entropy

IMPACT FACTOR 2.1

Indexed in: PubMed

CITESCORE 4.9

Article

An Improved GAS Algorithm

Zhijian Wang, Yuchen He, Tian Luan and Yong Long

Special Issue

Quantum Computing in the NISQ Era

Edited by

Dr. Xiao Yuan, Dr. Xiaoming Zhang and Dr. Bálint Koczor

MDPI

https://doi.org/10.3390/e27030240

entropy

MDPI

Article

An Improved GAS Algorithm

Zhijian Wang [1,†], Yuchen He [2,†], Tian Luan [1,2,*] and Yong Long [1]

[1] Department of Computer Technology, Institute of Advanced Technology, University of Science and Technology of China, No. 96 Jinzhai Road, Hefei 230088, China; wangzhijian@mail.ustc.edu.cn (Z.W.); longyong@mail.ustc.edu.cn (Y.L.)
[2] Yangtze Delta Region Industrial Innovation Center of Quantum and Information Technology, Suzhou 215100, China; heyuchen@tgqs.net
* Correspondence: luantian@tgqs.net; Tel.: +86-13681052686
† These authors contributed equally to this work.

Abstract: This paper introduces an improved Grover Adaptive Search (GAS) algorithm. The GAS algorithm has been prove to achieve quadratic acceleration in the Constrained Polynomial Binary Optimization (CPBO) problem. Nevertheless, the acceleration effect of the GAS algorithm can be decreased by the poor threshold selection. This article uses the Quantum Approximate Optimization Algorithm (QAOA) to improve the initial threshold selection, thereby accelerating the convergence speed of the original GAS algorithm. The acceleration effect of the improved GAS algorithm is presented by the Max-Cut problem and the CPBO problem.

Keywords: GAS; QAOA; quantum computing

Academic Editors: Xiao Yuan, Xiaoming Zhang and Bálint Koczor

Received: 22 December 2024
Revised: 12 February 2025
Accepted: 18 February 2025
Published: 26 February 2025

Citation: Wang, Z.; He, Y.; Luan, T.; Long, Y. An Improved GAS Algorithm. *Entropy* **2025**, *27*, 240. https://doi.org/10.3390/e27030240

1. Introduction

Solving the large-scale quadratic unconstrained binary optimization (QUBO) [1,2] problem and CPBO problem with classical heuristic algorithms presents many challenges and disadvantages. In fact, the QUBO and CPBO problem is NP-hard [3,4], which means that as the size of the problem increases, exhaustively searching for all possible solutions to find the optimal one becomes infeasible due to the exponential growth of the complexity of time. Moreover, the classical heuristic algorithms (such as simulated annealing, genetic algorithms, and tabu search) are prone to becoming stuck in local optima. Consequently, these algorithms may not identify the optimal global solution for large or complex QUBO problems, frequently resulting in suboptimal solutions. Additionally, the objective function of the QUBO problem may consist of complex interaction terms, further increasing the difficulty of solving the problem. Although some classical algorithms can be parallelized to enhance performance, the efficiency of parallelization is frequently constrained by the intrinsic nature of the algorithm. In certain instances, parallelization may result in additional overhead and may not signally enhance the efficiency of the solution.

Compared with the classical algorithm, quantum algorithms show potential in solving the large-scale QUBO problem. The QAOA [5–8] leverages the properties of quantum mechanics to explore the solution space more efficiently. One potential benefit of quantum mechanics is that it can process multiple pieces of information at the same time. This ability can greatly improve the speed and efficiency of computation when dealing with large-scale problems. The GAS [9,10] algorithm is a quantum search algorithm based on the Grover [11,12] algorithm. It can be used to solve the CPBO [13] problem. The GAS algorithm utilizes an artificial threshold as the initial benchmark to identify all values that surpass it in the quest for a better solution. However, if the threshold is not appropriately

chosen, the algorithm's performance can degrade significantly, resulting in the loss of its acceleration advantage.

This paper introduces an improved GAS algorithm that uses the QAOA to select the initial threshold to avoid the problem of the algorithm's performance degradation mentioned above when dealing with CPBO problems. We evaluate our algorithm both on a simplified credit card scoring problem (CPBO problem) and the Max-Cut [14] problem (QUBO problem). Comparing the experimental results, the average time complexity of the experiments with the improved GAS algorithm is 56.74% lower than the GAS method in the credit card scoring problem.

