Incorporating Quantum Advantage in Quantum Circuit Generation through Genetic Programming

CHRISTOPH STEIN, Karlsruhe Institute of Technology, Germany
MICHAEL FÄRBER, TUD Dresden University of Technology, Germany

Designing efficient quantum circuits that leverage quantum advantage compared to classical computing has become increasingly critical. Genetic algorithms have shown potential in generating such circuits through artificial evolution. However, integrating quantum advantage into the fitness function of these algorithms remains unexplored. In this paper, we aim to enhance the efficiency of quantum circuit design by proposing two novel approaches for incorporating quantum advantage metrics into the fitness function of genetic algorithms.[1] We evaluate our approaches based on the Bernstein-Vazirani Problem and the Unstructured Database Search Problem as test cases. The results demonstrate that our approaches not only improve the convergence speed of the genetic algorithm but also produce circuits comparable to expert-designed solutions. Our findings suggest that automated quantum circuit design using genetic algorithms that incorporate a measure of quantum advantage is a promising approach to accelerating the development of quantum algorithms.

CCS Concepts: • **Hardware → Quantum computation**; • **Computer systems organization → Quantum computing**; • **Computing methodologies → Genetic algorithms**; • **General and reference** → *Empirical studies.*

Additional Key Words and Phrases: Quantum Computing, Genetic Programming, Genetic Algorithms, Quantum Circuit Generation, Quantum Advantage

ACM Reference Format:
Christoph Stein and Michael Färber. 2024. Incorporating Quantum Advantage in Quantum Circuit Generation through Genetic Programming. *ACM Trans. Quantum Comput.* 37, 4, Article 111 (August 2024), 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In contrast to classical computing, quantum computing makes explicit use of phenomena that only exist in the quantum realm, such as superposition states and quantum entanglement. Based on these differences, quantum algorithms have been developed that show significant improvements in time complexity compared to their best performing classical counterparts (e.g., [6, 10, 17]). Despite their potential, however, comparatively few quantum algorithms with a significant quantum advantage have been found over the years [18, 19, 21]. One promising approach to bridge this gap is through the use of genetic algorithms. These algorithms have shown to be capable of producing quantum circuits for various tasks from entanglement production [2, 16] to algorithm development [1, 13, 19] and quantum error correction [20].

Within the literature on quantum circuit generation through the use of genetic algorithms, approaches mainly differ in the way they represent individual quantum circuits, the quantum gates they utilize, and how they measure the fitness of a generated circuit or algorithm (see Sec. 2). This last aspect, the fitness function, determines the

[1]The code for this paper is available at https://github.com/westoun/gp4qc.

Authors' Contact Information: Christoph Stein, christoph.stein@student.kit.edu, Karlsruhe Institute of Technology, Karlsruhe, Germany; Michael Färber, michael.faerber@tu-dresden.de, TUD Dresden University of Technology, Dresden, Germany.

ACM 2643-6817/2024/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

shape of the search space the genetic algorithm will seek to find the optimal circuit or algorithm within. Multiple authors defined their fitness function to not just include the precision of the generated circuits on the chosen target task, but also their complexity and gate count [8, 19, 20].

However, while previous work successfully showed the feasibility of learning valid quantum circuits through genetic algorithms for a wide range of studied problems [1, 2], to the best of our knowledge, no approach exists that incorporates the measurement of quantum advantage in the genetic algorithm itself. Even if the generated circuits are able to produce the desired target states of a problem, by omitting the use of superposition and entanglement, they can fail to produce an advantage over classical non-quantum solutions in terms of their time or space complexity.

In this paper we investigate whether incorporating a measure for the quantum advantage in the fitness function of a genetic algorithm has an impact on the quality of the generated circuits and the convergence behavior of the genetic algorithm. To evaluate our approach, we chose the Bernstein-Vazirani Problem [3] and the Unstructured Database Search Problem [10] as test studies. Since quantum algorithms with a proven quantum advantage exist for each of these test problems, generated circuits can be compared against the state of the art in regard to their performance, circuit architecture, and complexity. While most work on the use of genetic algorithms for quantum circuit generation is evaluated based on the resulting states of the produced quantum circuits, we additionally evaluate with respect to convergence behavior and speed across multiple experiment rounds to assess the robustness of our approaches.

In short, we make the following main contributions in the paper:

- We propose a novel fitness function for genetic algorithms that incorporates an explicit measure of the degree to which quantum advantage is achieved.
- We propose a second fitness function that incorporates quantum advantage indirectly based on the differences between quantum and classical computation.
- We evaluate our approaches based on two established test studies, the Bernstein-Vazirani Problem [3] and the Unstructured Database Search Problem [10], and report the best performing circuits generated by each fitness function as well as the convergence behavior of the genetic algorithm.

