Postprocessing Optimisation Algorithms for QAOA in MaxCut problem

Emanouil Atanassov[1,2][0000−0002−7442−7096], Aleksandar Kirilov[1,2][0009−0004−3338−5813], Mariya Durchova[1,2][0000−0003−2548−282X], and Ivaylo Georgiev[1]

[1] Institute of Information and Communication Technologies - Bulgarian Academy of Sciences (IICT-BAS), Sofia, Bulgaria
[2] Centre of Excellence in Informatics and Information and Communication Technologies, Sofia, Bulgaria
emanouil@parallel.bas.bg, alex.kirilov@iict.bas.bg, mabs@parallel.bas.bg, igeorgiev@parallel.bas.bg
https://iict.bas.bg

Abstract. The well known Quantum Approximate Optimization Algorithm (QAOA) has been shown to be useful in finding solutions to the MaxCut problem in graph theory. However, the results when using quantum hardware are significantly affected by noise. In addition we have the natural sampling errors, coming from using limited number of shots. Substantial effort has been dedicated to development of error mitigation techniques. In this work we concentrate on goal-oriented postprocessing algorithms, that directly target improvement of the original combinatorial objective. We compare a family of widely used classical postprocessing algorithms, when applied to the QAOA measurement outcomes for the MaxCut problem. Due to our limited access to quantum hardware, we use statevector simulation in order to determine good values of the parameters of the quantum circuit and submit only one circuit for each problem instance on quantum hardware. Thus we are able to compare the performance of the postprocessing optimisation algorithms on results from both software simulation and hardware execution. For some of the optimisation algorithms we introduce versions that utilize low-discrepancy sequences with the goal to better explore the underlying solution spaces. Our results demonstrate that objective-based postprocessing can substantially improve solution quality even when the raw QAOA output has far from optimal objective value.

Keywords: MaxCut problem· QAOA · low-discrepancy sequences

1 Introduction

The Quantum Approximate Optimization Algorithm (QAOA), introduced in [11], is one of the most popular hybrid quantum–classical frameworks for solving combinatorial optimization problems on quantum hardware, with good performance on the currently available Noisy Intermediate-Scale Quantum (NISQ)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

devices. QAOA combines multiple layers of problem-specific and mixing unitary transformations, so that the measured bitstrings encode candidate solutions to discrete optimization tasks. Among the widely studied benchmark problems, MaxCut is relevant not only in pure graph theory, but also with applications in statistical physics, VLSI circuit design, data clustering and community detection. In classical MaxCut formulation, one tries to partition graph vertices into two subsets such that the number (or weight) of edges crossing the partition is maximized. The problem admits a natural encoding into an Ising-type Hamiltonian, H_C. QAOA prepares a quantum state through an alternating sequence of unitary operators parameterized by angles $\{\gamma_k, \beta_k\}_{k=1}^{p}$, where p denotes the number of layers or depth of the circuit. These parameters can be physically interpreted as evolution times; γ_k represents the duration the system evolves under the cost Hamiltonian H_C, while β_k can be interpreted as the time spent under the mixing Hamiltonian H_B. The final quantum state depends nonlinearly on these parameters and induces a probability distribution over bitstrings when measurement is performed. The parameters $\{\gamma_k, \beta_k\}$ are optimized classically in order to maximize the expected value of the cost Hamiltonian.

This optimization is itself a nontrivial noisy optimisation problem, however it is not the focus of this work. In fact, we performed this optimisation using statevector simulation, so we assume that good parameters have been obtained, and we instead concentrate on what can be achieved after measurement samples are obtained, with these parameters fixed. The selection of the number of layers p is also an open problem, as larger circuits may result in worse noise, but in this work we simply tested different values of p.

Apart from the hardware noise, we have statistical noise due to the limited number of shots. Thus we are not sure if we are able to detect the most probable bitstring, even in absence of noise. Our main goal given the noisy distribution of candidate bitstrings obtained from measurement, is to evaluate classical optimisation algorithms that increase the probability of finding high-quality cuts. In the next section we describe in more detail our methodology for producing the measurement outputs, which in turn serve as inputs to our optimisation algorithms. Then we describe the various optimisation algorithms and their quasi-Monte Carlo variants [4,2]. Numerical results obtained by postprocessing results from both statevector simulation and hardware execution are presented and discussed.