The remainder of this paper is organized as follows. In the Results section, we define the credit card scoring problem, describe the details of the improved GAS algorithm, and apply the proposed algorithm to both the credit card scoring problem and the Max-Cut problem. Finally, we provide a conclusion.

2. Define CPBO Problem

The credit card scoring problem is very important in the modern financial system. Usually, after designing an evaluation model, this problem can be converted to a CPBO problem to find the best credit cart portfolio. The Table 1 explains the meaning of the variable and objective function is given as

$$F(x_1, x_2, ..., x_n) = \sum_{0<i\leq n} crt_i(1-h_i)x_i - ct_ih_ix_i, \qquad x_i \in \{0,1\}, \quad n \in N, \tag{1}$$

where x_i represents whether the credit card is picked, n is the total number of the credit cards, and $h_i \in [0,1]$, $c_i \in R$, $t_i \in [0,1]$ and $r \in R$ represent the bad debt rate, the total loan, the loan approval rate, and the interest rate for card i, respectively.

Table 1. Notations.

Notation	Description
$F(x_1, x_2, \ldots, x_n)$	Payoff function
n	The number of credit card
c	Total loan of each card
r	Credit card interest rate
t_i	Card i loan approval rate
h_i	Bad debt rate for card i

In practice, several constraints have to be considered in the credit card scoring problem. Here, we consider a simple constraint $\sum_{0<i\leq n} x_i = B$.

The CPBO problem is defined as

$$\underset{\substack{x_i\in\{0,1\}\\ i\in\{1,2,\ldots,n\}}}{\arg\max}\ F(x_1, x_2, \ldots, x_n), \quad s.t. \sum_{0<i\leq n} x_i = B.$$

Penalty Factors and Initial States

To incorporate the aforementioned constraint into the payoff equation, two different approaches have been considered for constructing the constraint in the QAOA module and the GAS module. Specifically, for the GAS module, the initial state and objective are defined as follows:

$$|\psi_0\rangle_G = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle_n,$$

$$F^{(0)}(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n) - P(\sum_{0<i\leq n} x_i - B)^2,$$

where the budget constraint is added as a penalty term to F, with P being the penalty coefficient.

The P should be chosen to be large enough s.t. $F^{(f)} > F^{(0)}_{max} > F^{(v)}$ for all infeasible states, yielding

$$F^{(v)}(x_1, x_2, \ldots, x_n) = \max F(x_1, x_2, \ldots, x_n),$$

$$F^{(f)}(x_1, x_2, \ldots, x_n) = \max_{\sum_{0<i\leq n} x_i = B} F(x_1, x_2, \ldots, x_n).$$

In the credit card scoring problem, we choose $P \leq \frac{2^{m-1}}{(\max\{n-B,B\})^2}$, s.t. the performance of GAS is not expected to deteriorate with the P. The m is the number of qubits available to store the result.

For the QAOA module, we set the Dicked state which satisfies the budget constraints as the initial state [8].

$$|\psi_0\rangle_M = \frac{1}{\sqrt{\binom{n}{B}}} \sum_{\substack{i_1,i_2,\ldots,i_n\in\{0,1\} \\ i_1+i_2+\cdots+i_n=B}} |i_1 i_2 \ldots i_n\rangle.$$

3. The Improved GAS Algorithm

The improved GAS algorithm combines the advantages of the QAOA and GAS algorithm to solve the CPBO problem. The overall flow of the algorithm is shown in Figure 1, our algorithm improves the original GAS with a better initial threshold selection.