The rest of this paper is structured as follows: Sec. 2 contains an overview of related work in the field of quantum circuit generation through genetic algorithms. Sec. 3 elaborates on the suggested approach and introduces the proposed fitness functions in detail. The results of experiments run on the Bernstein-Vazirani Problem as well as the Unstructured Database Search Problem are presented in Sec. 4. Sec. 5 distills the implications of the afore presented results and Sec. 6 concludes and points to future work.

2 Related Work

Multiple techniques have been developed with the goal of automatically generating quantum circuits and quantum algorithms. While recent work often makes use of reinforcement learning to select which gates to add to a circuit [11, 14], a vast body of work exists that uses genetic algorithms to generate quantum circuits for various tasks and performance goals. Since this paper builds on recent work in the context of genetic algorithms, we focus on this aspect in the following paragraphs.

2.1 Fitness Functions

The fitness function serves as the basis of selection within a genetic algorithm. Its choice determines which individuals are more likely to survive and, hence, which traits are more likely to spread throughout the population. Since most quantum circuits are developed to fulfill a specific computational need, the difference between the actual and the desired final state of the quantum circuit forms the basis of most fitness functions used in the

Table 1. Related approaches of genetic algorithms for quantum circuit design, sorted by publication year.

Authors	**Fitness Function**	**Use of Parameterized Gates**
Gemeinhardt et al. [8]	Multi-objective including accuracy, gate count, and circuit depth.	Nelder-Mead optimization.
Tandeitnik et al. [20]	Custom quality score combined with circuit depth.	None.
Franken et al. [7]	Deviation between realized and target Hamiltonian.	Mutation operator.
Ahsan et al. [1]	Absolute deviation between target and desired state.	None.
Bautu et al. [2]	Absolute deviation between target and desired state.	None.
Massey et al. [13]	Magnitude and phase combination, with an efficiency term.	Mathematical operators.
Massey et al. [12]	Adapted from Spector et al. with threshold adjustments.	As per Spector et al.
Rubinstein et al. [16]	Normalized deviation between target and desired state.	Randomized rotation parameters ($[-\pi, \pi]$).
Williams et al. [22]	Absolute deviation between desired and realized unitary.	None.
Spector et al. [19]	Multi-step fitness involving hits, deviation, and circuit length.	Mathematical operators.

literature. Table 1 provides an overview of the fitness functions and use of parameterized gates of related genetic algorithms for quantum circuit design.

Bautu et al. [2] and Ahsan et al. [1] use the absolute deviation between the actual and the target state of a quantum circuit as its fitness value, while [22] utilizes the absolute deviation as the starting point for a custom ranking approach. Gemeinhardt [8] uses a combination of accuracy, gate count, parameter count, and other measures in connection with the multi-objective optimization algorithm NSGA-III [5] to account for the trade-offs between circuit performance and circuit complexity. [20] also incorporates the trade-off between performance and complexity, although they do so through a weighted combination of a quality measure and circuit depth. While the above mentioned approaches apply both a measure for the performance and complexity of the generated circuits in their fitness functions, to the best of our knowledge no approach exists that incorporates a measure of quantum advantage, the driving force behind quantum algorithm development.

Acknowledging the probabilistic nature of quantum computing, Spector et al. [19] developed a custom multi-component fitness function, further refined by [12]. The first component, hits, rewards any additional test case where the probability of producing the correct solution is above a threshold of 0.52 (0.5 + 0.02 to account for rounding errors). The second component, correctness, adds an error term for each incorrect test case based on the difference between the desired result and the produced distribution. Finally, if a circuit passes all test cases, an efficiency term is applied based on the amount of gates used by a circuit. In total, this fitness function first drives the population towards optimizing the amounts of correct test cases, nudging it to reduce circuit complexity from that point on. The fitness functions introduced in this work can be considered as extensions of the fitness function of Spector et al. [19], further incorporating a measure of quantum advantage into the optimization procedure.

2.2 Gate Types and Parameterized Gates

Regarding the types of gates used, related work can be dichotomized based on its use or avoidance of parameterized gates. While all gates have at least one parameter describing which qubit(s) they apply to, some gates, like the phase gate, take additional parameters influencing the scope of the manipulation they perform. These gates usually correspond to controlled rotations about one or more axes, with their parameters falling in the range of $[-\pi, \pi]$.

