IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 11 April 2024, accepted 14 May 2024, date of publication 20 May 2024, date of current version 3 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3403200

RESEARCH ARTICLE

Grover Adaptive Search With Fewer Queries

HIROAKI OMINATO[1], TAKAHIRO OHYAMA[1], (Graduate Student Member, IEEE), AND KOICHIRO YAMAGUCHI[2]
[1]Panasonic System Networks Research and Development Laboratory Company Ltd., Sendai 981-3206, Japan
[2]Panasonic Connect Company Ltd., Research and Development Division, Yokohama, Kanagawa 224-0054, Japan

Corresponding authors: Hiroaki Ominato (ominato.hiroaki@jp.panasonic.com) and Takahiro Ohyama (ohyama.takahiro-pmcs@jp.panasonic.com)

ABSTRACT In binary optimization problems, where the goal is to find the input $\boldsymbol{x}$ that minimizes a given objective function, Grover adaptive search (GAS) is a well-known quantum algorithm that provides quadratic speedup when compared with the brute-force approaches of classical computers. GAS calls a renowned quantum search algorithm, Grover search (GS), as a subroutine to search for inputs with function values less than the threshold value. If such an input is found, the threshold value is updated with the function value and the search targets are narrowed. To search efficiently for a solution, an appropriate number of queries must be selected to amplify the desired state of the GS in each step. This paper discusses this issue and proposes a new method for selecting the number of queries and provides proof that our method achieves quadratic speedup as well as the original GAS. Furthermore, the simulation results for 40-bit problems confirm that our method allows an optimal solution with 22% fewer queries than the standard method, thus offering the possibility of significantly reduced computation times for combinatorial optimization problems.

INDEX TERMS Computational complexity, Grover search, Grover adaptive search, optimization, quantum algorithm, quantum computing, search problem.

I. INTRODUCTION

The combinatorial optimization problem involves determining an input that minimizes a certain objective function. This problem has extensive applications in various fields such as supply chains [1], [2], [3], [4], [5], [6], [7], energy management [8], [9], [10], and finance [11], [12]. Combinatorial optimization problems can always yield optimal solutions through brute-force searches on conventional classical computers. However, in the case of large-scale problems, the computational cost is extremely high owing to the exponential nature of the computation. Hence, several approaches have been proposed to solve these problems using quantum computers, which support solutions that were previously unfeasible on classical platforms.

Quantum annealing [13] is a heuristic technique that leverages quantum effects such as the superposition of states. Quadratic unconstrained binary optimization (QUBO) can be directly input into quantum annealing, whereas handling higher-order unconstrained binary optimization (HUBO) requires additional bits for conversion to QUBO. This increases the number of bits and adversely affects the solution search. Among the hardware available for quantum computing, quantum annealing hardware can support a high qubit count. Therefore, several attempts have been made to apply it to real-world problems [14], [15], [16], [17], [18], [19]. Additionally, variational quantum eigensolver methods, such as the quantum approximate optimization algorithm (QAOA) [20], are available. These involve heuristics as well as quantum annealing but have the advantage of being able to handle HUBO without conversion. They can be implemented on noisy intermediate-scale quantum (NISQ) devices, which are general-purpose quantum computers with limited bits and no error-correction capability. However, the superiority of the abovementioned algorithms over classical algorithms remains debatable.

In this context, Grover adaptive search (GAS) [21] has theoretically been proven to achieve quadratic speedup when compared with classical methods. In other words, the time complexity of classical approaches is $\mathcal{O}(N)$, whereas that of GAS is $\mathcal{O}(\sqrt{N})$, where N is the size of the search space. In GAS, the objective function value of the best solution

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

at each search step is taken as a threshold and Grover search (GS), which targets solutions with objective function values below this threshold, is repeatedly executed to explore better solutions. Similar to the QAOA, GAS can handle HUBO directly. Moreover, GAS has an advantage over the aforementioned methods, as it can provide optimal solutions. These characteristics have fostered research on applying GAS [22], [23] and reducing computational costs [24]. However, GAS is unavailable in NISQ devices because its execution requires a deep circuit. Therefore, a fault-tolerant quantum computer may hold promise for combinatorial optimization.

Another challenge in GAS is that despite the impossibility of knowing the ideal number of queries in each loop, the number of times the query is to be implemented must be chosen to obtain the desired state with a high probability (for details, see Section II-B). This is also known as the soufflé problem [25] in the context of GS, where too many (overcooked) or too few (undercooked) queries do not yield good results.

Three types of approaches are possible for addressing this problem: trial-and-error method [26], quantum counting [27], and fixed-point quantum search [25], [28]. In the trial-and-error method, the number of queries is selected iteratively in accordance with some strategy. A method proposed in the original paper on GAS is classified as the trial-and-error method. This method has been the standard approach for GAS, because its strategy is simple and it preserves the quadratic speedup of GS. In addition to this method, alternative approaches for the selection of number of queries have been proposed by Giuffrida et al. [29] and Baritompa et al. [30], which are also classified as trial-and-error methods. Although the query complexity of GS has been analytically compared using several methods [31], the optimal method remains unclear in the context of GAS. This is a crucial issue for applying GAS, because optimizing the control method for the number of queries would probably reduce the constant overhead of the query complexity.

In this paper, we discuss the conditions for more efficient computation by considering the simulation results when using the control method in the original paper on GAS [21] and propose a new method for controlling the number of queries. The time complexity of the proposed method is $\mathcal{O}(\sqrt{N})$, which is the same as that of the original GAS, and its proof is shown in Appendix A. Furthermore, we introduce a faster and low-cost simulation method for GAS to verify the efficacy of the proposed method. Considering the simulation results of a 40-bit problem, the proposed method achieves a 22% reduction in the constant overhead of the query complexity when compared with the standard method. Moreover, we compare the performance of the proposed method with those of several existing methods for three combinatorial optimization problems and reveal the superiority of the proposed method over the others. We further propose a GAS framework that incorporates termination conditions based on the simulation results. Our framework finds optimal solutions to combinatorial optimization problems, such as QUBO or HUBO in a shorter time, which can contribute to faster solutions for many real-world optimization problems.

The remainder of this paper is organized as follows. Section II presents the theoretical background for GS and GAS. Section III discusses the number of queries and proposes a new control method. In Section IV, we explain the method for GAS simulation and validate the effectiveness of the proposed method based on the simulation results. In Section V, we discuss the optimal termination conditions based upon the simulation results and suggest avenues for future research. Finally, we present the concluding remarks in Section VI.

II. THEORETICAL FOUNDATION

A. GROVER SEARCH: GS

GS is an essential element of GAS by which the amplitude of the target state is amplified before the measurement using these operators:

1) $H^{\otimes n}$: Hadamard gate on n qubits
2) $\hat{O} = I - 2|A\rangle\langle A|$: Oracle
3) $\hat{D} = 2|s\rangle\langle s| - I$: Diffusion operator

The oracle operator $\hat{O}$ inverts the sign of only the desired state and must be individually prepared depending on the problem. We explain the details of the GS when the size of the search space is $N = 2^n$ (n: number of bits) and number of search targets is t. Let the set of search target indices be $\mathcal{I}$. First, we create an overlap of all N states by using the following Hadamard gate:

$$|s\rangle = H^{\otimes n}|0\dots0\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle \tag{1}$$

$$= \sqrt{\frac{t}{N}}|A\rangle + \sqrt{\frac{N-t}{N}}|B\rangle \tag{2}$$

$$= \sin\theta\,|A\rangle + \cos\theta\,|B\rangle\,, \tag{3}$$

where $|A\rangle$ is the desired state, that is, the superposition of all search targets, and $|B\rangle$ is the undesired state, i.e., the superposition of the others.

$$|A\rangle = \frac{1}{\sqrt{t}}\sum_{i\in\mathcal{I}}|i\rangle\,, \tag{4}$$

$$|B\rangle = \frac{1}{\sqrt{N-t}}\sum_{i\notin\mathcal{I}}|i\rangle\,. \tag{5}$$

Applying the operator $\hat{O}$ to $|s\rangle$, we obtain the state in which $|A\rangle$ is multiplied by -1:

$$\hat{O}H^{\otimes n}|0\dots0\rangle = -\sin\theta\,|A\rangle + \cos\theta\,|B\rangle\,. \tag{6}$$

Then, using the operator $\hat{D}$, we invert the sign of the state perpendicular to $|s\rangle$, as follows:

$$\hat{D}\hat{O}H^{\otimes n}|0\dots0\rangle = \sin 3\theta\,|A\rangle + \cos 3\theta\,|B\rangle\,. \tag{7}$$

By repeatedly applying the operators $\hat{O}$ and $\hat{D}$, the desired state $|A\rangle$ is amplified and observed. The number of iterations is the same as the number of queries; i.e., the number of times that the oracle operations are queried. When the number of queries is r, the final state $|\psi\rangle$ is given by

$$|\psi\rangle = \sin\left((2r+1)\theta\right)|A\rangle + \cos\left((2r+1)\theta\right)|B\rangle, \quad (8)$$

$$\theta = \arcsin\sqrt{\frac{t}{N}}. \quad (9)$$

The probability of observing the desired state $|A\rangle$ from $|\psi\rangle$ is $\sin^2((2r+1)\theta)$. Therefore, the ideal number of queries for observing the desired state with a probability of 1 is given by

$$r_{opt} = \frac{1}{2}\left(\frac{\pi}{2\theta} - 1\right) \quad (10)$$

$$= \frac{1}{2}\left(\frac{\pi}{2\arcsin\sqrt{\frac{t}{N}}} - 1\right). \quad (11)$$

The number of targets, t, is unknown. For GAS, the number of inputs with objective function values below the threshold corresponds to t. Because it is impossible to determine t accurately without an exhaustive search for the distribution of solutions, it is difficult to select the ideal number of queries. In such cases, where t is unknown, the method proposed in [26] shows that the expected number of queries required to obtain the desired state is at most $9/4\sqrt{N/t}$. This method involves repeating the aforementioned GS and randomly selecting the number of queries from integer values that are less than an exponentially increasing upper limit according to the number of repetitions. This method was applied at each threshold update in the GAS, as detailed in the next section.