2 Methodology

2.1 QAOA for MaxCut

For a graph $G = (V, E)$, the MaxCut problem [7,14] can be encoded using binary variables $z_i \in \{-1, 1\}$ representing the respective vertex assignments. Consequently the objective function can be encoding in operator form as a diagonal Hamiltonian expressed in terms of Pauli-Z operators acting on the qubits associated with the vertices:

$$H_C = \frac{1}{2} \sum_{(i,j)\in E} (1 - Z_i Z_j)\,,$$

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

where Z_i are Pauli-Z operators acting on the qubit associated with vertex i. This Hamiltonian assigns lower energy to bitstrings that cut more edges. The mixing Hamiltonian is usually chosen as a transverse-field operator,

$$H_M = \sum_{i \in V} X_i,$$

where X_i is the Pauli-X operator acting on the respective qubit i. This idea of the mixing Hamiltonian is to ensure better exploration of the solution space.

The QAOA state of depth p is generated by alternating exponentials of the cost Hamiltonian H_C and a mixing Hamiltonian H_M. The resulting state is then measured in the computational basis to obtain bitstrings representing candidate cuts. Thus the full QAOA state is given by

$$|\psi_p(\gamma, \beta)\rangle = \prod_{\mathbf{k=1}}^{\mathbf{p}} \mathbf{e}^{-\mathbf{i}\beta_\mathbf{k}\mathbf{H_M}}\, \mathbf{e}^{-\mathbf{i}\gamma_\mathbf{k}\mathbf{H_C}}\, |+\rangle^{\otimes \mathbf{n}},$$

where $\gamma = (\gamma_\mathbf{1}, \ldots, \gamma_\mathbf{p})$ and $\beta = (\beta_\mathbf{1}, \ldots, \beta_\mathbf{p})$ are the variational parameters, which can be considered as angles or also evolution times. The exact values for the parameters $\{\gamma_k, \beta_k\}$ are obtained by maximizing the expectation value $\langle H_C \rangle$ with respect to the QAOA state. As we had limited access to quantum hardware, in our workflow, we first performed statevector simulation in order to find good values of these parameters. Once we fixed these parameters, for each problem instance we have one correponding quantum circuit, that we proceeded to use on actual quantum hardware as well as in statevector simulation. We used two quantum devices, available on IBM Quantum platform [21] - Fez and Torino, which have 156 and 133 qubits respectively. Statevector simulation was performed on the HEMUS [20]. Our statevector simulation was exact in the sense that we can obtain the actual probabilities of the various bitstrings and detect the bitstrings with highest probability without sampling error (disregarding the rounding errors). As we will see in the next section, the candidate bitstrings obtained in this way are much closer to solution of the original problem.

We also consider the candidate bitstrings obtained via sampling in the same conditions. The natural candidates are in this case bitstrings that appear most frequently. Obviously, the difference with the previous case will come from the use of limited number of shots, so we can expect that the bitstring with highest probability may not always appear with highest counts. Increasing the number of qubits presents some other complications in selecting the candidates, as we may face a situation where bitstrings rarely repeat. In such cases we would need to implement a data-mining algorithm to select bitstrings based on counts of appearances of substrings. Since our limitation in this work has been mostly the number of qubits that we can simulate, we did not have to deal with this problem and it will remain for future work.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

3 Postprocessing procedure

From statevector simulation we can obtain the probabilities of bitstrings directly (disregarding numerical errors), while hardware execution or sampling from the statevector simulation results in empirical distributions. The postprocessing step, which is the focus of this work, applies optimisation algorithms to the top bitstrings, obtained in these 3 ways. One logical approach is to only use the most frequent bitstring (we call this top1). Another is to apply the same optimisation to multiple bitstrings with high probabilities or counts, e.g., using the top 10 bitstrings. All the computations required in this case are done on classical hardware, the objective function being simply the number of cuts and the inputs being the bitstrings. The algorithms tested were as follows:

- Simulated Annealing: The input bitstring is the initial state for simulated annealing. The algorithm performs bit flips according to a temperature schedule, accepting changes that do not improve the objective function with a probability that decreases over time [9,18].
- Greedy Improvement: Each measured bitstring is tested for possibility of improvement via single-bit updates that strictly increase the objective value until a local maximum is reached [15,12].
- Binary Particle Swarm Optimisation A variant of particle swarm optimisation, where the particle coordinates are binary. Typically in such cases velocities are interpreted as probabilities [8,17].
- Genetic Algorithm: Since inputs are bitstrings, genetic algorithms present a natural choice. All elements in the initial population are obtained with random modification of the input bitstring [6,16].
- Hill Climbing: We attempt to improve the solution by flipping random bits and accepting improving solutions [13].
- Hill Climbing with restarting: Similar to the previous algorithm, but the process is restarted multiple times with the hope to obtain better local maximum [10].

4 Use of Low-Discrepancy Sequences in the optimisation algorithms

Most optimisation algorithms incorporate randomness in their steps. Low - discrepancy sequences (LDS) are infinite sequences that have special construction that guarantees that their discrepancy (a measure of uniformity of distribution) has order of $(logN)^s/N$, where s is the dimension and N is the number of points. Usually such sequences combine deterministic and stochastic elements, in which case we speak of scrambled sequences. It has been shown that scrambling provides theoretical and practical benefits. Scrambling achieves better error estimates, and allows error estimates to be computed by repeating the computations with different scramblings. It also removes certain undesirable artefacts and is generally safe to use. Sobol sequences [19] are digital sequences in base 2 which

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

are widely used in high-dimensional numerical integration, especially in financial applications.

One theoretical justification for the use of low-discrepancy sequences is the Koksma–Hlawka inequality [see, e.g., Niederreiter, 1992 [3]], which relates integration error to discrepancy of the sequence and variation of the integrand. It should be noted that this inequality has mostly theoretical value, while in practice scrambled Sobol sequences have been shown to produce excellent results and that is why we decided to test their use in the optimisation algorithms above, the justification being the supposed better coverage of the underlying solution spaces.

So far we implemented LDS-based variants of the algorithms that are population - based (genetic, binary particle swarm optimisation) by using the sequences in the creation of the initial random population. In the hill climbing algorithms we integrated LDS-based sampling in the selection of the successive bits to flip. In the next section we discuss numerical results obtained using the various algorithms and their LDS versions where applicable.

5 Numerical Results and Discussion

In this section we discuss the results from applying the various optimisation algorithms, in some cases including LDS-based modifications, on the results from 3 types of computations. To achieve fair comparison, we used approximately equal number of evaluations of the objective function, which resulted in approximately equal computational times.

The direct probability evaluation that is obtained from the statevector-based simulation can be expected to yield bitstrings with almost perfect objective values. When we ignore the fact that we can access these probabilities directly but instead perform shots-based simulation, we obtain simulation counts. Ranking bitstrings based on these counts results in very similar solutions, with the practical limitation on number of shots leading to inability to detect the bitstring with the highest probability. The nature of the error is statistical, but our experience is that the results between using probabilities or simulation counts are very similar.

The more interesting situation is when the counts are obtained with hardware execution and measurement, where we observe substantial differences due to relatively large noise effects. In Table 1 we see comparison of the performance of various optimisation methods when probabilities are used directly. We denote by Δ the improvement observed, i.e., the difference between initial value of the objective function for the bitstring and the final value after local optimisation/refinement. As inputs to the optimisation algorithms we used either the top bitstring, i.e., the one with the highest probability, or the top 10 bitstrings, consecutively.

We compute not only best but also average improvements for these top10 bitstrings as performance metrics. In this table we observe that the Simulated Annealing produces best average Top10 performance, while multiple algorithms

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

are tied if only the best results are considered. In Table 2 we see similar comparison, but using simulation counts in the same problems. There is slight difference in the starting point, as apparently the bitstring with highest probability did not get highest count, because of the limited number of shots, but was still within the top10. Again the best performance is produced by the Simulated Annealing algorithm for the average of top10, while multiple algorithms are able to find perfect solution, when starting from the top1 bitstring. In Table 3 we see the same comparison based on hardware counts as a selector for the bitstring candidates. We immediately notice how the top1 bitstring gives us a difficult starting point, while the best of top10 has significantly better objective function. Again the average performance is best for the SA algorithm, while all algorithm were able to reach a perfect solution when starting from top10.