Since these parameterized gates allow for a fine control of the states of one or more qubits, they form the basis of many advanced quantum algorithms like the variational quantum eigensolver [15] or the quantum fourier transform [4]. For their use in the context of genetic programming, suitable parameters for those gates have to be learned. While tree-based approaches like [19] and [12] incorporate parameter optimization through the use of mathematical operators in their representation scheme, other approaches like [8] incorporate parameter optimization as an additional step in their genetic algorithm. [7] and [9] adjust rotation parameters through the use of an additional mutation operator within the genetic algorithm itself.

The approach presented in this work will follow the work of Gemeinhardt et al. [8] and include parameter optimization as an additional step in the genetic algorithm. While this does require additional computational resources during the execution, it greatly enhances the effectiveness of the algorithm on complex problems.

3 Method

The goal of this work is to develop a genetic algorithm for quantum circuit generation that incorporates in its fitness function whether or not a quantum advantage is achieved. Two proposed fitness functions, each approaching this objective from a different angle, are introduced and defined in Sec. 3.1 and Sec. 3.2 respectively.

3.1 Direct Incorporation of Quantum Advantage

Most established quantum algorithms define their quantum advantage based on the amount of black box oracle calls they require to solve a computational problem. This number is then contrasted to the amount of oracle calls the best performing classical (non-quantum) algorithm would need [3, 6, 10]. In the case of the Unstructured Database Search Problem, for example, the best performing classical algorithm would require on average $N/2$ oracle calls to find a specific entry within an unstructured list. Grover's Algorithm [10], in contrast, has been shown to require only $O(\sqrt{N})$ oracle calls. In this section, we introduce a fitness function that incorporates this respective speedup, the quantum advantage a generated circuit achieves in contrast to its best performing classical counterpart.

In the work of Spector et al. [19], the fitness function comprises of multiple factors that each target different aspects of the circuits to be created. At first, each circuit is shaped to solve as many test cases as possible. Then, once the algorithm finds a circuit that passes all test cases, the fitness function further optimizes by preferring circuits that make use of fewer gates.

While the amount of gates used serves as an approximate measure of the simplicity of a circuit, it is an insufficient indicator of the degree of quantum advantage achieved. In contrast to the amount of oracle calls employed, the amount of gates used by a circuit does not indicate the scaling behavior of the underlying algorithm. Furthermore, the amount of gates used for implementing an algorithm heavily depends on the gate set used, a concern driven by hardware and design considerations, irreverent of the underlying algorithm itself.

In this work, in addition to incorporating the amount of a gates as a factor indicating circuit simplicity, we incorporate a custom measure for the degree of quantum advantage achieved by a circuit in its fitness value computation.

Algorithm 1 shows the first novel fitness function proposed in this paper. It iterates over all possible test cases (input values or oracle implementations) and compares the probability distributions generated by the current

Algorithm 1 The fitness function of Spector et al. [19], adjusted to directly incorporate a measure of quantum advantage.

1: $probabilities \leftarrow evaluate(circuit)$
2: $hits \leftarrow \#TestCases$
3: $error \leftarrow 0$
4: **for** i = 0; i < #TestCases; i++ **do**
5: $\quad j \leftarrow getTargetIndex(targetProbabilities[i])$
6: $\quad$ **if** probabilities[i][j] >= 0.52 **then**
7: $\quad\quad hits \leftarrow hits - 1$
8: $\quad$ **else**
9: $\quad\quad \delta \leftarrow jensenShannonDistance(probabilities[i], targetProbabilities[i])$
10: $\quad\quad error \leftarrow error + \delta$
11: $\quad$ **end if**
12: **end for**
13: **if** hits > 0 **then**
14: $\quad fitness \leftarrow hits + \frac{error}{max(hits,1)}$
15: **else**
16: $\quad fitness \leftarrow \frac{countOracleGates(circuit)}{classicalOracleCalls} + \frac{countGates(circuit)}{100000}$
17: **end if**
18: **return** fitness

circuit with the desired. If the desired target state for each test case is measured with a probability of at least 0.52 (based on [19]), the hits counter that measures the amount of incorrect test cases is reduced by one. If the produced state does not meet this threshold for a test case, the Jensen Shannon Distance between the actual and desired probability distribution is computed and added to the existing error term. After all test cases have been evaluated, the fitness score is computed based on the amount of incorrect test cases measured by the hits counter. If at least one test case has not been passed, the fitness score is defined as the sum of the amount of remaining test cases and the average error term of these cases. If all test cases have been passed, the fitness score is computed based on the amount of gates used in the circuit and the relation of quantum oracle calls to classical oracle calls.