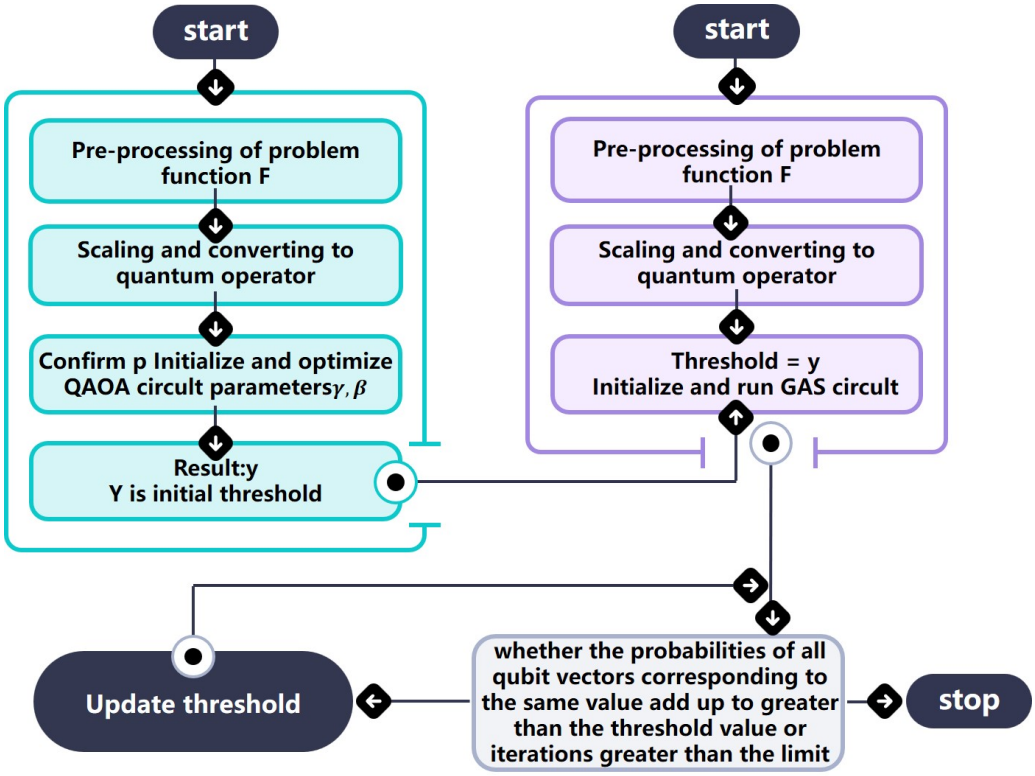


Figure 1. The chart shows all the steps of the improved GAS algorithm; the left part is the QAOA module which is used to calculate the initial threshold y. The right part is the GAS module which is used to receive and update y and calculate the final result.

3.1. The QAOA Module

QAOA can leverage quantum superposition to represent multiple solution states simultaneously in a single computational step, thereby improving search efficiency. Moreover, classical algorithms often rely on heuristic searching or gradient descent, making them prone to getting stuck in local optima. For NP-hard problems, solving them on classical computers requires exponential time, while QAOA theoretically offers the potential for exponential speedup. Finally, since this case involves a CPBO problem and we only need to obtain an approximate solution, QAOA not only theoretically demonstrates the possibility of polynomial speedup but also inherently possesses an approximate adiabatic evolution property, which makes it well suited to finding approximate solutions. Therefore, in our study, we used QAOA to construct the threshold value. To use the QAOA, we convert $F^{(f)}$ into a quantum operator.

$$\hat{F}_{QAOA} = F(\frac{(\hat{I}_1 - \hat{Z}_1)}{2}, \frac{(\hat{I}_2 - \hat{Z}_2)}{2}, \ldots, \frac{(\hat{I}_n - \hat{Z}_n)}{2}) = \sum_{0<i\leq n} crt_i(1-h_i)\frac{(\hat{I}_i - \hat{Z}_i)}{2} - ct_i h_i \frac{(\hat{I}_i - \hat{Z}_i)}{2}, \tag{2}$$

where $\hat{Z}_i$ denotes the *Pauli-*$\hat{Z}$ gate, acting on the i-th qubit. Due to the XY full mixer being able to ensure that the number of ones in the state remains unchanged, thus satisfying the constraint conditions, we use the XY full mixer to search for the optimal solution in all feasible states to explore a broader solution space and increase the probability of finding the global optimum. The XY full mixer is given as

$$\hat{U}_M(\beta) = \prod_{(i,j)\in S_M} \hat{R}_{i,j}^{(XY)}(\beta),$$

and the $\hat{R}_{i,j}^{XY}$ are arranged into n subsets of $\frac{n-1}{2}$ commuting operations where $i + j \; mod \; n = k$ in subset k. (For example, for n = 3, S = {(1, 3), (2, 3), (1, 2)} with the subsets {(1, 3)}, {(2, 3)}, {(1, 2)}.) If the n is even, we first generate the subsets for the $n-1$. The first step is to generate the subsets as described above. Then, for each subset, we add the missing pair of qubits. (For example, for the $n = 4$, we add (2, 4) to the first subset {(1, 3)}, (1, 4) to the second, and so on. The resulting set is {(1, 3), (2, 4), (2, 3), (1, 4), (1, 2), (3, 4)}), where M represents the XY full mixer model. Under this model, the set S contains all pairs of qubits, where the order is chosen such that as many gates as possible can be performed in parallel, thus minimizing the depth of the circuit. If n is odd,

$$\hat{R}_{i,j}^{(XY)}(\beta) = e^{i\beta(\hat{X}_i\hat{X}_j + \hat{Y}_i\hat{Y}_j)}.$$