Table 1. Comparison of optimisation methods when using statevector-simulation probabilities.

Method	Top1 (old → new)	Avg Top10 (old → new)	Best Top10 (old → new)
Random-Restart Hill Climbing	14 → 21 ($\Delta + 7$)	14.000 → 20.400 ($\Delta + 6.400$)	14 → 21 ($\Delta + 7$)
Hill Climbing	14 → 19 ($\Delta + 5$)	14.000 → 19.800 ($\Delta + 5.800$)	14 → 21 ($\Delta + 7$)
Greedy Algorithm	14 → 20 ($\Delta + 6$)	14.000 → 20.000 ($\Delta + 6.000$)	14 → 20 ($\Delta + 6$)
Simulated Annealing	14 → 21 ($\Delta + 7$)	14.000 → 20.900 ($\Delta + 6.900$)	14 → 21 ($\Delta + 7$)
Genetic Algorithm	14 → 19 ($\Delta + 5$)	14.000 → 19.600 ($\Delta + 5.600$)	14 → 20 ($\Delta + 6$)
Binary PSO	14 → 20 ($\Delta + 6$)	14.000 → 20.800 ($\Delta + 6.800$)	14 → 21 ($\Delta + 7$)
BEST METHOD (SVPROB): anneal | top1 = 21, avg top10 = 20.900, best top10 = 21			

Table 2. Comparison of optimisation methods when using statevector-simulation counts.

Method	Top1 (old → new)	Avg Top10 (old → new)	Best Top10 (old → new)
Random-Restart Hill Climbing	13 → 21 ($\Delta + 8$)	13.500 → 20.800 ($\Delta + 7.300$)	14 → 21 ($\Delta + 7$)
Hill Climbing	13 → 20 ($\Delta + 7$)	13.500 → 20.400 ($\Delta + 6.900$)	14 → 21 ($\Delta + 7$)
Greedy Algorithm	13 → 20 ($\Delta + 7$)	13.500 → 20.100 ($\Delta + 6.600$)	14 → 21 ($\Delta + 7$)
Simulated Annealing	13 → 21 ($\Delta + 8$)	13.500 → 21.000 ($\Delta + 7.500$)	14 → 21 ($\Delta + 7$)
Genetic Algorithm	13 → 20 ($\Delta + 7$)	13.500 → 20.400 ($\Delta + 6.900$)	14 → 21 ($\Delta + 7$)
Binary PSO	13 → 20 ($\Delta + 7$)	13.500 → 20.500 ($\Delta + 7.000$)	14 → 21 ($\Delta + 7$)
BEST METHOD (SIMCNT): anneal | top1 = 21, avg top10 = 21.000, best top10 = 21			

Table 3. Comparison of optimisation methods when using quantum hardware measurement counts.

Method	Top1 (old → new)	Avg Top10 (old → new)	Best Top10 (old → new)
Random-Restart Hill Climbing	10 → 21 ($\Delta + 11$)	11.300 → 20.900 ($\Delta + 9.600$)	15 → 21 ($\Delta + 6$)
Hill Climbing	10 → 21 ($\Delta + 11$)	11.300 → 20.000 ($\Delta + 8.700$)	15 → 21 ($\Delta + 6$)
Greedy Algorithm	10 → 21 ($\Delta + 11$)	11.300 → 20.300 ($\Delta + 9.000$)	15 → 21 ($\Delta + 6$)
Simulated Annealing	10 → 21 ($\Delta + 11$)	11.300 → 21.000 ($\Delta + 9.700$)	15 → 21 ($\Delta + 6$)
Genetic Algorithm	10 → 20 ($\Delta + 10$)	11.300 → 20.400 ($\Delta + 9.100$)	15 → 21 ($\Delta + 6$)
Binary PSO	10 → 21 ($\Delta + 11$)	11.300 → 20.600 ($\Delta + 9.300$)	15 → 21 ($\Delta + 6$)
BEST METHOD (HWCNT): anneal | top1 = 21, avg top10 = 21.000, best top10 = 21			