Using this fitness function, the genetic algorithm is driven to first improve the amount of correct test cases (line 13-15 in Alg. 1). Then, once all test cases have been cleared, an additional fitness term is applied that serves as a punishment factor for the amount of quantum oracle calls used (line 16 in Alg. 1). Since many quantum algorithms define the degree of quantum advantage achieved based on the lower amount of oracle calls needed, we divide the amount of oracle calls used in a quantum circuit by the amount of oracle calls the best performing classical (non-quantum) algorithm would require.

The fewer oracle calls a quantum circuit uses in relation to the best performing classical algorithm, the lower its comparative fitness value and the higher its likelihood of surviving selection. Thereby, circuits that solve all test cases but use fewer oracle calls are more likely to move on to the next generation. In the end, these circuits show a higher degree of quantum advantage. The remaining term of Spector's original fitness function, efficiency, is left as it was originally defined.

3.2 Indirect Incorporation of Quantum Advantage through Constraints

As shown in [3], any computation a quantum computer can do can also be performed by a classical (non-quantum) computer. The difference between quantum and non-quantum computers lies not in *what* they are able do, but in *how efficiently* they are able to do it. Quantum computers are able to exhibit quantum advantages because in

a quantum computer superposition and entanglement happen at the physical level. Thus, the fitness function introduced in this section punishes all circuits that do not make use of entanglement and superposition, since such circuits cannot exhibit significant quantum advantage.

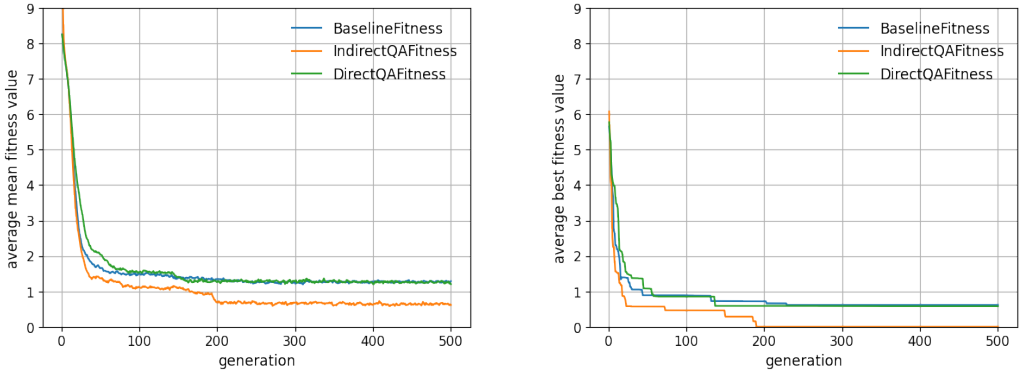
From a quantum circuit perspective, multiple gates can be used to create a state of *superposition*. Each of these gates transforms a qubit from one of its base states to a gate-specific superposition state, thereby fulfilling the proposed superposition condition. While the most obvious candidate, the Hadamard gate, creates an equal superposition state between both basis states, rotational gates targeting the x- or y-axis can also be used to create superposition states. Furthermore, same holds for the controlled versions of each of those gates.

To achieve a state of *entanglement* in which the states of individual qubits cannot be described independently of one another, the state of one qubit has to depend on the state of another. In the quantum circuit representation, this resembles the idea of controlled gates, where a specific operation is only applied to a qubit if the corresponding control qubit is in a specific state. In the case of the CNOT gate, for example, the Pauli-X gate is applied to the target qubit iff the control qubit is in state $|1\rangle$. The degree of entanglement between the qubits of a system can thereby be increased by applying controlled gates to previously un-entangled qubits.

Algorithm 2 The fitness function of Spector et al. [19], adjusted to incorporate the potential for quantum advantage through the aforementioned constraints.

$probabilities \leftarrow evaluate(circuit)$
$hits \leftarrow \#TestCases$
$error \leftarrow 0$
for i = 0; i < #TestCases; i++ **do**
    $j \leftarrow getTargetIndex(targetProbabilities[i])$
    if probabilities[i][j] >= 0.52 **then**
        $hits \leftarrow hits - 1$
    else
        $\delta \leftarrow jensenShannonDistance(probabilities[i], targetProbabilities[i])$
        $error \leftarrow error + \delta$
    end if
end for
$fitness \leftarrow 0$
if not containsSuperpositionGates(circuits) **then**
    $fitness \leftarrow fitness + (\#TestCases + 1)$
end if
if not containsEntanglementGates(circuits) **then**
    $fitness \leftarrow fitness + (\#TestCases + 1)$
end if
if hits > 0 **then**
    $fitness \leftarrow fitness + hits + \frac{error}{max(hits,1)}$
else
    $fitness \leftarrow fitness + \frac{countGates(circuit)}{100000}$
end if
return fitness

Similar to our approach proposed in Sec. 3.1, we incorporate in this subsection the aforementioned quantum-specific differences in the genetic algorithm as aspects of the fitness function. In contrast to Sec. 3.1, however,



(a) Average Mean Fitness

(b) Average Minimum Fitness

Fig. 1. Averaged fitness values on the Bernstein-Vazirani Problem.

