently exploited in studies of optimization theory as a benchmark [27], while its objective is closely correlated to the real-world problems such as optimized vehicle

routing, delivery schedule arrangement, and efficient placement of pin in an integrated circuit. The mentioned problems often include the massive number of possible combinations that requires the sufficiently large size of population in terms of the genetic algorithmic approach, and the proposed improvement can certainly contribute positively to maintaining the population size to an acceptable degree when an attempt to solve it via SS-QGA is made.

CRediT authorship contribution statement

Jun Suk Kim: Conceptualization, Methodology, Software, Writing – original draft, Visualization, Investigation. **Chang Wook Ahn:** Validation, Writing – review and editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. NRF-2021R1A2C3013687, No. NRF-2021R1A6A3A13046634).

References

[1] F. Arute, K. Arya, R. Babbush, et al., Quantum supremacy using a programmable superconducting processor, Nature 574 (2019) 505–510.
[2] N.S. Yanofsky, M.A. Mannucci, Quantum Computing for Computer Scientists, first ed., Cambridge University Press, 2008.
[3] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, IEEE Trans. Evol. Comput. 6 (6) (2002) 580–593, http://dx.doi.org/10.1109/TEVC.2002.804320.
[4] S. Shirazi, M. Menhaj, A new genetic based algorithm for channel assignment problems, in: B. Reusch (Ed.), Computational Intelligence, Theory and Applications, Springer, 2006, pp. 85–91, http://dx.doi.org/10.1007/3-540-34783-6_10.
[5] Z. Laboudi, S. Chikhi, Comparison of genetic algorithm and quantum genetic algorithm, Int. Arab J. Inf. Technol. 9 (2012).
[6] A. Malossini, E. Blanzieri, T. Calarco, Quantum genetic optimization, IEEE Trans. Evol. Comput. 12 (2008) http://dx.doi.org/10.1109/TEVC.2007.905006.
[7] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, thirteenth ed., Addison-Wesley Professional, 1988.
[8] C.C. Moran, Mastering Quantum Computing with IBM QX, first ed., Packt Publishing, 2019.
[9] R. Lahoz-Beltra, Quantum genetic algorithms for computer scientists, Computers 5 (4) (2016) http://dx.doi.org/10.3390/computers5040024, URL https://www.mdpi.com/2073-431X/5/4/24.
[10] A. SaiToh, R. Rahimi, M. Nakahara, A quantum genetic algorithm with quantum crossover and mutation operations, Quantum Inf. Process. 13 (2014) 737–755.
[11] H. Abraham, A. Offei, R. Agarwal, et al., Qiskit: An open-source framework for quantum computing, 2019, http://dx.doi.org/10.5281/zenodo.2562110.
[12] M. Udrescu, L. Prodan, M. Vlăduţiu, Implementing quantum genetic algorithms: A solution based on Grover's Algorithm, in: Proceedings of the 3rd Conference on Computing Frontiers, in: CF '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 71–82, http://dx.doi.org/10.1145/1128022.1128034.
[13] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, first ed., Cambridge University Press, 2004.
[14] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (7671) (2017) 195–202, http://dx.doi.org/10.1038/nature23474.
[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, in: STOC '96, Association for Computing Machinery, New York, NY, USA, 1996, pp. 212–219, http://dx.doi.org/10.1145/237814.237866.
[16] M. Boyer, G. Brassard, P. Høyer, A. Tapp, Tight bounds on quantum searching, Fortschr. Phys. 46 (4–5) (1998) 493–505, http://dx.doi.org/10.1002/(sici)1521-3978(199806)46:4/5<493::aid-prop493>3.0.co;2-p.
[17] H. Wang, J. Liu, J. Zhi, C. Fu, The improvement of quantum genetic algorithm and its application on function optimization, Math. Probl. Eng. 2013 (2013) http://dx.doi.org/10.1155/2013/730749.
[18] D. Sofge, Prospective algorithms for quantum evolutionary computation, in: Proceedings of the Second Quantum Interaction Symposium, QI-2008, College Publications, Oxford, UK, 2008, pp. 26–28.
[19] P.R. Giri, V.E. Korepin, A review on quantum search algorithms, Quantum Inf. Process. 16 (12) (2017) 1–36, http://dx.doi.org/10.1007/s11128-017-1768-7.
[20] C. Dürr, P. Hoyer, A quantum algorithm for finding the minimum, 1996, CoRR quant-ph/9607014.
[21] A. Ahuja, S. Kapoor, A quantum algorithm for finding the maximum, 1999, arXiv:arXiv:quant-ph/9911082.
[22] Y. Akimoto, Saddle point optimization with approximate minimization oracle, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, New York, NY, USA, 2021, pp. 493–501.
[23] M. Chiesa, G. Maioli, G.I. Colombo, L. Piacentini, GARS: Genetic algorithm for the identification of a robust subset of features in high-dimensional datasets, BMC Bioinformatics 21 (2020) http://dx.doi.org/10.1186/s12859-020-3400-6.
[24] S. Katoch, V. Chahar, S. Chauhan, A review on genetic algorithm: Past, present, and future, Multimedia Tools Appl. 80 (2021) http://dx.doi.org/10.1007/s11042-020-10139-6.
[25] D.M. Himmelblau, Applied Nonlinear Programming, first ed., McGraw-Hill, 1972.
[26] W. He, Y. Shi, Multiobjective construction optimization model based on quantum genetic algorithm, Adv. Civ. Eng. 2019 (2019) 1–8, http://dx.doi.org/10.1155/2019/5153082.
[27] S. Conforto, A. Hussain, Y.S. Muhammad, et al., Genetic algorithm for traveling salesman problem with modified cycle crossover operator, Comput. Intell. Neurosci. 2017 (2017).

Jun Suk Kim – was born in South Korea, in August 1989. He received his Bachelor's degree in Physics from University of Illinois in Urbana-Champaign, USA in 2016 and received his Master's degree in Computer Science from Gwangju Institute of Science and Technology, South Korea in 2019. He is currently working for his degree of Ph.D. in Gwangju Institute of Science and Technology. His focused area of research is Reinforcement Learning, Quantum AI and Quantum Genetic Algorithm.

Chang Wook Ahn is working as a Professor in the AI Graduate School, Gwangju Institute of Science and Technology (GIST), Republic of Korea. He received a Ph.D. degree from the Department of Information and Communications at GIST in 2005. From 2005 to 2007, he worked in Samsung Advanced Institute of Technology, Korea. From 2007 to 2008, he was a Research Professor at GIST. From 2008 to 2016, he was an Assistant/Associate Professor at the Department of Computer Engineering, Sungkyunkwan University (SKKU), Republic of Korea. His research interests include genetic algorithms/programming, multi-objective optimization, neural networks, and quantum machine learning.