The QAOA algorithm uses adiabatic evolution to get the final result, with the following quantum state depending on the parameters $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$ and $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$, with p (p is the depth of the QAOA quantum circuit) being the number of iterations,

$$|\psi_p(\vec{\gamma}, \vec{\beta}) >_M = \hat{U}_M(\beta_p)e^{-i\gamma_p\hat{F}} \ldots \hat{U}_M(\beta_2)e^{-i\gamma_2\hat{F}}\hat{U}_M(\beta_1)e^{-i\gamma_1\hat{F}}|\psi_0\rangle_M.$$

Finally, all qubits are measured with respect to the standard basis in order to determine the mean value,

$$\langle F\rangle_{\vec{\gamma},\vec{\beta}} = \langle\psi_{\vec{\gamma},\vec{\beta}}|\hat{F}|\psi_{\vec{\gamma},\vec{\beta}}\rangle.$$

The mean value is then passed to a classical optimizer, which gives new values to the parameters γ and β to minimize the expectation $\langle F\rangle_{\vec{\gamma},\vec{\beta}}$.

The QAOA circult can be shown as Figure 2.

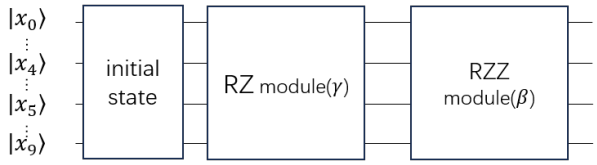


Figure 2. This circuit implies the equation $\hat{U}_{S_M}(\beta)e^{-i\gamma\hat{F}}=\prod_{(i,j)\in S_M}\hat{R}_{i,j}^{XY}(\beta)\prod_{(i,j)\in S_M}e^{-i\gamma W_{ij}\hat{Z}_i\hat{Z}_j}\prod_{i=1}^{n}e^{-i\gamma w_i\hat{Z}_i}$. The RZ module represents $\prod_{i=1}^{n}e^{-i\gamma w_i\hat{Z}_i}$. The RZZ module represents $\prod_{(i,j)\in S_M}e^{-i\gamma W_{ij}\hat{Z}_i\hat{Z}_j}$. W_{ij} is the coefficient of the $\hat{Z}_i\hat{Z}_j$ in $\hat{F}_{QAOA}$; the w_i is the coefficient of the $\hat{Z}_i$ in $\hat{F}_{QAOA}$.

3.2. GAS Module

The GAS algorithm has three components:

1. The A_y prepares an n-qubits input register to represent the equal superposition of all $|x\rangle_n$ and m-qubits output register to (approximately) represent the corresponding objective function $|f(x)-y\rangle_m$.

$$A_y|0\rangle_n|0\rangle_m=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle_n|f(x)-y\rangle_m,$$

   where $|x\rangle$ is the binary encoding of the integer x.
2. The oracle O recognizes the states of interest and multiplies their amplitudes by -1.

$$O|x\rangle_n|z\rangle_m=sign(z)|x\rangle_n|z\rangle_m.$$

3. The Grover diffusion operator D has the effect of flipping all amplitudes in the quantum state according to their mean. This causes all the amplitudes of the states of interest to be magnified, while the amplitudes of all other states are decreased. For more details on building the GAS algorithm, see the Appendix A.

$$D=H^{\otimes n+m+1}(2|0\rangle\langle 0|-I)H^{\otimes n+m+1}.$$

Furthermore, applying the Grover operator $A_yDA_y^{\dagger}O$ r times to state $A_y|0\rangle_n$ for an integer $r\geq 0$ will maximally amplify the amplitudes of the states of interest.

To ensure the probability of sampling the target state of at least $1/2$, the optimal number of r depends on the number of all states $M=2^n$ and the number of target states N, with $r=\lfloor\frac{\pi}{4}\sqrt{\frac{M}{N}}\rfloor$.

This is a quadratic speed-up with respect to the classical search. Since N is in general unknown, in this article, we take a randomised strategy to set N.