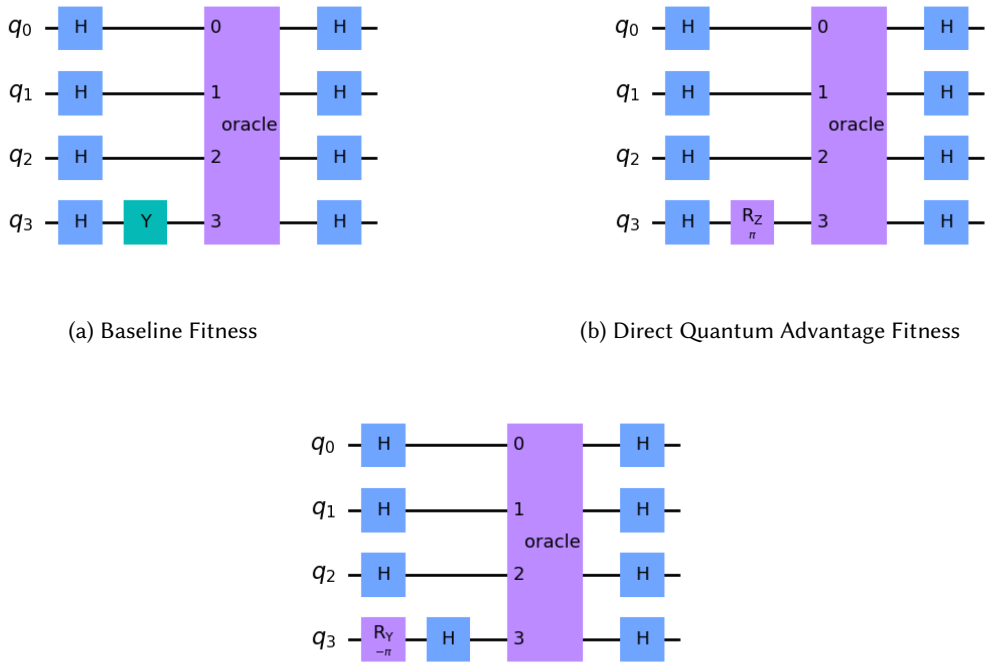
these factors are incorporated as constraints *prior* to the optimization of fitness test cases (see lines 14-19 in Alg. 2). By opting the genetic algorithm to first create circuits with the potential of a quantum advantage, the algorithm will deprioritize any circuit that might solve the problem but does not include any gates incorporating the potential for quantum advantage.

Considering that the maximum value of the baseline fitness function defined in [19] is equal to the amount of test cases used, the punishment terms for violating the aforementioned constraints shall take a value slightly above this maximum, in this case $\#TestCases + 1$ (lines 14-19 in Alg. 2). That way, the genetic algorithm is incentivized to optimize constraint satisfaction first, before improving performance on the specific problem set. The absolute value of the punishment term does not matter, as long as it is large enough to avoid trade-offs against other fitness components down the line.

4 Evaluation

To evaluate the approaches proposed, we chose the Bernstein-Vazirani Problem [3] and the Unstructured Database Search Problem [10] as test studies. For each of the test studies, experiments were run with the fitness function of Spector et al. [19] (*BaselineFitness*), our fitness function proposed in Sec. 3.1 (*DirectQAFitness*), and our fitness function proposed in Sec. 3.2 (*IndirectQAFitness*). Since genetic algorithms make strong use of random choices both during population initialization as well as during each generation, each experiment configuration was run 12 times.

During each experiment run, the population size was set to 1,000, the probability for crossover was set to 0.4, the probability of randomly swapping individual gates was set to 0.03, and elitism was performed with 10% of the population. These parameters were chosen based on experience and systematic trial and error ahead of the actual experiment runs. On the Unstructured Database Search Problem, the chromosome length was set to 30 and each algorithm was run for 800 generations. On the Bernstein-Vazirani Problem, the chromosome length was set to 15 and each algorithm was run for 500 generations. Different parameters have been chosen for each of the test studies to account for the difference in complexity between both problems.