B. GROVER ADAPTIVE SEARCH: GAS

In GAS, a candidate bit string for the argument that provides the minimum value for a certain objective function $f : \{0,1\}^n \rightarrow \mathbb{R}$ (i.e., the optimal solution) is generated by sampling $|\psi\rangle$, the final state of GS.

$$\underset{\boldsymbol{x}\in\{0,1\}^n}{\operatorname{argmin}} f(\boldsymbol{x}), \quad (12)$$

where $\boldsymbol{x}$ is a binary variable vector of length n that can be represented by n bits. The quantum state to be sampled is constructed by amplifying the probability amplitude of the desired state from the initial state, which is a superposition of all possible solutions, as in GS. In general, the number of queries for an oracle is used as an evaluation metric for search algorithms. In a brute-force search, to evaluate the function value, an oracle must be applied once for each bit string of length n to determine the optimal solution, resulting in a query complexity of $\mathcal{O}(N)$ ($N = 2^n$). However, according to [21], the number of queries required in GAS to obtain an optimal solution with a probability of more than 1/2 is at most $\frac{45}{4}\sqrt{N}$ [21]. This indicates that GAS can achieve a quadratic speedup in query complexity when compared with the classical brute-force search.

The steps used in GAS [21] are shown in Algorithm 1. Lines 3-11 in Algorithm 1 is referred to as the "search loop." r is the number of queries randomly chosen between 0 and the upper limit m. λ is the increase rate of the upper limit m, which is multiplied by λ for each search loop. The parameter λ influences the performance of GAS, which will be considered through simulation results in Section IV-C. R is the cumulative number of queries, representing the computational cost up to each search loop. $\hat{O}_{y_i}$ is an oracle that applies -1 to the state $|\boldsymbol{x}\rangle$ satisfying $f(\boldsymbol{x}) < y_i$, which can be constructed efficiently by embedding the function value into the ancilla qubits and comparing it with the threshold [32]. Algorithm 1 repeatedly executes the GS, which samples the quantum state $|\psi\rangle$ to search for solutions with objective function values below the threshold. If the search is successful, where the sampling result x satisfies $f(x) < y_i$, it narrows the solution space by updating the threshold with the objective function value. Typically, the number of solutions with objective function values below the threshold is unknown; therefore, the ideal number of queries r_{opt} represented by Equation (11) is also unknown.

Algorithm 1 Grover Adaptive Search [21]

```
Require: $f : \{0,1\}^n \rightarrow \mathbb{R}, \lambda > 1$
 1: Uniformly sample $\boldsymbol{x}_1 \in \{0,1\}^n$
 2: Set $\boldsymbol{y}_1 = f(\boldsymbol{x}_1)$, $m = 1$, $i = 1$, $R = 0$
 3: repeat
 4:     Randomly choose $r$ from $\{0, 1, \ldots, [m-1]\}$ and $R = R + r$
 5:     Execute GS using $r$ queries with oracle of $\hat{O}_{y_i}$, and obtain the sampling result $\boldsymbol{x}$ and its function value $y = f(\boldsymbol{x})$
 6:     if $y < y_i$ then
 7:         $\boldsymbol{x}_{i+1} = \boldsymbol{x}$, $y_{i+1} = y$ and $m = 1$
 8:     else
 9:         $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i$, $y_{i+1} = y_i$ and $m = \lambda m$
10:     end if
11:     $i = i + 1$
12: until $R > 22.5\sqrt{N}$
13: return $\boldsymbol{x}_i$
```

In addition to the standard method adopted in Algorithm 1, several methodologies for selecting the number of queries have been proposed. A notable contribution of this study is the development of a new method for selecting the number of queries. The details are presented in Section III-A.

III. PROPOSED METHOD

A. NEW METHOD TO CONTROL THE NUMBER OF QUERIES: WITHOUT UPPER LIMIT RESET

First, we explain the standard method for controlling the number of queries, which is adopted in Algorithm 1 as well as in the libraries of Qiskit [33] and in the literature on GAS [32], [34], [35]. This method randomly selects the number of queries r from the integer values below the upper limit m. Starting from 1, m is multiplied by λ for each search loop; each time a solution with an objective function value below the threshold is observed and the threshold is updated, m is reset to 1 (line 7 in Algorithm 1). This is because, as mentioned in Section II-A, we apply the GS method in [26] for each threshold. Figure 1 shows an example of the transition of the number of queries (blue solid line), the upper limit (green dashed line), and the ideal number of queries r_{opt} (red dotted line) in the GAS simulation. Having thereby randomly selected the number of queries, the probability amplitudes of the solutions with objective function values below the threshold can be sufficiently amplified in certain search loops.

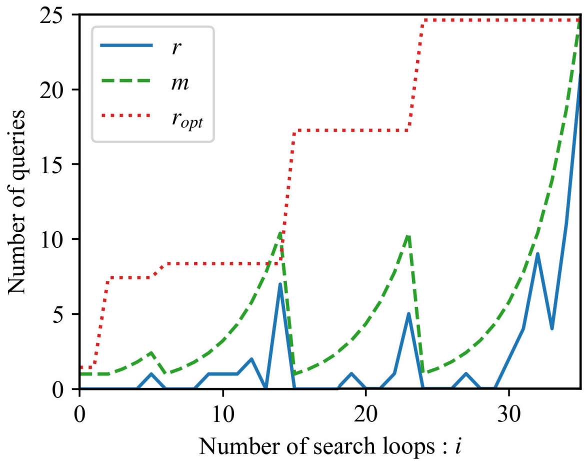

FIGURE 1. Transition of three values in each search loop of the GAS simulation using the standard method: selected number of queries (blue solid line), upper limit (green dashed line), and ideal number of queries (red dotted line). The gap between the ideal and selected numbers widens with each reset of the upper limit.

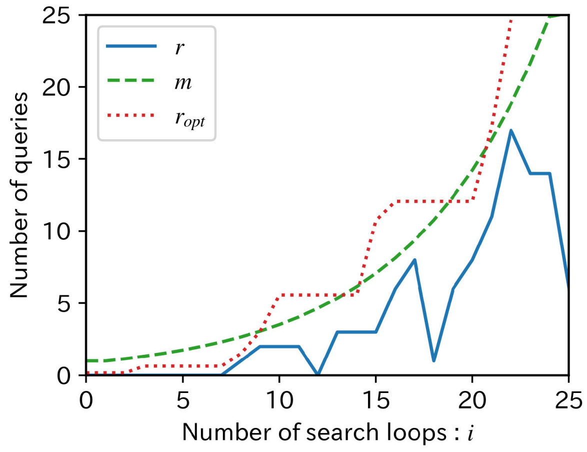

FIGURE 2. Transition of three values in each search loop of the GAS simulation using the proposed method: selected number of queries (blue solid line), upper limit (green dashed line), and ideal number of queries (red dotted line). The selected number closely tracks the ideal number better than the standard method.

Whenever the threshold is updated, the ideal number of queries increases. This is because the number of solutions with objective function values below the threshold decreases as the threshold decreases, which corresponds to a decrease in t in Equation (11). Therefore, resetting the upper limit widens the difference between m and r_{opt}, and it is speculated that the amplification process in some search loops is wasted. Accordingly, in this study, we propose a new method that eliminates resetting of the upper limit as in the above method.

Figure 2 presents an example of the transition of the number of queries using the proposed method. We focus on the red dotted line representing the ideal number of queries and the green dashed line representing the upper limit value of the number of queries. In Figure 1, the difference between these lines may widen because of the reset; however, in Figure 2, the difference remains small. Thus, the proposed method improves the tracking performance to r_{opt} and efficiently calls the oracle, which is expected to reduce the total number of queries (i.e., the query complexity) in the algorithm. The proof of the quadratic speedup capability of our method is provided in Appendix A.

Alternative methods have been proposed, such as linearly increasing the upper limit while keeping the reset of the upper bound intact [29] and selecting the number of queries according to a pre-optimized pattern [30]. These methods are compared in Section IV.

B. TERMINATION CONDITION

The termination condition for GAS occurs when the cumulative number of queries exceeds $22.5\sqrt{N}$. The value $22.5\sqrt{N}$ is set based on the upper limit of the expected number of queries required to find the optimal solution [21]. Therefore, the probability of finding an optimal solution before the termination condition is satisfied is sufficiently high. In this study, we propose a termination condition by considering the simulation results of the proposed method in Section V. This avoids redundant processes, as mentioned previously, and is expected to realize an efficient GAS that can terminate a search in a shorter time.