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

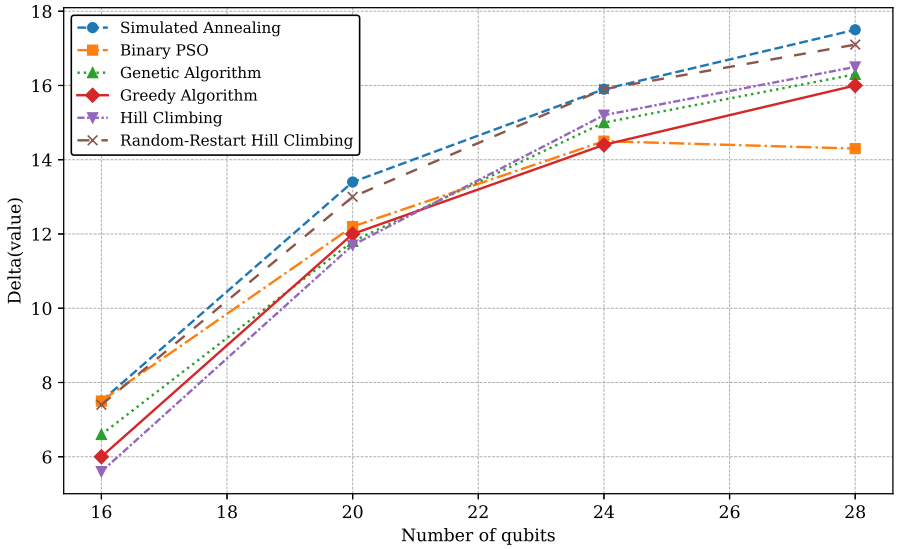

Fig. 1. Performance comparision of various optimization methods at circuit depth 8.

Table 4. Performance comparison table of optimization algorithms for hardware counts.

Hardware counts averaged over top-10 results							
Qubits	**depth**	**binary PSO**		**rand hillclimb**		**restart hillclimb**	
		random	sobol	random	sobol	random	sobol
20	2	25.1	23	24.2	24.8	**25.3**	25.2
24	2	28.2	26.1	29	29.2	**29.5**	**29.5**
28	2	33.5	31	34.2	34.5	**35.7**	35.6
24	10	28.3	26.4	28.8	28.8	**30.1**	**30.1**
28	10	34	33.6	34.8	34.4	**36.6**	**36.6**
Hardware counts averaged over top-1 results							
Qubits	**depth**	**binary PSO**		**rand hillclimb**		**restart hillclimb**	
		random	sobol	random	sobol	random	sobol
20	2	25	21	23	24	**26**	**26**
24	2	28	25	28	**29**	**29**	**29**
28	2	33	29	31	31	**34**	**34**
24	10	28	25	27	27	**29**	**29**
28	10	33	31	33	**36**	**36**	**36**

An exhaustive performance comparison between usual versions of the algorithms and their QMC implementations with scrambled Sobol sequences is shown in Table 4,Table 5 and Table 6. We can see that in many instances the best results (in boldface) are obtained from the QMC version of the methods.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

Table 5. Performance comparison table of optimization algorithms for simulation counts.

Simulation counts averaged over top-10 results							
Qubits	**depth**	**binary PSO**		**rand hillclimb**		**restart hillclimb**	
		random	sobol	random	sobol	random	sobol
20	2	25.4	22.5	24.6	24.5	**25.7**	25.6
24	2	29	26.3	28.4	28.2	**30.4**	30.2
28	2	33.3	29.9	34.4	34.8	**35.6**	35.5
24	10	29.2	26.6	28.8	29.1	**30.2**	29.9
28	10	34.5	31.2	30.2	34.1	**36.5**	**36.6**
Simulation counts averaged over top-1 results							
Qubits	**depth**	**binary PSO**		**rand hillclimb**		**restart hillclimb**	
		random	sobol	random	sobol	random	sobol
20	2	25	23	24	23	25.7	**26**
24	2	30	25	27	28	**30.4**	30
28	2	33	30	34	34	**35.6**	35
24	10	28	25	28	28	**30.2**	30
28	10	33	31	34	32	**36.5**	36

Table 6. Performance comparison table of optimization algorithms for probability counts.