When we use the GAS algorithm to solve credit card scoring problems, it is necessary to convert the payoff equation into the matrix form.

$$\begin{aligned}\hat{F}_{GAS}&=\sum_{0<i\leq n}crt_i(1-h_i)x_i-ct_ih_ix_i-P(\sum_{0<i\leq n}x_i-B)^2\\&=\sum_{i,j=1}^{n}Q_{ij}x_ix_j+\sum_{i=1}^{n}b_ix_i+c.\qquad x_i=0,1,\end{aligned}\tag{3}$$

where $Q=\begin{bmatrix}-P & 0 & \cdots & 0\\ -2P & -P & \cdots & 0\\ \vdots & \cdots & \ddots & \vdots\\ -2P & -2P & \cdots & -P\end{bmatrix}$,

$b = \left[2PB + crt_1(1-h_1) - ct_1h_1 \quad \cdots \quad 2PB + crt_n(1-h_n) - ct_nh_n\right]$.
The improved GAS algorithm (Algorithm 1) pseudocode is given below:

Algorithm 1: Improved-GAS algorithm

Input: $f : k \leftarrow \mathbb{R}^+$, $\Theta_0 = [0 \ldots n]$, $\Theta_0[j]_{j\in[0,n)} \in (0,1)$, $\lambda > 1$;
Output: y

set $shot\_QAOA = 512$;
set $P = F_v$;
set $i = 1$;
set $shot\_GAS = 200$ and $max\_iter = 60$;
set $threshold = 0.7$;
repeat
    Run QAOA circuit;
    Measure result;
    Update Θ_0;
until The difference between the updated Θ_0 of the optimizer and the last Θ_0 is less than a certain precision;

Run QAOA circuit;
$y = y_1 =$ Measure result;

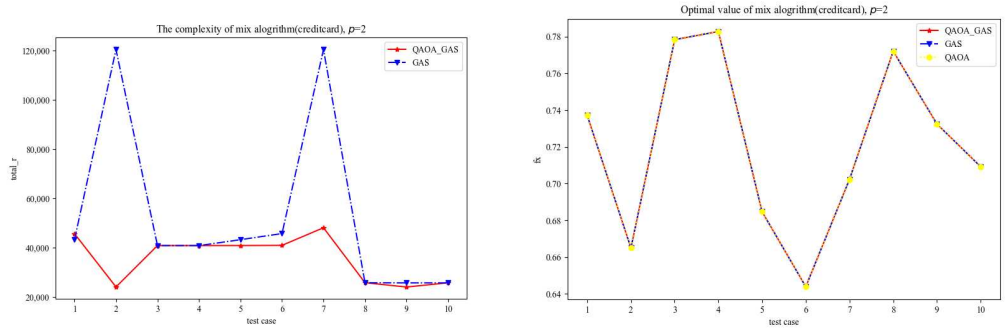
repeat
    Select the rotation count r_i from the set$\{\lfloor k/2 \rfloor, \ldots, \lceil k-1 \rceil\}$ randomly;
    Run the GAS quantum circuit and apply Grover Search with r_i iterations using oracles A_{y_i} and O_{y_i}. We denote the outputs y respectively;
    if $y < y_i$ **then**
        y=y_i;
        ans = y_i;
    else
        $ans = y$;
        $i = i + 1$;
    end if
until The $i = max\_iter$ or y correspond to all of the frequencies greater than the threshold;

4. Test Case

In this section, we present an experiment. Currently, the credit card scoring problem we considered includes 1000 credit card data, and the existing machines are not enough to run that amount of data at the same time. So we have to divide the credit card data into 200 groups and test with 10 cards' data each time. All of the following experiments are performed with the initial $y = 10$ in the GAS module and $p = 2$ in the QAOA module.

Figure 3a,b represent the result of the credit card scoring problem. In Figure 3a, the blue line represents the time complexity of the GAS algorithm, which is calculated by the actual number of iterations multiplied by the quantum circuit depth, and the red line represents the time complexity of the improved GAS algorithm, which is obtained by adding the complexity of the GAS algorithm to the complexity of the QAOA algorithm; the time complexity of the QAOA is obtained by multiplying the number of layers p by the maximum number of updates of Θ. The Y-label represents the total complexity, and the X-label represents the different test cases. According to Figure 3a, we can conclude that the time complexity of the improved GAS algorithm is better than that of the GAS algorithm. In Figure 3a, there are three different-colored lines, each line represent the improved GAS algorithm, the GAS algorithm and the QAOA. The Y-label represents the value of the algorithm, the X-label still represents the different test cases. It can be seen