IV. NUMERICAL EXPERIMENTS AND ANALYSIS

In this section, we analyze the control methods described in Section III-A from the perspectives of convergence to optimal solutions and the number of queries. In Section IV-A, we introduce a method for faster and more memory-efficient GAS simulations. In Section IV-B, we introduce the combinatorial optimization problems used to assess the performance. In Section IV-C, we evaluate the performance of the proposed method in comparison with those of conventional methods. In Section IV-D, we compare the proposed and conventional methods as well as previous research methods through several combinatorial optimization problems and clarify the superiority of the proposed method. Finally, in Section IV-E, we demonstrate that the proposed method is effective for various problem scales.

A. GAS SIMULATION

Quantum circuits cannot be simulated efficiently using classical computations [36], [37]. However, it is also known that certain quantum circuits that are shallow and have low entanglement can be simulated by tensor network methods, which can significantly reduce the simulation costs in certain cases [38], [39]. However, this approach is not effective for quantum circuits in GAS, as they exponentially increase in depth with the number of bits, and the oracle increases entanglement.

Here, we devised a new simulation method for GAS, as shown in Algorithm 2, to clarify the scalability of the GAS performance with respect to the number of bits and to obtain sufficient data for quantitatively evaluating the performance through rapid simulations.

Algorithm 2 GAS Simulation

```
Require: Generator of the number of queries: $A$
 1: Randomly choose initial rank $\rho_0$ from $\{1, \ldots, N\}$
 2: Set $m = 1, i = 0, R = 0$
 3: repeat
 4:     $r_i = A(\boldsymbol{b})$
 5:     $R = R + r_i$
 6:     $\theta = \arcsin\sqrt{\frac{\rho_i - 1}{N}}$
 7:     Sample $b_i$ from Bernoulli distribution with parameter
        $\sin^2((2r+1)\theta)$
 8:     Append $b_i$ into $\boldsymbol{L}$
 9:     if $b_i = 1$ then
10:         Randomly choose $\rho_{i+1}$ from $\{1, \ldots, \rho_i - 1\}$
11:     else
12:         $\rho_{i+1} = \rho_i$
13:     end if
14:     $i = i + 1$
15: until $R > 22.5\sqrt{N}$ or $\rho_i = 1$
16: return $\boldsymbol{\rho}, \boldsymbol{r}$
```

In this simulation method, a "rank" is introduced to represent the order relating to the magnitude of the objective function values. We assigned ranks from 1 to $N = 2^n$ to each bit string of the candidate solutions, considering 1 as the rank of the optimal solution. If the same function value is obtained from two or more inputs, sequential numbers are assigned for convenience (e.g., for $f(x_1, x_2) = x_1 + x_2$, 2 (3) is assigned $x = (1, 0)$ ($x = (0, 1)$). When simulating various control methods for the number of queries, the generator A should be designed for each method.

The initial rank ρ_0 is randomly selected from integers ranging from 1 to N, and the lowest rank among the candidate solutions sampled up to the i-th search loop is denoted as ρ_i. In this notation, in the $(i+1)$-th search loop, candidate solutions with ranks lower than ρ_i are the targets of the GS. From Equation (8), the probability of observing any of the solutions and updating the threshold is $\sin^2((2r+1)\theta)$.

This behavior of GAS is simulated in Algorithm 2 through the use of a binary random variable b_i. This variable takes the value of 1 with a probability of $\sin^2((2r+1)\theta)$, corresponding to a successful search, and the value of 0 with a probability of $\cos^2((2r+1)\theta)$, corresponding to a failed search. Upon a successful search ($b_i = 1$), ρ_{i+1} is updated by selecting an integer randomly within the range 1 to ρ_i.

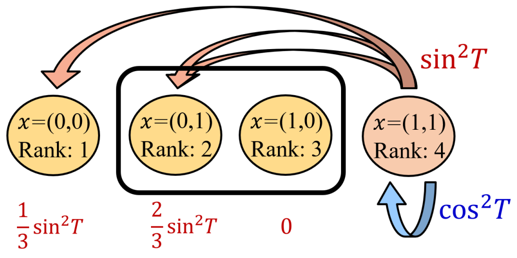

FIGURE 3. Rank transition for $f(x) = x_0 + x_1$ with $\rho_0 = 4$ and $T = (2r+1)\theta$.

Conversely, if the search fails ($b_i = 0$), ρ_{i+1} is set to retain the value of ρ_i.

The probability that $\rho_i = 1$ is referred to as the success probability in the i-th search loop. In Section IV-C, we compare the methods by examining the relationship between the cumulative number of queries and the success probability.

Algorithm 2 corresponds to the case of finding the optimal solution using GAS for a problem such as $f(\boldsymbol{x}) = \sum_{k=0}^{N-1} 2^k x_k$, where the input bit string and function values have one-to-one correspondence. We recognize this as a problem with a uniform rank distribution. When some inputs yield the same function value, the selection method for rank ρ_{i+1} must be modified in the subsequent search loop in line 10 of Algorithm 2 as follows. Groups are formed with inputs having the same function values, and the lowest rank within that group is set as the rank of the group. Subsequently, when the rank of the input within a group is selected, ρ_{i+1} becomes the rank of that group. Thus, the next search loop targets the inputs with a lower function value than that of the group. Note that the uniform rank distribution is the worst case to solve because we can regard the rank transition to a rank group as skipping some ranks in the group. Thus, performance evaluation using a problem with a uniform rank distribution guarantees that given the same problem size, any other problem can be solved with fewer queries than that required for the problem with a uniform rank distribution.

Consider the example of the function $f(\boldsymbol{x}) = x_0 + x_1$ with an initial rank of $\rho_0 = 4$. Figure 3 illustrates the rank transition and its probability in the first search loop. The circles represent the inputs and the numbers inside the circles indicate their ranks. Because $(x_0, x_1) = (0, 1), (1, 0)$ belongs to a group with a function value of 1, $\rho_1 = 2$ regardless of the selected rank. To understand this group structure, information is needed about the objective function values of all inputs. Therefore, Algorithm 2 can only be used for problems that allow for a brute-force search. However, unlike the simulation using state vector calculations, which requires matrix operations of size $N \times N$, Algorithm 2 involves only simple probabilistic processes, allowing a faster and more memory-efficient GAS behavior simulation.

B. EXAMPLES OF COMBINATORIAL PROBLEMS

When solving combinatorial problems with the GAS, they are mapped to QUBO or HUBO, which represents the objective functions to minimize. In this subsection, three combinatorial problems and their QUBO form are introduced to evaluate the performance of GAS in the subsequent subsections.

1) TRAVELING SALESMAN PROBLEM (TSP)

The traveling salesman problem focuses on finding the most efficient route for a salesman visiting a set of cities exactly once before returning to the starting point. The goal is to minimize the total distance traveled.

$$f(\boldsymbol{x}) = \sum_{i,j} d_{ij} x_i^t x_j^{t+1} + A \sum_i \left(\sum_t x_i^t - 1 \right)^2 + A \sum_t \left(\sum_i x_i^t - 1 \right)^2, \tag{13}$$

where d_{ij} is the distance between cities i and j, and x_i^t is a binary variable that takes the value 1 when city i is visited at time t. The first term represents the travel distance, the second represents the constraint of visiting each city only once, and the third represents the constraint of not visiting multiple cities simultaneously. Figure 4 shows a plot of the rank distribution for

$$d_{ij} = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}, \quad A = 2. \tag{14}$$

The horizontal axis represents the index when the input groups with the same function values are sorted in the order of the function values (1 represents the optimal solution), and the vertical axis represents the number of inputs in each group.

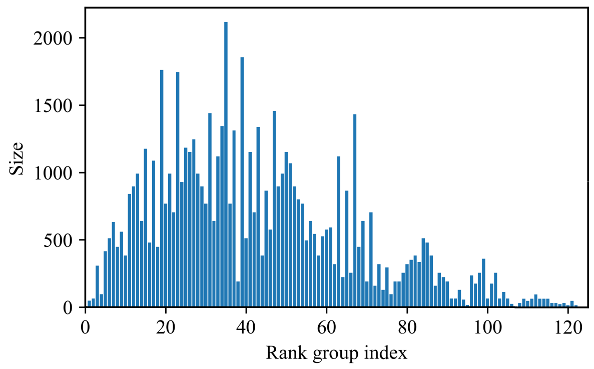

FIGURE 4. TSP rank distribution.

2) MAXCUT PROBLEM

The maxcut problem relates to the division of nodes of an undirected graph into two clusters such that the number of edges (or the total weight of the edges) spanning between the clusters is maximized. The objective function can be expressed as follows.

$$f(z) = -\sum_{i<j} \frac{1}{2} w_{ij}(1 - z_i z_j),$$

where $z_i \in \{-1, 1\}$, and w_{ij} is the weight of the edge connecting the nodes (i, j). In the simulations in the subsequent sections, w_{ij} is a randomly chosen integer.

3) NUMBER PARTITION PROBLEM (NPP)

The number partition problem involves dividing integers into two sets such that the sum of the numbers in each set is equal. The objective function can be represented as follows, where n_i is the i-th number.

$$f(z) = \left(\sum_i n_i z_i \right)^2,$$

where $z_i \in \{-1, 1\}$, and n_i is the i-th integer number. In the simulations in the subsequent sections, n_i is a randomly chosen integer.