(a) Baseline Fitness

(b) Direct Quantum Advantage Fitness

(c) Indirect Quantum Advantage Fitness

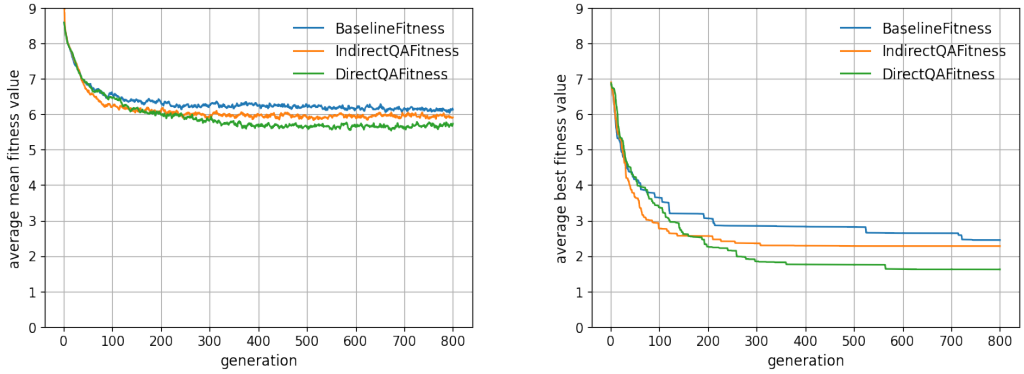
Fig. 2. Best performing circuits of each fitness function for the Bernstein-Vazirani Problem.

4.1 Results on the Bernstein-Vazirani Problem

Fig. 1 shows the development of the mean and minimum fitness values for each generation of the genetic algorithm, averaged across multiple experiment runs, and for each of the fitness functions used. *BaselineFitness* refers to the fitness function proposed by Spector et al. in [19].

While *BaselineFitness* and *DirectQAFitness* show similar convergence behavior, both in regard to their averaged mean as well as their averaged minimum fitness values, the fitness values for *IndirectQAFitness* converge faster and reach lower overall fitness values. The higher fitness values of *IndirectQAFitness* during early generations can be explained through the punishment terms applied in the fitness value computation of this fitness function (s. Sec. 3.2). Once the superposition and entanglement constraints have been satisfied, *IndirectQAFitness* outperforms the other two fitness functions in terms of its average fitness values.

Fig. 2 lists the best performing circuits generated by each of the experiment configurations. In this context, a circuit is considered better the lower its corresponding fitness value is. While all of the generated circuits follow the structure of the state of the art solution [3], individual circuits vary in the way they manipulate the ancillary qubit q3. Aside from a global phase factor, each of these circuits is equivalent to the state of the art circuit of [3], which was created manually. Each of these circuits returns the desired result state with a probability of 1.



(a) Average Mean Fitness

(b) Average Minimum Fitness

Fig. 3. Averaged fitness values on the Unstructured Database Search Problem, grouped by fitness function.

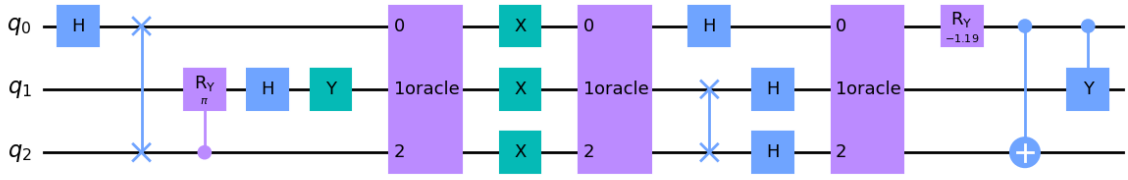


Fig. 4. Best performing circuit for *BaselineFitness* on the Unstructured Database Search Problem.

4.2 Results on the Unstructured Database Search Problem

Fig. 3 shows a slight out-performance of the *IndirectQAFitness* and *DirectQAFitness* in comparison to the *BaselineFitness* function in regards to both their averaged mean values as well as their averaged minimum fitness values. While the *IndirectQAFitness* has a higher averaged mean fitness value during early generations, due to the punishment terms applied in the fitness value computation, this fitness function converges faster, although to a less beneficial value compared to the *DirectQAFitness* function.

In general, the averaged minimum fitness values as well as the minimum fitness values of each experiment run are monotonically decreasing, due to the use of elitism in the genetic algorithm. In our implementation, elitism is realized by copying the best performing circuits of each generation into the next generation unaltered. As a consequence, the minimum fitness value of a population can only stay stable or decrease as generations progress.

Figures 4 to 6 show the best performing quantum circuits across all runs of each of the experiment configurations. Each of these circuits achieved a fitness value close to 0, indicating that it passed all possible test cases of the Unstructured Database Search Problem on 3 qubits.

In the following, we provide a case study by discussing specific parallels and differences between the generated best performing circuits and the state of the art solution from [10].