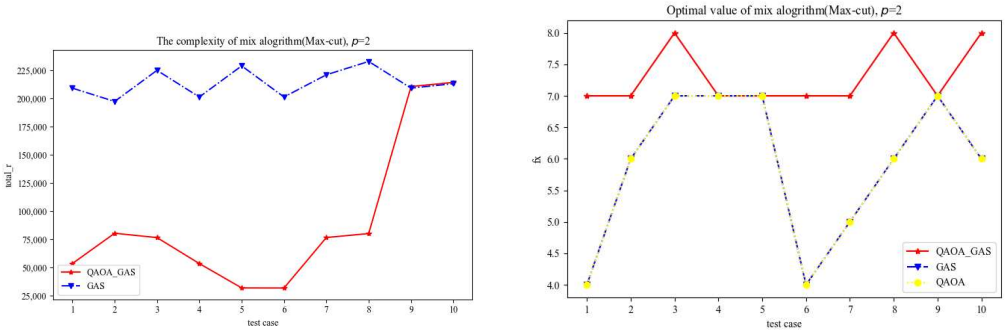
from the figure that the accuracy of the improved GAS algorithm is still consistent with the other two algorithms.



(**a**) (**b**)

Figure 3. (**a**) is speed chart of the credit card scoring problem; (**b**) is accuracy chart of credit card scoring problem.

Next, we test our algorithm on a Max-Cut problem with 20 nodes and 12 edges, as shown in Figure 4a,b. Obviously, in the QUBO problem, the improved GAS algorithm performs better than the GAS algorithm in time complexity. Similarly, the improved GAS algorithm has advantages over the QAOA algorithm in terms of accuracy. Note that the accuracy of the improved GAS algorithm should theoretically be the same as that of the GAS algorithm. However, in this experiment, we set the max-iteration which causes both the GAS algorithm and the improved GAS algorithm to stop before turning over the correct result. For this reason, the GAS algorithm or the enhanced GAS algorithm may not achieve the desired accuracy. Overall, the experiment achieved the expected results. The algorithm used a shallow QAOA algorithm to find the solution, which was faster than both the deep QAOA algorithm and non-quantum algorithms. Compared with the GAS algorithm, its time consumption was negligible.



(**a**) (**b**)

Figure 4. (**a**) is speed chart of the Max-Cut; (**b**) is accuracy chart of the Max-Cut.

Moreover, unlike the GAS algorithm, where the threshold value is either randomly chosen or manually set, the new algorithm avoids this issue. The GAS algorithm determines whether a condition is met by checking the sign bit. If the initial threshold is lower than the solution value, the iteration will continue indefinitely unless a maximum iteration count is set. Conversely, if the threshold is higher than the solution value but the gap is too large, the iterative update logic may lead to excessive iterations.

As previously discussed, the time required for a single iteration of the entire algorithm model is approximately $\lfloor \frac{\pi}{4}\sqrt{\frac{M}{N}} \rfloor$. The formula indicates that if a good initial value is provided, the time complexity increases as the number of iterations grows. This explains why the execution time of the new algorithm has been significantly reduced.

5. Conclusions

In this paper, we introduce an improved GAS algorithm which uses QAOA to generate the initial threshold y applied to the GAS algorithm to solve the CPBO problem. Our approach improves the speed of the GAS algorithm without affecting the accuracy. In this study, we demonstrate the efficacy of our algorithm in two distinct domains: the credit card scoring problem and Max-Cut problem. In this manuscript, we focus on integers and two decimal places since, as the number of decimal places increases, the number of qubits required must also increase, potentially exceeding the computing power of existing machines. Since this algorithm can be adapted to credit card problems, and credit card data typically consist of both an integer part and a decimal part (rounded to two decimal places), the model can be fully applied to search problems with similar data characteristics. Examples include stock selection problems, real bounding box filtering in object detection, and more.

Furthermore, by improving the quantum circuit construction of the algorithm and leveraging advancements in quantum hardware, it will be possible to process data with more decimal places. In the NISQ era, this search algorithm provides a more efficient approach to handling CPBO problems and offers a new solution for data cleaning and classification in the AI era.

This algorithm can serve as a valuable complement to existing methods and play a significant role in CPBO problems.