C. EVALUATION OF THE PROPOSED METHOD

First, to determine the optimal value of λ for the proposed method without resets, we conducted simulations with 10,000 shots while changing λ in increments of 0.05 from 1.05 to 1.5 for uniform-rank-distribution problems with a bit size of $n = 40$. The benchmark results are presented in Figure 5. The horizontal axis represents the normalized value of the cumulative number of queries with $\sqrt{N}(= \sqrt{2^{40}} = 2^{20})$, and the vertical axis represents the probability of obtaining the rank 1 optimal solution when the cumulative number of queries on the horizontal axis is used as the termination condition. The color of the graph represents the value of λ. In Figure 5, the data points in the upper left indicate that the optimal solution was obtained with fewer queries. By comparing λ based on the cumulative number of queries when the success probability reached 99%, the best performance was found to be achieved for $\lambda = 1.10$.

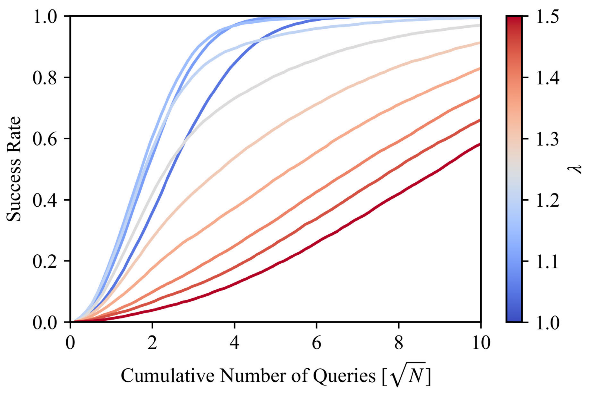

FIGURE 5. Amplification factor λ of the upper bound and solution performance.

TABLE 1. Normalized number of queries to obtain the optimal solution at 99% among λ. The numbers are normalized by column (problem). The best values for each method are highlighted in green.

Method	λ	Uniform	NPP	Maxcut
Proposed	1.05	1.34	1.32	1.35
	1.10	1.00	1.00	1.00
	1.15	1.17	1.22	1.05
	1.20	1.94	≥ 2	1.78
	≥ 1.25	≥ 2	≥ 2	≥ 2
Standard	1.05	≥ 2	1.69	≥ 2
	1.10	1.83	1.36	1.55
	1.15	1.54	1.23	1.39
	1.20	1.43	1.19	1.26
	1.25	1.34	1.15	1.19
	1.30	1.29	1.35	1.18
	1.35	1.29	1.56	1.22
	1.40	1.29	1.86	1.29
	1.45	1.29	≥ 2	1.27
	1.50	1.29	≥ 2	1.34

In addition to the problem of a 40-bit uniform rank distribution, the parameter λ was evaluated for a 20-bit NPP and maxcut problem. Furthermore, the standard method of Algorithm 1 was evaluated in the same way as followed for the proposed method. These evaluation results are shown in Table 1. In the case of $\lambda = 1.10$, which showed the best performance with the proposed method, the values in the table have been normalized for each column (problem) in order such that the number of cumulative queries required to achieve an optimal solution with a 99% probability becomes 1.00. According to [30], the standard method shows the best performance at $\lambda = 1.34$. However, according to the evaluation metric in Table 1, $\lambda = 1.30$ exhibits better performance. Additionally, the proposed method consistently achieves the best results at $\lambda = 1.10$, regardless of the problem.

Based on the above results, we compared the performances of Algorithm 1 using $\lambda = 1.30$ and the proposed method using $\lambda = 1.10$. The results are presented in Figure 6. In the figure, the orange solid line on the upper left side represents the numerical data of the proposed method. The number of queries required to obtain the optimal solution with 99% probability in the proposed method was approximately 22% less than that in the standard method. This provides evidence that the computational effort can be reduced by 22% while maintaining the quality required of the solution, which demonstrates the superiority of the proposed method.

In the following evaluations, the number of shots was fixed at 10,000 for each condition, and λ, the increment coefficient of the upper limit, was fixed at the optimal values.

D. PERFORMANCE COMPARISON WITH PREVIOUS STUDIES

The method that linearly increases the upper bound is denoted as "Linear" [29], which replaces $m = \lambda m$ in line 9 of Algorithm 1 with $m = m + \lambda$. The method that calculates the probability distribution of rank ρ for the uniform rank distribution and pre-generates the optimal sequence of the number of queries for GAS is denoted as "Sequence" [30].

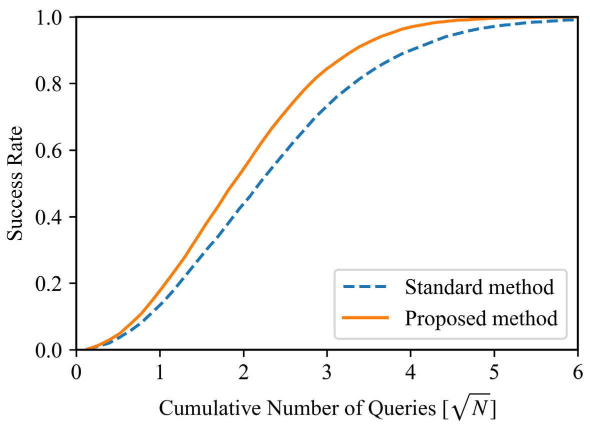

FIGURE 6. Comparison of solution performance.

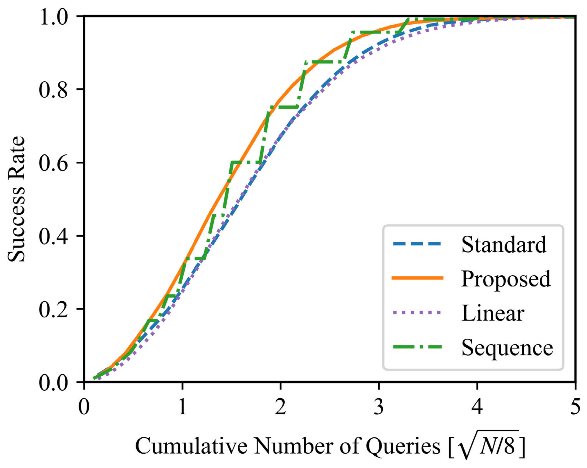

FIGURE 7. Comparison with the proposed method using 4-point TSP.

In methods other than the Sequence, the time complexity to obtain the number of queries in each search loop is $\mathcal{O}(1)$, because of the update of the upper limit and generation of random numbers. However, in Sequence, to determine the best number of queries in the k-th search loop, the change in the probability distribution of the solution is comprehensively evaluated for different numbers of queries. The number of queries in GAS is proportional to $\sqrt{N}$; thus, the number of evaluations is of the same order $\Theta(\sqrt{N})$. Owing to the decay of the change in the probability distribution and error propagation in the k-th repeated integration of the probability distribution, the time complexity necessary for a reasonable integration evaluation amounts to at least a polynomial $P(k)$. Hence, an issue arises wherein the time complexity required for generating the sequence of query numbers is $\Omega(\sqrt{N}P(k))$, resulting in a larger time complexity than that of GAS. Considering this issue, numerical simulations for performance comparison were conducted on a small-scale TSP.

Figure 7 presents the results of comparing the control methods using TSP introduced in Section IV-B. The normalization factor of the horizontal axis is $\sqrt{N/8}$ because the number of optimal solutions is eight. The results for the best

TABLE 2. Normalized number of queries to obtain the optimal solution with 99%. The numbers are normalized by column (problem) in order such that the number for the proposed method is 1.

Problem	TSP	NPP	Maxcut
Standard	1.12	1.15	1.26
Proposed	1.00	1.00	1.00
Linear	1.15	1.17	2.03

λ are shown in Figure 7. The proposed method consistently achieved a higher success probability than the standard and linear methods. In the case of the Sequence, the solution outperformed the proposed method in some areas. However, due to the calculation cost for the best number of queries, it is difficult to apply the Sequence to larger-scale problems.

Furthermore, Table 2 presents the cumulative number of queries required to secure the optimal solution with a 99% probability for three problems, using normalization where the proposed method's value is set to one. Regarding the Sequence, there are cases where the success rate did not reach the 99% in the range of the query sequence generated by us; therefore, it is not included in the table. The proposed method consistently achieves the 99% success rate with the least number of queries for all problems. These results highlight the performance of the proposed method and demonstrate its capability to reduce the solution search time by at least 10%, even in the case of small-scale problems.

E. SCALABILITY OF PERFORMANCE WITH RESPECT TO THE NUMBER OF BITS

Next, we explain the results of investigating the performance change of the proposed method when the problem bit size n is varied. Figure 8 compares the performance on problems with a uniform rank distribution. The data were measured while increasing n, the number of bits, from 10 to 60 in increments of 5, with λ fixed at 1.10. The horizontal axis represents the cumulative number of queries R normalized by $\sqrt{N} = \sqrt{2^n}$. Thus, the scale of the horizontal axis differs for each n. From 20 bits upward, it was confirmed that there was almost no change in the success probability with respect to $\sqrt{N}$, which indicates that the performance of the proposed method is invariant. Therefore, we conclude that the proposed method can be applied to large-scale problems.

V. DISCUSSION

A. TERMINATION CONDITION

In Algorithm 1, a termination condition occurs when the number of oracle calls R exceeds $22.5\sqrt{N}$. From the described results, this termination condition leads to a redundant algorithm that continues to call the oracle for amplification, even though the success rate is nearly 1 when executing GAS.