Probability counts averaged over top-10 results							
Qubits	**depth**	**binary PSO**		**rand hillclimb**		**restart hillclimb**	
		random	sobol	random	sobol	random	sobol
20	2	25.1	22.9	24.2	24.5	26	25.9
24	2	28.6	26.3	27	27.7	29.6	25
28	2	33.7	30.9	34	34.2	**36**	29.5
24	10	28.7	26.5	28	28.7	**30.2**	30.1
28	10	34	30.6	34.7	34.4	**36.4**	36.3
Probability counts averaged over top-1 results							
Qubits	**depth**	**binary PSO**		**rand hillclimb**		**restart hillclimb**	
		random	sobol	random	sobol	random	sobol
20	2	**26**	24	24	24	**26**	25
24	2	29	25	28	29	**30**	**30**
28	2	33	30	31	32	**35**	**35**
24	10	29	24	28	28	**31**	**31**
28	10	33	31	33	32	**36**	**36**

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

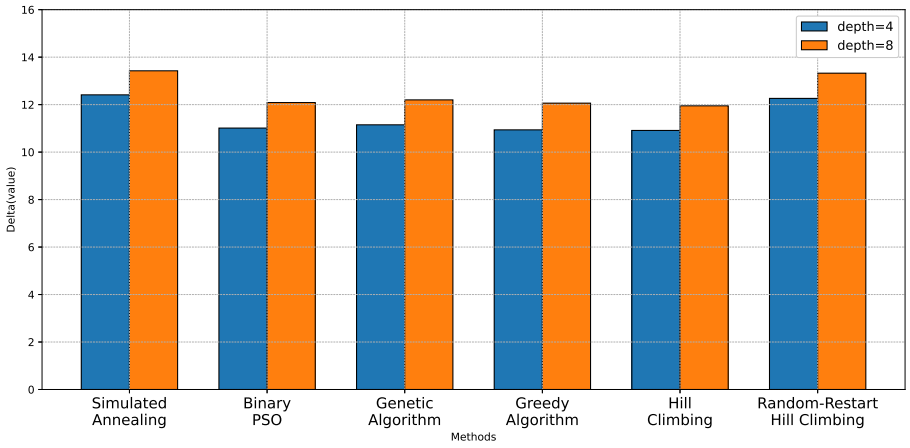

Fig. 2. Performance comparison for hardware execution of circuits with depth 4 and depth 8.

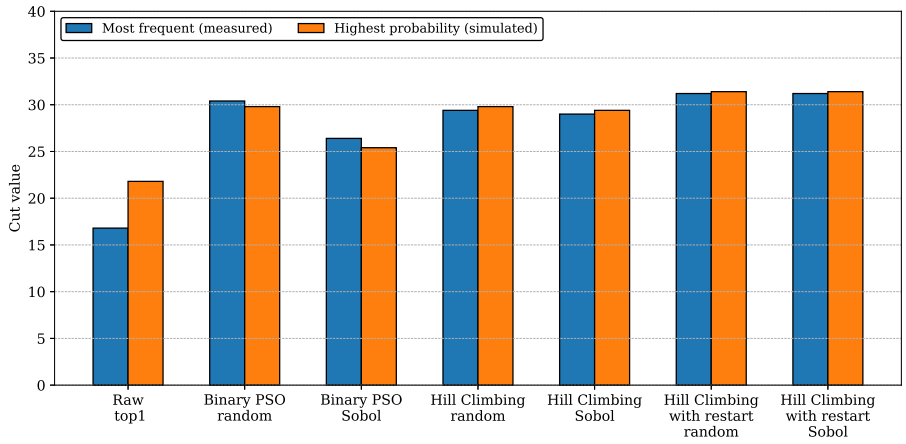

Fig. 3. Performance comparison of various optimisation algorithms starting from the top bitstrings obtained execution on quantum hardware and simulation, for 24 qubits.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