Author Contributions: Conceptualization, Z.W. and Y.H.; methodology, T.L., Y.H. and Z.W.; software, Z.W. and Y.H.; validation, Z.W., Y.H. and Y.L.; formal analysis, Z.W.; investigation, Z.W.; resources, T.L. and Y.H.; data curation, Z.W.; writing—original draft preparation, Z.W.; writing—review and editing, Z.W. and Y.H.; visualization, Z.W.; supervision, Y.H. and T.L.; project administration, Z.W.; funding acquisition, T.L. and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: 1. Special Funding for Basic Research in Jiangsu Province (Soft Science Research) (BK20243046). 2. Science and Technology Innovation Support Platform Project (SZS2022416).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of 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.

Appendix A

Appendix A.1

The main idea of the GAS algorithm is to construct Ay and Oy for a given threshold y such that the y flag all states x$\in$ {0,1} satisfying $f(x) < y$, such that we can use the Grover algorithm to find a solution $\hat{x}$ that makes $f(\hat{x})$ smaller than y. Then, we set the $y = f(\hat{x})$ and repeat the process until some formal termination criteria are met.

Appendix A.2. Construct Operator A

In this section, we show how to construct the operator O. Due to the m-register representing the corresponding $|\hat{F}_{GAS} - y\rangle_m$, we have to use a relatively simple structure to achieve this operator. The simplest implementation is to use the phase gate $R(\theta)$: when the gate is applied to one qubit, it rotates the phase of the amplitudes of the states, having 1 in the position corresponding to that qubit.

Consider that if there is a kx_ix_j term that exists in $\hat{F}_{GAS}$, when we use matrix restore k, the k can be expressed as a_{ij}. If we apply $UG(\frac{2\pi}{2^m}k)$ followed by the inverse Quantum Fourier Transform (QFT) to an m-qubit register, we end up with $k(mod2^m)$ being encoded in the register.

Note that credit card scoring problems have a degree of monomials less than or equal to 2, so we only need to control single qubits and pairs of qubits (i.e there only exists a_i or $a_{ij}, i, j \in [0, n)$ in $\hat{F}_{GAS}$).

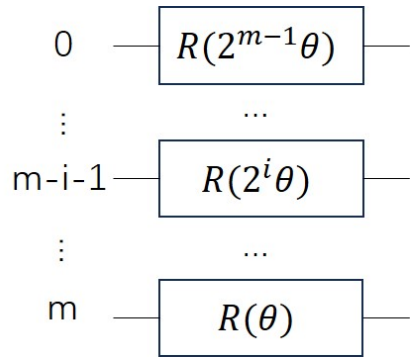


Figure A1. A circuit for unitary operator $U_G(\theta)$, where $\theta \in [-\pi, \pi)$ is applied to an m-qubit register. The symbol R denotes the phase gate, which rotates the amplitudes of states containing 1 in the position corresponding to the qubit to which it is applied.

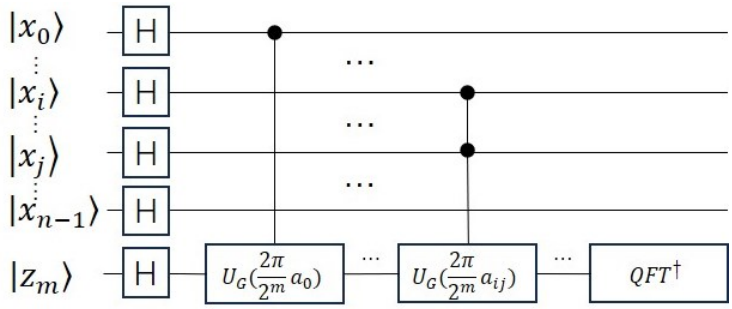


Figure A2. This circuit represents operator A, each U_G operator represents the non-zero term in $\hat{F}_{GAS}$, and the main function of the circuit is the encode of an integer $k \in [-2^{m-1,2^{m-1}})$, which can apply to a register of m-qubits in an equal superposition.

Appendix A.3. Construct Operator O

At each stage of the algorithm, a fixed coefficient is incorporated into the polynomial, and the remaining negative values are identified. This implies that the oracle is only required to recognize negative integers. Since values are represented in complement-on-two, where the most significant (left-most) bit designates the sign of the number, a single quantum bit in the value register can be employed to identify negative integers. The conventional oracle that multiplies target amplitudes by -1 may be utilized. It should be noted that the oracle remains unchanged throughout the iterations, as the addition of a constant to the polynomial may result in overflow within the value register. Consequently, in order to circumvent this issue, it may be necessary to increase the number of qubits in the value register by one.