Considering this and the method proposed in Section III-A, an improved version of Algorithm 1 is presented as Algorithm 3. The termination condition $4.55\sqrt{N}$ represents the cumulative number of queries that would produce an optimal solution with 99% probability when solving a problem with a uniform rank distribution of 40 bits. Furthermore, considering the relationship between the cumulative number of queries and the average rank of the obtained solution, as shown in Figure 9, if $\sqrt{N}$ is set as the termination condition, on average, the seventh-best solution can be obtained. Therefore, the termination condition can be set smaller if a solution close to the optimal solution is acceptable.

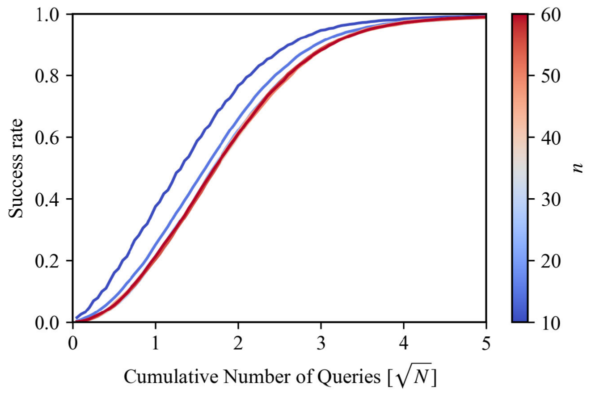

FIGURE 8. Performance vs. number of quantum bits.

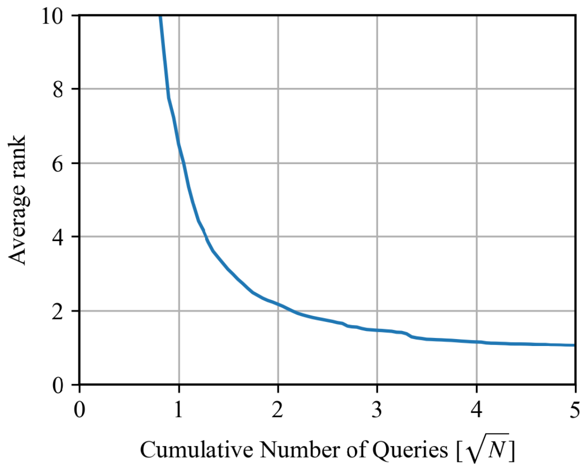

FIGURE 9. Rank progression of the 40-bit problem with uniform rank distribution.

B. FUTURE CHALLENGES

In Sections III and IV, we discussed the issue of the standard method for selecting the number of queries in GAS and confirmed through numerical simulations that the proposed method can find the optimal solution with fewer queries. However, this research has potential for further refinement and expansion.

Firstly, methods other than the trial-and-error method require further consideration. As mentioned in Section I, three approaches, namely trial-and-error, quantum counting, and fixed-point quantum search, were identified for addressing the "soufflé problem" in GAS. The simulations

Algorithm 3 GAS suggested in this research

```
Require: $f : \{0,1\}^n \to \mathbb{R}$
1:  Uniformly sample $x_1 \in \{0,1\}^n$
2:  Set $y_1 = f(x_1)$, $m = 1$, $i = 1$, $R = 0$, $\lambda = 1.10$
3:  repeat
4:      Randomly choose $r$ from $\{0, 1, \dots, \lceil m-1 \rceil\}$ and $R = R + r$
5:      Execute GS using $r$ iterations with oracle of $\hat{O}_{y_i}$, and obtain sampling results $x$ and $y = f(x)$
6:      if $y < y_i$ then
7:          $x_{i+1} = x$, $y_{i+1} = y$
8:      else
9:          $x_{i+1} = x_i$, $y_{i+1} = y_i$
10:     end if
11:     $m = \lambda m$
12:     $i = i + 1$
13: until $R > 4.55\sqrt{N}$
14: return $x_i$
```

in this study were based on the trial-and-error method, and we improved the method of selecting the number of queries. In addition, we compared quantum counting with our proposed method in Appendix B and confirmed that GAS using our method is superior. Future research should evaluate the computational cost of GAS using fixed-point quantum search [40].

Next, the proposed method could be improved and investigated. In Appendix A, we analytically showed that GAS using the proposed method achieves the same quadratic speedup as the original GAS. However, the scope of this study was limited to the numerical demonstration of the ability of the proposed method to reduce the constant overhead of query complexity when compared with the conventional method. To determine the superiority of the proposed method, an analytical demonstration can be considered in future research. Additionally, research has been reported on GS for databases with a prior distribution [41]. Similarly, by adopting a Bayesian approach with the prior information of combinatorial optimization problems and update of y, it might be possible to further reduce the constant overhead of GAS. This aspect also requires further investigation.

VI. CONCLUSION

In this study on GAS, we examined the number of queries required to solve the QUBO and HUBO problems and proposed a new method to control the number of queries in each loop by removing the upper limit reset. We compared the results of GAS simulations and confirmed the superiority of the proposed method. Moreover, by conducting simulations with different numbers of bits, we verified that the proposed method is effective, even for large-scale problems. Furthermore, based on the simulation results, we proposed a modified GAS framework in Algorithm 3. This framework is applicable when solving QUBO and HUBO problems with GAS and is expected to provide optimal solutions more quickly.

APPENDIX A
PROOF OF QUADRATIC SPEEDUP OF PROPOSED METHOD

Lemma 1: For any natural number n and real numbers α, β,

$$\sum_{k=0}^{n-1} \cos(\alpha + 2k\beta) = \frac{\sin n\beta \cos(\alpha + (n-1)\beta)}{\sin\beta}. \tag{15}$$

Proof: Let $C = \sum_{k=0}^{n-1} \cos(\alpha + 2k\beta)$.

$$\begin{aligned} 2C\sin\beta &= \sum_{k=0}^{n-1} 2\cos(\alpha + 2k\beta)\sin\beta \\ &= \sum_{k=0}^{n-1} \sin(\alpha + (2k+1)\beta) - \sin(\alpha + (2k-1)\beta) \\ &= \sin(\alpha + (2n-1)\beta) - \sin(\alpha - \beta) \\ &= 2\cos(\alpha + (n-1)\beta)\sin(n\beta). \end{aligned}$$

Therefore,

$$C = \frac{\sin n\beta \cos(\alpha + (n-1)\beta)}{\sin\beta}.$$

■

Lemma 2: Consider a search space with size N and number of targets t. Let $\theta = \arcsin\sqrt{t/N}$. When performing GS once, we uniformly select the number of queries from 0 to $m-1$.

$$P_m^\theta = \frac{1}{2} - \frac{\sin 4m\theta}{4m\sin 2\theta},$$

where P_m^θ is the probability of observing one of the search targets.

Proof: In Lemma 1, when $\alpha = \beta = 2\theta$, it follows that

$$\begin{aligned} \sum_{k=0}^{n-1} \cos(2k+1)2\theta &= \frac{\sin 2n\theta \cos 2n\theta}{\sin 2\theta} \\ &= \frac{\sin 4n\theta}{2\sin 2\theta}. \end{aligned} \tag{16}$$

The probability of observing a search target at the number of queries j is $\sin^2(2j+1)\theta$, and the probability of selecting j as the number of queries is $1/m$. Thus,

$$\begin{aligned} P_m^\theta &= \sum_{j=0}^{m-1} \frac{1}{m} \times \sin^2((2j+1)\theta) \\ &= \frac{1}{2m} \sum_{j=0}^{m-1} [1 - \cos((2j+1)2\theta)] \\ &= \frac{1}{2} - \frac{\sin 4m\theta}{4m\sin 2\theta}. \end{aligned}$$

Note that the transformation from the second to the third line uses Equation (16). ■

Theorem 3: Let $m_0 = 1/\sin 2\theta$. The search loops for $m > m_0$ are referred to as the critical stage. During the critical stage,

$$\frac{1}{4} < P_m^\theta < \frac{3}{4}. \tag{17}$$

Proof: The result follows immediately from Lemma 2. ■

GAS is a quantum algorithm designed to determine the input $\boldsymbol{x} \in \{0,1\}^n$ that yields the minimum value of the objective function $f : \{0,1\}^n \to \mathbb{R}$ (i.e., the optimal solution). Until the termination conditions are satisfied, as shown in Algorithms 1 and 3, the GS is repeated for r iterations. r is selected as an integer between 0 and m. m starts from 1 and is multiplied by λ in each loop. In Algorithm 1, m is initialized to 1 when the threshold y is updated. In the following discussion on query complexity, we consider an algorithm that continues indefinitely. Furthermore, we assume that the problem has a uniform rank distribution (details are provided in Section IV-A), which is the worst case in terms of query complexity.

Definition 4: We denote the number of search loops (i.e., lines 3-11 in Algorithm 1) as s. Without considering the reset of m, in the s-th search loop, $m(s) = \lambda^{s-1}$ holds. Furthermore, a "rank" is assigned to the inputs of the objective function, representing the order of the function value. With this assignment in place, the rank of the optimal solution is 1. We refer to the rank of the input x that provides the smallest objective function value obtained up to a certain search loop as the provisional rank. If the search continues sufficiently, the provisional rank becomes 1. The history of provisional ranks followed up to rank 1 is termed the rank path P, and the probability that the rank path is P is denoted by $q(P)$.