In addition to employing the same amount of oracle calls, the circuits in Fig. 4 and Fig. 5 follow a shared structure that is notably structurally different from the state of the art circuit. Instead of creating an equal

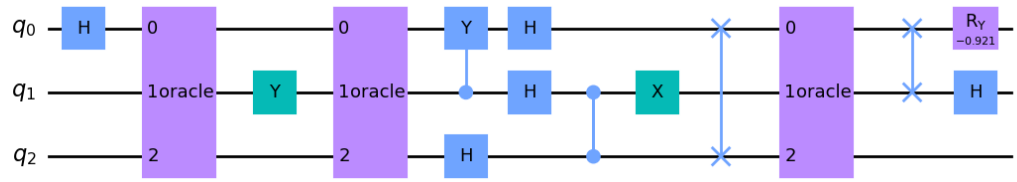


Fig. 5. Best performing circuit for *DirectQAFitness* on the Unstructured Database Search Problem.

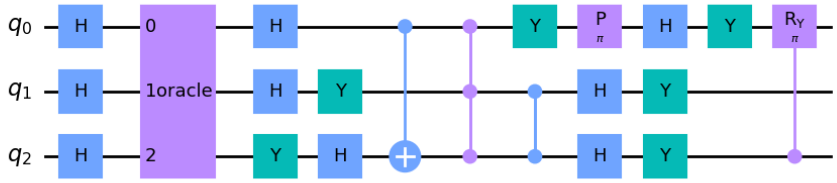


Fig. 6. Best performing circuit for *IndirectQAFitness* on the Unstructured Database Search Problem.

superposition at the beginning of the circuit, as the state of the art solution does, these circuits start by placing the system in a partial superposition state. The circuit illustrated in Fig. 5, for example, begins by putting only the first qubit in superposition. Since the state of the system after the first Hadamard gate is $|\psi\rangle = \frac{|000\rangle+|100\rangle}{\sqrt{2}}$, the first oracle call only produces an effect, if the desired target state of the oracle follows the schema $|?00\rangle$, where ? serves as a placeholder for either $|0\rangle$ or $|1\rangle$. If the second or the third qubit of the target state is in a state of $|1\rangle$, the first oracle call does not have any effect. Similarity, the second oracle calls only has an effect, if the desired target state follows the schema $|?10\rangle$, since the Y gate acts on the second qubit by flipping it from $|0\rangle$ to $|1\rangle$ while adding a global phase value of $-i$. Following the second oracle call, a superposition over all possible states is created through the use of multiple Hadamard gates.

In contrast to circuits in Fig. 4 and Fig. 5, the circuit of Fig. 6 makes use of only a single oracle call. It starts by creating an equal superposition of all possible states using Hadamard gates, similar to the state of the art [10]. Of special interest is the unitary matrix of the gates following the oracle call in Fig. 6:

$$Unitary_{CircuitFig.6} = \begin{pmatrix} -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 & -0.25 & -0.25 & -0.25 & -0.25 \\ 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 & -0.25 & -0.25 & -0.25 & -0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & -0.75 \end{pmatrix}$$

Aside from the sign of row 2 and 4, this matrix is exactly equal to the diffusion operator described by Lov Grover in [10]. Since no operations take place after the gates composing this operator have been applied, the signs of its rows do not have an effect on the observed probabilities. Without any knowledge of or limitations in regard to the gate set used by the state of the art solution, the genetic algorithm was able to construct a complex operator similar to the state of the art in both functionality and form.

5 Main Findings

In regards to the main contributions of this paper, we found that:

- The results from both test studies demonstrate that the proposed fitness functions result in faster and more effective convergence on average, compared to the baseline fitness function from [19]. This suggests that incorporating a measure of quantum advantage enhances the genetic programming approach, enabling the generation of more effective quantum circuits within fewer generations.
- Each experimental configuration successfully produced circuits for the Bernstein-Vazirani Problem that are similar in structure and performance to state-of-the-art solutions. The differences were minimal, with variations typically involving only a single gate, while the overall structure remained consistent with established methods.
- In the more complex Unstructured Database Search Problem, the best-performing circuits exhibited greater variability in both structure and gate selection. Notably, two of the top circuits shared similarities in their entanglement strategies and oracle call usage, offering an effective alternative to the state-of-the-art approach by utilizing a step-wise entanglement procedure.
- Among the circuits generated for the Unstructured Database Search Problem, the one illustrated in Fig. 6 closely mirrors the diffusion operator described by [10]. This circuit effectively redistributes probability amplitudes towards the marked state, functioning similarly to Grover's diffusion operator. Remarkably, the genetic algorithm independently generated a circuit that aligns closely with the state-of-the-art solution in both performance and interpretability, despite lacking prior knowledge of the existing solution.