Appendix A.4. Construct Operator D and Update y

The operator D is the last stage of algorithm. When we combine all the operations together, we get the $\hat{F}_{GAS} + y$ from the quantum circuit. The circuit of Figure A3 is executed in order to achieve the desired counts. In the event that the counts are divided by the

total number of samples, and the result is less than the set threshold, the circuit must be repeated until the algorithm reaches the maximum number of iterations, or the threshold is exceeded. It is possible for it to be observable that multiple maxima have the same value. In order to address this issue, a pruning operation has been implemented which involves determining whether the same frequency corresponds to the same value and then selecting one of them then terminate the algorithm.

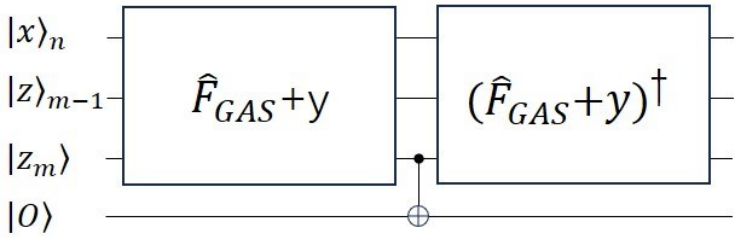


Figure A3. In this circuit (the $\hat{F}_{GAS} + y$ module contains operator D), there are n binary variables in the $\hat{F}_{GAS} + y$, and the global indicator qubit (shown at the bottom of the circuit) is set to $|1\rangle$ if and only if $\hat{F}_{GAS} + y$—where y is the threshold parameter from the GAS algorithm that is less than 0.

References

1. Dunning, I.; Gupta, S.; Silberholz, J. What works best when? A systematic evaluation of heuristics for Max-Cut and QUBO. *INFORMS J. Comput.* **2018**, *30*, 608–624. [CrossRef]
2. Dury, B.; Di Matteo, O. A QUBO formulation for qubit allocation. *arXiv* **2020**, arXiv:2009.00140.
3. Arora, S. The approximability of NP-hard problems. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, TX, USA, 24–26 May 1998; pp. 337–348.
4. Woeginger, G.J. Exact algorithms for NP-hard problems: A survey. In Proceedings of the Combinatorial Optimization—Eureka, You Shrink! Jack Edmonds 5th International Workshop, Aussois, France, 5–9 March 2001; Revised Papers; Springer: Berlin/Heidelberg, Germany, 2003; pp. 185–207.
5. Bärtschi, A.; Eidenbenz, S. Grover mixers for QAOA: Shifting complexity from mixer design to state preparation. In Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), Denver, CO, USA, 12–16 October 2020; pp. 72–82.
6. Choi, J.; Kim, J. A tutorial on quantum approximate optimization algorithm (QAOA): Fundamentals and applications. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 16–18 October 2019; pp. 138–142.
7. Guerreschi, G.G.; Matsuura, A.Y. QAOA for Max-Cut requires hundreds of qubits for quantum speed-up. *Sci. Rep.* **2019**, *9*, 6903. [CrossRef] [PubMed]
8. Brandhofer, S.; Braun, D.; Dehn, V.; Hellstern, G.; Hüls, M.; Ji, Y.; Polian, I.; Bhatia, A.S.; Wellens, T. Benchmarking the performance of portfolio optimization with QAOA. *Quantum Inf. Process.* **2022**, *22*, 25. [CrossRef]
9. Gilliam, A.; Woerner, S.; Gonciulea, C. Grover adaptive search for constrained polynomial binary optimization. *Quantum* **2021**, *5*, 428. [CrossRef]
10. Federer, M.; Lenk, S.; Müssig, D.; Wappler, M.; Lässig, J. Constrained Grover Adaptive Search for Optimization of the Bidirectional EV Charging Problem. In Proceedings of the INFORMATIK 2023, Berlin, Germany, 26–28 September 2023.
11. Long, G.L. Grover algorithm with zero theoretical failure rate. *Phys. Rev. A* **2001**, *64*, 022307. [CrossRef]
12. Jozsa, R. Searching in Grover's algorithm. *arXiv* **1999**, arXiv:quant-ph/9901021.
13. Elloumi, S.; Verchère, Z. Efficient linear reformulations for binary polynomial optimization problems. *Comput. Oper. Res.* **2023**, *155*, 106240. [CrossRef]
14. Kapralov, M.; Krachun, D. An optimal space lower bound for approximating MAX-CUT. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, Phoenix, AZ, USA, 23–26 June 2019; pp. 277–288.