For instance, the search space of $N = 4$ and rank path of $\{4,2\}$ imply that after obtaining a solution of rank 4 in the initial candidate sampling (i.e., line 1 in Algorithm 1), the search loop is repeated to obtain the inputs in the descending order of the rank (2,1). Because the state amplified by GS is a uniformly weighted superposition of the search targets, the next provisional rank is uniformly selected from the integer values that are less than the current provisional rank. Therefore,

$$q(\{4,2\}) = \frac{1}{4}\cdot\frac{1}{3}\cdot\frac{1}{1} = \frac{1}{12}.$$

Definition 5: Let the expected number of queries required to update a provisional rank $r \in P$ for a rank path P be denoted by $O_1^P(r)$ and $O_2^P(r)$. Indices 1 and 2 refer to the conventional and proposed methods, respectively. $E_{1,2}$, the expected number of queries necessary to obtain the optimal solution in the GAS, is given by

$$E_{1,2} = \sum_{r=2}^{N}\sum_{P\ni r} q(P)O_{1,2}^P(r). \tag{18}$$

The sum $\sum_{P\ni r}$ denotes the summation over all the rank paths containing rank r.

Lemma 6: [Section IV in [26]] For the provisional rank r, let $\theta_r = \arcsin\sqrt{r-1/N}$ and $m_0^r = 1/\sin 2\theta_r$. Then, for $1 < \lambda < 4/3$,

$$O_1^P(r) < \frac{m_0^r\lambda}{2}\frac{1}{\lambda-1} + \frac{m_0^r\lambda}{2}\frac{4}{4-3\lambda}. \tag{19}$$

Proof: Let $\lfloor\cdot\rfloor$ denote the floor function, which is a mathematical function that takes a real number x and provides the greatest integer less than or equal to x. When the conventional method is used, the expected number of queries required to reach the critical stage is

$$\begin{aligned}\sum_{s=1}^{\lfloor\log_\lambda m_0^r\rfloor}\sum_{j=0}^{\lfloor m(s)\rfloor}\frac{j}{\lfloor m(s)\rfloor+1} &= \frac{1}{2}\sum_{s=1}^{\lfloor\log_\lambda m_0^r\rfloor}\lfloor m(s)\rfloor\\ &< \frac{1}{2}\sum_{s=1}^{\lfloor\log_\lambda m_0^r\rfloor}\lambda^{s-1}\\ &< \frac{1}{2}\frac{m_0^r\lambda-1}{\lambda-1}\\ &< \frac{1}{2}\frac{m_0^r\lambda}{\lambda-1}.\end{aligned} \tag{20}$$

The expected number of queries required to succeed in the search and update the rank after reaching the critical stage is

$$\begin{aligned}&\sum_{s=\lfloor\log_\lambda m_0^r\rfloor+1}^{\infty}\sum_{j=0}^{\lfloor m(s)\rfloor}\left\{\frac{j}{\lfloor m(s)\rfloor+1}\sin^2((2j+1)\theta_r)\right.\\ &\qquad\left.\prod_{s'=1}^{s-1}(1-P^{\theta_r}_{\lfloor m(s')\rfloor+1})\right\}\\ &<\sum_{s=1}^{\infty}\sum_{j=0}^{\lfloor m(s)\rfloor}\frac{j}{\lfloor m(s)\rfloor+1}\prod_{s'=1}^{s-1}(1-P^{\theta_r}_{\lfloor\lambda^{s'}\rfloor})\\ &<\frac{1}{2}\sum_{u=1}^{\infty}\lambda^{\lfloor\log_\lambda m_0^r\rfloor+u}\prod_{u'=1}^{u-1}\frac{3}{4}\qquad(\because \text{Theorem } 3)\\ &=\frac{m_0^r\lambda}{2}\sum_{u=1}^{\infty}\left(\frac{3\lambda}{4}\right)^{u-1}\\ &=\frac{2m_0^r\lambda}{4-3\lambda}.\end{aligned} \tag{21}$$

By adding Equations (20) and (21), we obtain Equation (19). ■

Theorem 7 (Quadratic Speedup by the Conventional Method): For $1 < \lambda < 4/3$,

$$\begin{aligned}E_1 &< \frac{5\lambda}{8}\left(\frac{1}{\lambda-1}+\frac{4}{4-3\lambda}\right)\sqrt{N} &(22)\\ &= \frac{45}{4}\sqrt{N}\quad(\text{when } \lambda = \frac{6}{5}). &(23)\end{aligned}$$

Proof: From Lemma 6, we have

$$E_1 < \frac{\lambda}{2}\left(\frac{1}{\lambda-1}+\frac{4}{4-3\lambda}\right)\sum_{r=2}^{N} m_0^r\left(\sum_{P\ni r} q(P)\right).$$

According to Lemma 1 in [21], the probability of a rank path containing r is $1/r$.

$$\sum_{P \ni r} q(P) = \frac{1}{r} \tag{24}$$

Hence,

$$\begin{aligned}
E_1 &< \frac{\lambda}{2}\left(\frac{1}{\lambda-1} + \frac{4}{4-3\lambda}\right)\sum_{r=2}^{N}\frac{1}{\sin 2\theta_r}\frac{1}{r} \\
&< \frac{\lambda\sqrt{N}}{4}\left(\frac{1}{\lambda-1} + \frac{4}{4-3\lambda}\right)\sum_{r=2}^{N}\frac{1}{\sqrt{r-1}}\frac{1}{r} \\
&< \frac{\lambda\sqrt{N}}{4}\left(\frac{1}{\lambda-1} + \frac{4}{4-3\lambda}\right)\left(\frac{1}{2} + \int_1^{N-1} r^{-\frac{3}{2}} dr\right) \\
&< \frac{5\lambda\sqrt{N}}{8}\left(\frac{1}{\lambda-1} + \frac{4}{4-3\lambda}\right) \\
&= \frac{45}{4}\sqrt{N} \quad (\text{when } \lambda = \frac{6}{5}).
\end{aligned}$$

■

In contrast, in the proposed method, which does not reset the upper limit, the upper limit changes depending on the number of loops required for the provisional rank to reach r. Therefore, the number of queries required to update a provisional rank r depends on the rank path P. Consequently, Equation (24) cannot be used. We define a new symbol to consider this effect.

Definition 8: Let s_b denote the number of loops when the provisional rank is r and let r_b denote the rank before r in path P (i.e., $P = \{\ldots, r_b, r, \ldots\}$). In this case,

$$\begin{aligned}
O_2^P(r) = \sum_{s_b=1}^{\infty} a_r^P(s_b) \sum_{s=s_b+1}^{\infty} \sum_{j=0}^{\lfloor m(s) \rfloor} \Bigg\{ &\frac{j}{\lfloor m(s) \rfloor + 1} \\
&\sin^2((2j+1)\theta_r) \prod_{s'=s_b+1}^{s-1} (1 - P_{\lfloor \lambda^{s} \rfloor}^{\theta_r}) \Bigg\}.
\end{aligned} \tag{25}$$

Here, $a_r^P(s_b)$ is the probability distribution function of s_b when rank r is reached in the rank path P.

Definition 9: Let $c(r)$ denote the starting point of the critical stage for the rank r.

$$c(r) = 2 - \lfloor \log_\lambda \sin 2\theta_r \rfloor \quad (\theta_r = \arcsin\sqrt{(r-1)/N}) \tag{26}$$

Lemma 10: When $s_b > c(r)$, the following holds:

$$\begin{aligned}
\sum_{s_b=1}^{\infty} a_r^P(s_b) \sum_{s=s_b+1}^{\infty} \sum_{j=0}^{\lfloor m(s) \rfloor} \Bigg\{ &\frac{j}{\lfloor m(s) \rfloor + 1} \\
&\sin^2((2j+1)\theta_r) \prod_{s'=s_b+1}^{s-1} (1 - P_{\lfloor \lambda^{s} \rfloor}^{\theta_r}) \Bigg\} < \frac{2\lambda^{s_b}}{4-3\lambda}.
\end{aligned} \tag{27}$$

Proof:

(Left Hand Side)

$$\begin{aligned}
&< \sum_{s=s_b+1}^{\infty} \sum_{j=1}^{\lfloor m(s) \rfloor} \frac{j}{\lfloor m(s) \rfloor + 1} \prod_{s'=s_b+1}^{s-1} (1 - P_{\lfloor \lambda^{s} \rfloor}^{\theta_r}) \\
&= \frac{1}{2} \sum_{s=s_b+1}^{\infty} \lfloor m(s) \rfloor \prod_{s'=s_b+1}^{s-1} (1 - P_{\lfloor \lambda^{s} \rfloor}^{\theta_r}) \\
&< \frac{1}{2} \sum_{s=s_b+1}^{\infty} \lambda^{s-1} \left(\frac{3}{4}\right)^{s-s_b-1} \\
&\qquad (\because \text{Theorem 3, } s_b > c(r)) \\
&= \frac{\lambda^{s_b}}{2} \sum_{u=1}^{\infty} \left(\frac{3\lambda}{4}\right)^{u-1} \\
&= \frac{2\lambda^{s_b}}{4-3\lambda}
\end{aligned}$$

■

Lemma 11: When $s_b > c(r_b)$, the following holds:

$$a_r^P(s_b) < \left(\frac{3}{4}\right)^{s_b - c(r_b) + 1}. \tag{28}$$

Proof: Consider the search loop between $c(r_b)$ and s_b. Because the provisional ranks in the rank path before r_b assume values greater than r_b, the search from loop count $c(r_b)$ to s_b is always in the critical stage. From this fact and Lemma 3, the probability of a successful search that updates the provisional rank, as well as the probability of an unsuccessful search where the provisional rank remains unchanged, is each less than $3/4$.