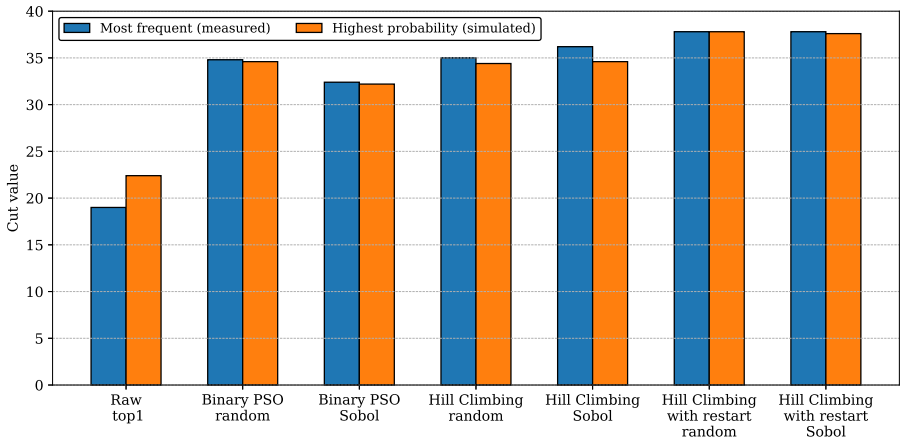

Fig. 4. Performance comparison of various optimisation algorithms starting from the top bitstrings obtained execution on quantum hardware and simulation, for 28 qubits.

Visual comparison of the average performance on top10 of the various algorithms can be seen in Fig 1. It can be seen that the average improvement (Δ) under different number of qubits, with fixed circuit depth $p = 8$ is best for Simulated Annealing and Random-restart Hill Climbing.

On Fig 2 we can see the difference in performance for the different methods when depths 4 and 8 were, for hardware execution. Despite the inherent noise, the larger circuits produced better results after optimisation, for all methods.

Fig 3 and Fig 4 present additional data, comparing the performance of the various optimisation algorithms (using both pseudo-random numbers and Sobol low-discrepancy sequences), starting from the top bitstring coming from execution on quantum device (left) and from probabilities computing via the simulation. Fig 3 uses 24 qubits, while Fig 4 uses 28 qubits, with the number of repetitions fixed at 10. One can see that the simulation gives better initial estimates, but the optimisation algorithms achieve very similar results, i.e., the optimisation algorithms are efficient in dealing with the inherent noise.

6 Conclusions and Future Work

We investigated multiple goal-oriented classical postprocessing strategies, essentially performing local optimisations on QAOA measurement outcomes for the MaxCut problem in both simulation and hardware settings. We demonstrate high-quality cuts can be obtained, with significant differences in performance between the different algorithms. We observed that simulated annealing–based refinement consistently demonstrated strong empirical performance. Close results are also obtained by the random hill climbing algorithm. Furthermore, quasi–Monte Carlo (QMC)–enhanced variants of the stochastic algorithms with

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

the scrambled Sobol' sequences in many instances showed better performance, demonstrating potential for further investigation. It is obvious that objective-driven classical refinement postprocessing is a highly usable strategy for QAOA under the limitations of the current quantum hardware. Low-discrepancy sampling also offers a promising direction for improving the overall performance of such refinement procedures. We intend to proceed to larger graph instances, once we find a better way to fit the first QAOA parameter optimisation step entirely on quantum hardware. We should also study the influence of graph structure and sparsity on the performance of the optimisation strategies. Better use of the specific qualities of low-discrepancy sequences will be also of high importance, perhaps looking for a better-matching optimisation algorithm that can exploit the uniformity of distribution of low-discrepancy sequences.