However, such searches are conducted $s_b - c(r_b) + 1$ times until the provisional rank reaches r. Therefore, in the rank path P, the probability that the provisional rank becomes r exactly at the s_b-th search is less than the value obtained by multiplying $3/4$ $s_b - c(r_b) + 1$ times, which demonstrates the validity of this lemma. ■

Theorem 12 (Quadratic Acceleration Property of the Proposed Method):

For $1 < \lambda < 4/3$, the following holds:

$$O_2^P(r) < \frac{\lambda^2(9\lambda - 8)}{2(\lambda-1)(4-3\lambda)^2} m_0^r, \tag{29}$$

$$E_2 < \frac{5\lambda^2(9\lambda - 8)}{8(\lambda-1)(4-3\lambda)^2}\sqrt{N}. \tag{30}$$

Proof:

$$\begin{aligned}
O_2^P(r) = \sum_{s_b=1}^{c(r)-1} a_r^P(s_b) \sum_{s=s_b+1}^{\infty} \sum_{j=0}^{\lfloor m(s) \rfloor} \Bigg\{ &\frac{j}{\lfloor m(s) \rfloor + 1} \\
&\sin^2((2j+1)\theta_r) \prod_{s'=s_b+1}^{s-1} (1 - P_{\lfloor \lambda^{s} \rfloor}^{\theta_r}) \Bigg\} \\
+ \sum_{s_b=c(r)}^{\infty} a_r^P(s_b) \sum_{s=s_b+1}^{\infty} \sum_{j=0}^{\lfloor m(s) \rfloor} \Bigg\{ &\frac{j}{\lfloor m(s) \rfloor + 1} \\
&\sin^2((2j+1)\theta_r) \prod_{s'=s_b+1}^{s-1} (1 - P_{\lfloor \lambda^{s} \rfloor}^{\theta_r}) \Bigg\}.
\end{aligned} \tag{31}$$

The first term represents the expected value in cases where the search loop count is less than $c(r)$ when the provisional rank reaches r. This is bounded above by $O_1^P(r)$, because this term can be seen as $O_1^P(r)$ with a reduced contribution from the first term in Equation (19). Thus,

$$\begin{aligned}(\text{First Term}) &< O_1^P(r) \\ &< \frac{m_0^r \lambda}{2}\frac{1}{\lambda-1} + \frac{m_0^r\lambda}{2}\frac{4}{4-3\lambda}. \end{aligned} \tag{32}$$

By utilizing Lemmas 10 and 11, the upper bound of the second term can be evaluated as

$$\begin{aligned}(\text{Second Term}) &< \sum_{s_b=c(r)}^{\infty}\left(\frac{3}{4}\right)^{s_b-c(r_b)+1}\cdot\frac{2\lambda^{s_b}}{4-3\lambda} \\ &= \frac{2}{4-3\lambda}\left(\frac{3}{4}\right)^{-c(r_b)+1}\sum_{s_b=c(r)}^{\infty}\left(\frac{3\lambda}{4}\right)^{s_b} \\ &= \frac{2}{4-3\lambda}\left(\frac{3}{4}\right)^{-c(r_b)+1}\sum_{u=1}^{\infty}\left(\frac{3\lambda}{4}\right)^{c(r)+u-1} \\ &= \frac{2\lambda^{c(r)}}{4-3\lambda}\left(\frac{3}{4}\right)^{c(r)-c(r_b)+1}\sum_{u=1}^{\infty}\left(\frac{3\lambda}{4}\right)^{u-1} \\ &= \frac{6\lambda^{c(r)}}{(4-3\lambda)^2}\left(\frac{3}{4}\right)^{c(r)-c(r_b)} \\ &< \frac{6m_0^r\lambda^2}{(4-3\lambda)^2}.\end{aligned}$$

By adding the first and second terms and organizing the results, Equation (29) holds.

Moreover, by denoting the coefficient of m_0^r in Equation (29) by A,

$$\begin{aligned}E_2 &= \sum_{r=2}^{N}\sum_{P\ni r} q(P)O_2^P(r) \\ &< A\sum_{r=2}^{N}\left(\sum_{P\ni r} q(P)\right)m_0^r \\ &< \frac{A\sqrt{N}}{2}\sum_{r=2}^{N}\frac{1}{r}\cdot\frac{1}{\sqrt{r-1}} \\ &< \frac{5A\sqrt{N}}{4} \\ &= \frac{5\lambda^2(9\lambda-8)}{8(\lambda-1)(4-3\lambda)^2}\sqrt{N}.\end{aligned} \tag{33}$$

Thus, Equation (30) also holds. ■

APPENDIX B
GAS WITH QUANTUM COUNTING

Here, we compare our method's performance with that of the GAS with quantum counting. Quantum counting is a quantum algorithm that estimates the ratio $\theta = \sqrt{\frac{t}{N}}$ to determine r_{opt} [27] and utilizes a controlled oracle. Although a controlled operation requires more gates than the original operation, we assume that their computational costs are the same, and the controlled oracle is counted as one oracle for comparison. In addition, we hypothesize that θ is obtained with certainty in a single sampling and that the number of ancilla bits n_a is determined based on the value of r_{opt}. These assumptions favor GAS with quantum counting. The details of this simplified simulation of a problem with a uniform rank distribution are presented in Algorithm 4, and the results of comparing its performance with the proposed method are shown in Figure 10.

Algorithm 4 GAS simulation with quantum counting
```
1:  Randomly choose initial rank $\rho_0$ from $\{1,\dots,N\}$
2:  Set $i = 0, R = 0$
3:  repeat
4:      $\theta = \arcsin\sqrt{\frac{\rho_i - 1}{N}}$
5:      $r_i = \left\lfloor \frac{1}{2}(\frac{\pi}{2\theta} + \frac{1}{2}) \right\rfloor$ and $n_a = \lceil \log_2(\pi/\theta) \rceil$
6:      $R = R + r_i + 2^{n_a} - 1$
7:      Sample $b_i$ from a Bernoulli distribution with the parameter $\sin^2((2r_i+1)\theta)$
8:      Append $b_i$ into $\boldsymbol{L}$
9:      if $b_i = 1$ then
10:         Randomly choose $\rho_{i+1}$ from $\{1,\dots,\rho_i - 1\}$
11:     else
12:         $\rho_{i+1} = \rho_i$
13:     end if
14:     $i = i + 1$
15: until $R > 22.5\sqrt{N}$ or $\rho_i = 1$
16: return $\boldsymbol{\rho}, R$
```

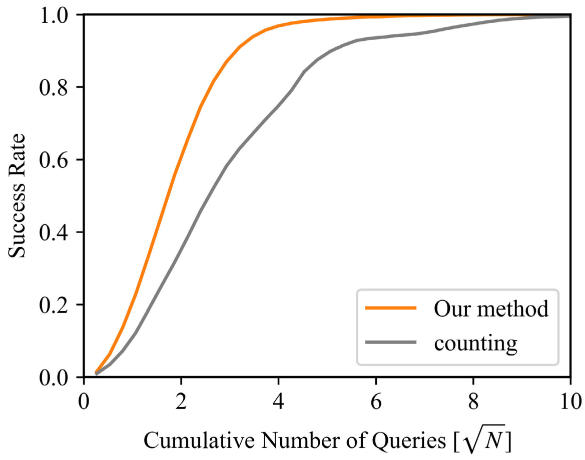

FIGURE 10. Proposed method vs. GAS with quantum counting.

Although the behavior of quantum counting is idealized to its advantage, the proposed method is located in the upper left corner, which indicates that the proposed method obtains an optimal solution in a shorter time. The cumulative number of queries required to achieve a 99% success rate was 38% less for the proposed method. Despite accurately selecting the ideal value r_{opt} as r through quantum counting, the performance of GAS with quantum counting is inferior.

The number of queries required for the estimation of θ and for Grover Search (GS) are of the same order, leading to the necessity for an increased cumulative number of queries when compared with that in the trial-and-error method where r is selected randomly. Without the idealization of quantum counting, a more computationally expensive control oracle is used for the estimation of θ. Therefore, the difference in the computational cost should be greater than that inferred from Figure 10.

REFERENCES

[1] T. Erdelić and T. Carić, "A survey on the electric vehicle routing problem: Variants and solution approaches," *J. Adv. Transp.*, vol. 2019, pp. 1–48, May 2019, doi: 10.1155/2019/5075671.

[2] M. Asghari and S. M. J. Mirzapour Al-e-hashem, "Green vehicle routing problem: A state-of-the-art review," *Int. J. Prod. Econ.*, vol. 231, Jan. 2021, Art. no. 107899, doi: 10.1016/j.ijpe.2020.107899.

[3] Z. Wang and J.-B. Sheu, "Vehicle routing problem with drones," *Transp. Res. B, Methodol.*, vol. 122, pp. 350–364, Apr. 2019, doi: 10.1016/j.trb.2019.03.005.

[4] T. Vidal, G. Laporte, and P. Matl, "A concise guide to existing and emerging vehicle routing problem variants," *Eur. J. Oper. Res.*, vol. 286, no. 2, pp. 401–416, Oct. 2020, doi: 10.1016/j.ejor.2019.10.010.