Acknowledgments. The work was partially supported by the Centre of Excellence in Informatics and ICT under the Grant No BG16RFPR002-1.014-0018-C01, financed by the Research, Innovation and Digitalization for Smart Transformation Programme 2021-2027 and also used infrastructure from National Roadmap for RI, financially coordinated by the MES of the Republic of Bulgaria (grant No D01-98/26.06.2025).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Combarro E., Gonzalez-Castillo, S.: A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms. 1st edn. Packt Publishing, UK (2023)
2. Keller, A.: Monte Carlo and Quasi-Monte Carlo Methods. Springer International Publishing, Oxford, United Kingdom (2020)
3. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics, Philadelphia, PA (1992)
4. Owen, Art B.: Monte Carlo Book: the Quasi-Monte Carlo parts. (2019)
5. Alam, T., Qamar, S., Dixit, A., Benaida, M., Genetic Algorithm: Reviews, Implementations, and Applications. International Journal of Engineering Pedagogy (iJEP), 10(6), pp. 55–57, (2020). https://doi.org/https://doi.org/10.3991/ijep.v10i6.14567
6. Albadr, M.A., Tiun, S., Ayob, M., AL-Dhief, F. Genetic Algorithm Based on Natural Selection Theory for Optimization Problems.Symmetry, 12, 1758, (2020) https://doi.org/10.3390/sym12111758
7. Benlic, U., Hao, J.: Breakout Local Search for the Max-Cutproblem. In: Engineering Applications of Artificial Intelligence (2013), vol. 26, pp. 1162–1173. (2012). https://doi.org/10.1016/j.engappai.2012.09.001
8. Bratton, D., Kennedy, J.: Defining a Standard for Particle Swarm Optimization. In: IEEE Access, IEEE Swarm Intelligence Symposium, ISBN: 1-4244-0708-7, pp. 120–127, Honolulu, HI, USA, (2007). https://doi.org/10.1109/SIS.2007.368035
9. Delahaye, D., Chaimatanan, S., Mongeau, M.: Simulated Annealing: From Basics to Applications. In: Gendreau, M., Potvin, JY. (eds) Handbook of Metaheuristics. International Series in Operations Research and Management Science, vol 272. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_1

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30

10. Escamilla-Sernan, N., Seck-Tuoh-Mora J., Medina-Marin J., et. all: A Hybrid Search Using Genetic Algorithms and Random-Restart Hill-Climbing for Flexible Job Shop Scheduling Instances with High Flexibility. In: Metaheuristic Computing, Sci. 2022, vol. 12, no. 16, p. 8050 (2022). https://doi.org/10.3390/app12168050
11. Farhi, E., Goldstone, J., Gutmann, S..: A Quantum Approximate Optimization Algorithm. In: Center for Theoretical Physics Massachusetts Institute of Technology (2014), Cambridge, MA 02139 (2016). https://doi.org/10.48550/arXiv.1411.4028
12. Ferber, P., Cohen, L., Seipp, J., Keller, T.: Learning and Exploiting Progress States in Greedy Best-First Search. In: Thirty-First International Joint Conference on Artificial Intelligence (2022), pp. 4740–4746, (2022). https://doi.org/10.24963/ijcai.2022/657
13. Filho, C., Costa, M., Filho, J., Oliveira A.: Using a random restart hill-climbing algorithm to reduce component assembly time in printed circuit boards. In: IEEE International Conference on Industrial Technology 2010, pp. 1706–1711. Via del Mar, Chile, (2010). https://doi.org/10.1109/ICIT.2010.5472443
14. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. In: Journal of the Association for Computing Machinery (1995), Vol. 42, pp. 1115–1114. (1995). https://doi.org/10.1145/227683.227684
15. Heusner, M., Keller, T., Helmert, M.: Understanding the Search Behaviour of Greedy Best-First Search. In: In Proc. International Symposium on Combinatorial Search (2017), LNCS, vol. 8(1), pp. 47–77. (2017) https://doi.org/10.1609/socs.v8i1.18425
16. Kumar, M.: Genetic Algorithm - an Approach to Solve Global Optimization Problems. In: Indian Journal of Computer Science and Engineering, Bajpai P. et al., vol. 1, pp. 199–206. ISSN : 0976-5166 (2010). https://doi.org/10.10007/1234567890
17. Shami, T., El-Saleh. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M., Mirjalili, S.: Particle Swarm Optimization: A Comprehensive Survey. In: IEEE Access, vol. 10, pp. 10031–10061, IEEE Xplore (2022). https://doi.org/10.1109/ACCESS.2022.3142859
18. Xianggen, L., Pengyong, L., Fandong, M., Hao, Z., Huasong, Z., Jie, Z., Lili, M., Sen, S.: Simulated annealing for optimization of graphs and sequences. In: Neurocomputing, vol. 465, pp. 310–324., ISSN 0925-2312, (2021). https://doi.org/10.1016/j.neucom.2021.09.003.
19. Sobol, I.: On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput. Math. Math. Phys. 7, 86–112 (1967)
20. HEMUS supercomputer, https://www.top500.org/system/180208, (2026)
21. Research IBM Quantim Platform, https://quantum.cloud.ibm.com, last accessed 2026/02/20

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29918-5_30