[5] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017, doi: 10.1109/TSMC.2016.2582745.

[6] J. J. Q. Yu, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3806–3817, Oct. 2019, doi: 10.1109/TITS.2019.2909109.

[7] X. Zhu, R. Yan, Z. Huang, W. Wei, J. Yang, and S. Kudratova, "Logistic optimization for multi depots loading capacitated electric vehicle routing problem from low carbon perspective," *IEEE Access*, vol. 8, pp. 31934–31947, 2020, doi: 10.1109/ACCESS.2020.2971220.

[8] Q. Li, Y. Liao, K. Wu, L. Zhang, J. Lin, M. Chen, J. M. Guerrero, and D. Abbott, "Parallel and distributed optimization method with constraint decomposition for energy management of microgrids," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 4627–4640, Nov. 2021, doi: 10.1109/TSG.2021.3097047.

[9] R. A. Shaikh, D. J. Vowles, A. Allison, and D. Abbott, "Evaluation of Australia's generation-storage requirements in a fully renewable grid with intermittent and flexible generation," *IEEE Access*, vol. 11, pp. 64201–64218, 2023, doi: 10.1109/ACCESS.2023.3286037.

[10] A. Chis and V. Koivunen, "Coalitional game-based cost optimization of energy portfolio in smart grid communities," *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 1960–1970, Mar. 2019, doi: 10.1109/TSG.2017.2784902.

[11] F. Black and R. Litterman, "Global portfolio optimization," *Financial Analysts J.*, vol. 48, no. 5, pp. 28–43, Sep. 1992, doi: 10.2469/faj.v48.n5.28.

[12] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 321–344, Jun. 2013, doi: 10.1109/TEVC.2012.2196800.

[13] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 58, no. 5, pp. 5355–5363, Nov. 1998, doi: 10.1103/physreve.58.5355.

[14] T. Ohyama, Y. Kawamoto, and N. Kato, "Resource allocation optimization by quantum computing for shared use of standalone IRS," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 4, pp. 950–961, Oct. 2023, doi: 10.1109/tetc.2023.3292355.

[15] T. Ohyama, Y. Kawamoto, and N. Kato, "Intelligent reflecting surface (IRS) allocation scheduling method using combinatorial optimization by quantum computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 3, pp. 1633–1644, Jul. 2022, doi: 10.1109/TETC.2021.3115107.

[16] T. Ohyama, Y. Kawamoto, and N. Kato, "Quantum computing based optimization for intelligent reflecting surface (IRS)-aided cell-free network," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 1, pp. 18–29, Jan. 2023, doi: 10.1109/TETC.2022.3161542.

[17] K. Hanakago, R. Takahashi, T. Ohyama, and F. Adachi, "User scheduling and clustering for distributed antenna network using quantum computing," *IEICE Trans. Commun.*, vol. E106.B, no. 11, pp. 1210–1218, Nov. 2023, doi: 10.1587/transcom.2023ebt0004.

[18] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Rev. Phys.*, vol. 4, Nov. 2019, Art. no. 100028, doi: 10.1016/j.revip.2019.100028.

[19] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Trans. Quantum Eng.*, vol. 1, pp. 1–24, 2020, doi: 10.1109/TQE.2020.3030314.

[20] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.

[21] C. Durr and P. Hoyer, "A quantum algorithm for finding the minimum," 1996, *arXiv:quant-ph/9607014*.

[22] N. Ishikawa, "Quantum speedup for index modulation," *IEEE Access*, vol. 9, pp. 111114–111124, 2021, doi: 10.1109/ACCESS.2021.3103207.

[23] J. Zhu, Y. Gao, H. Wang, T. Li, and H. Wu, "A realizable GAS-based quantum algorithm for traveling salesman problem," 2022, *arXiv:2212.02735*.

[24] Y. Sano, K. Mitarai, N. Yamamoto, and N. Ishikawa, "Accelerating Grover adaptive search: Qubit and gate count reduction strategies with higher-order formulations," 2023, *arXiv:2308.01572*.

[25] T. J. Yoder, G. H. Low, and I. L. Chuang, "Fixed-point quantum search with an optimal number of queries," *Phys. Rev. Lett.*, vol. 113, no. 21, Nov. 2014, Art. no. 210501, doi: 10.1103/physrevlett.113.210501.

[26] M. Boyer, G. Brassard, P. Hoeyer, and A. Tapp, "Tight bounds on quantum searching," 1996, *arXiv: quant-ph/9605034*.

[27] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemp. Math.*, vol. 305, pp. 53–74, Jun. 2002, doi: 10.1090/conm/305/05215.

[28] L. K. Grover, "Fixed-point quantum search," *Phys. Rev. Lett.*, vol. 95, no. 15, Oct. 2005, Art. no. 150501, doi: 10.1103/physrevlett.95.150501.

[29] L. Giuffrida, D. Volpe, G. A. Cirillo, M. Zamboni, and G. Turvani, "Engineering Grover adaptive search: Exploring the degrees of freedom for efficient QUBO solving," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 3, pp. 614–623, Sep. 2022, doi: 10.1109/JETCAS.2022.3202566.

[30] W. P. Baritompa, D. W. Bulger, and G. R. Wood, "Grover's quantum algorithm applied to global optimization," *SIAM J. Optim.*, vol. 15, no. 4, pp. 1170–1184, Jan. 2005, doi: 10.1137/040605072.

[31] T. Li, S. Zhang, X.-Q. Fu, X. Wang, Y. Wang, J. Lin, and W.-S. Bao, "Quantum search for unknown number of target items by hybridizing fixed-point method with trail-and-error method," *Chin. Phys. B*, vol. 28, no. 12, Nov. 2019, Art. no. 120301, doi: 10.1088/1674-1056/ab4e88.

[32] A. Gilliam, S. Woerner, and C. Gonciulea, "Grover adaptive search for constrained polynomial binary optimization," *Quantum*, vol. 5, p. 428, Apr. 2021, doi: 10.22331/q-2021-04-08-428.

[33] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, and D. Bucher, "Qiskit: An open-source framework for quantum computing," Zenodo, 2019, doi: 10.5281/zenodo.2562110.

[34] M. Norimoto, R. Mori, and N. Ishikawa, "Quantum algorithm for higher-order unconstrained binary optimization and MIMO maximum likelihood detection," *IEEE Trans. Commun.*, vol. 71, no. 4, pp. 1926–1939, Apr. 2023, doi: 10.1109/TCOMM.2023.3244924.

[35] Y. Sano, M. Norimoto, and N. Ishikawa, "Qubit reduction and quantum speedup for wireless channel assignment problem," *IEEE Trans. Quantum Eng.*, vol. 4, pp. 1–12, 2023, doi: 10.1109/tqe.2023.3293452.

[36] I. M. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," *Rev. Modern Phys.*, vol. 86, no. 1, pp. 153–185, Mar. 2014, doi: 10.1103/revmodphys.86.153.

[37] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, "The variational quantum eigensolver: A review of methods and best practices," *Phys. Rep.*, vol. 986, pp. 1–128, Nov. 2022, doi: 10.1016/j.physrep.2022.08.003.

[38] Y. Zhou, E. M. Stoudenmire, and X. Waintal, "What limits the simulation of quantum computers?" *Phys. Rev. X*, vol. 10, no. 4, Nov. 2020, Art. no. 041038, doi: 10.1103/physrevx.10.041038.

[39] P. Seitz, I. Medina, E. Cruz, Q. Huang, and C. B. Mendl, "Simulating quantum circuits using tree tensor networks," *Quantum*, vol. 7, p. 964, Mar. 2023, doi: 10.22331/q-2023-03-30-964.

[40] Á. Nagy, J. Park, C. Zhang, A. Acharya, and A. Khan, "Fixed-point Grover adaptive search for QUBO problems," 2023, *arXiv:2311.05592*.

[41] Y. Huang and S. Pang, "Optimization of a probabilistic quantum search algorithm with a priori information," *Phys. Rev. A, Gen. Phys.*, vol. 108, no. 2, Aug. 2023, Art. no. 022417, doi: 10.1103/physreva.108.022417.

HIROAKI OMINATO received the B.Sc. and M.Sc. degrees in physics from Kyoto University, in 2020 and 2022, respectively. He has been involved in quantum computing research with Panasonic System Networks Research and Development Laboratory Company Ltd., since 2022. He is engaged in research on quantum computing methods for solving optimization problems in logistics and wireless communications.

TAKAHIRO OHYAMA (Graduate Student Member, IEEE) received the B.E. degree in electronic engineering and the M.E. degree in electrical and electronic systems engineering from Nagaoka University of Technology, Niigata, Japan, in 2001 and 2003, respectively. He is currently pursuing the Ph.D. degree with the Graduate School of Information Sciences (GSIS), Tohoku University, Sendai, Japan. Since his association with Panasonic System Networks Research and Development Laboratory Company Ltd., from 2003, he has been engaged in research on mobile communications and quantum computing. He is also a member of the Institute of Electronics, Information, and Communication Engineers (IEICE).

KOICHIRO YAMAGUCHI received the B.E. and M.E. degrees in computer science from Tokyo Institute of Technology, Japan, in 1996 and 1998, respectively. He began working with the Research and Development Division, Panasonic Company, in 1998. He is currently researching the optimization of logistics and quantum computing with the Research and Development Division, Panasonic Connect Company. He was engaged in developing a digital broadcasting system.