6 Conclusion

We proposed two novel fitness functions, each taking a different route to incorporate a measure of quantum advantage in the genetic algorithm. When evaluated on the Bernstein-Vazirani Problem and the Unstructured Database Problem, the suggested fitness functions outperformed the baseline fitness function of Spector et al. [19] in terms of convergence speed and the quality of the produced circuits. While the experiments run in the context of this paper have been able to closely reproduce the state of the art solution on the Bernstein-Vazirani Problem, they yielded interesting new circuit architectures on the Unstructured Database Problem. In one of the experiments, a circuit was learned that closely resembles the diffusion operator suggested by Lov Grover in [10]. These findings affirm the overall potential of genetic algorithms for the discovery and generation of meaningful new quantum algorithms and circuit architectures.

In future work, we plan to incorporate a measure of well-formedness for the probability amplitude distributions generated by each quantum circuit. Currently, all oracle cases are treated independently. However, since quantum oracles are designed as black boxes, quantum algorithms should exhibit consistent behavior across different oracle cases and implementations. If the probability amplitude distributions produced by a circuit vary significantly across different oracle implementations, it suggests that the circuit may be overfitting to specific test cases rather than addressing the underlying abstract problem. Exploring how to integrate a well-formedness measure into the fitness function of the genetic algorithm, and assessing whether this adjustment impacts the quality and convergence of the algorithm, represents a promising direction for future research.

References

[1] Usama Ahsan et al. 2020. AutoQP: Genetic Programming for Quantum Programming. In *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. IEEE, 378–382.

[2] Andrei Bautu and Elena Bautu. 2007. Quantum circuit design by means of genetic programming. *Romanian Physics* 52, 5-7 (2007), 697–704.

[3] Ethan Bernstein and Umesh Vazirani. 1993. Quantum complexity theory. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*. 11–20.

[4] Don Coppersmith. 2002. An approximate Fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067* (2002).

[5] Kalyanmoy Deb and Himanshu Jain. 2013. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE transactions on evolutionary computation* 18, 4 (2013), 577–601.

[6] David Deutsch. 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400, 1818 (1985), 97–117.

[7] Lukas Franken, Bogdan Georgiev, Sascha Mucke, Moritz Wolter, Raoul Heese, Christian Bauckhage, and Nico Piatkowski. 2022. Quantum Circuit Evolution on NISQ Devices. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[8] Felix Günther Gemeinhardt, Stefan Klikovits, and Manuel Wimmer. 2023. Hybrid multi-objective genetic programming for parameterized quantum operator discovery. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 795–798.

[9] Alessandro Giovagnoli, Volker Tresp, Yunpu Ma, and Matthias Schubert. 2023. Qneat: Natural evolution of variational quantum circuit architecture. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 647–650.

[10] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.

[11] En-Jui Kuo, Yao-Lung L Fang, and Samuel Yen-Chi Chen. 2021. Quantum architecture search via deep reinforcement learning. *arXiv preprint arXiv:2104.07715* (2021).

[12] Paul Massey, John A Clark, and Susan Stepney. 2004. Evolving quantum circuits and programs through genetic programming. In *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part II*. Springer, 569–580.

[13] Paul Massey, John A Clark, and Susan Stepney. 2006. Human-competitive evolution of quantum computing artefacts by genetic programming. *Evolutionary Computation* 14, 1 (2006), 21–40.

[14] Keri A McKiernan, Erik Davis, M Sohaib Alam, and Chad Rigetti. 2019. Automated quantum programming via reinforcement learning for combinatorial optimization. *arXiv preprint arXiv:1908.08054* (2019).

[15] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature communications* 5, 1 (2014), 4213.

[16] Benjamin IP Rubinstein. 2001. Evolving quantum circuits using genetic programming. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, Vol. 1. IEEE, 144–151.

[17] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

[18] Peter W Shor. 2003. Why haven't more quantum algorithms been found? *Journal of the ACM (JACM)* 50, 1 (2003), 87–90.

[19] Lee Spector, Howard Barnum, Herbert J Bernstein, and Nikhil Swamy. 1998. Genetic programming for quantum computers. *Genetic Programming* (1998), 365–373.

[20] Daniel Tandeitnik and Thiago Guerreiro. 2022. Evolving quantum circuits. *arXiv preprint arXiv:2210.05058* (2022).

[21] Sagar Uprety, Dimitris Gkoumas, and Dawei Song. 2020. A survey of quantum theory inspired approaches to information retrieval. *ACM Computing Surveys (CSUR)* 53, 5 (2020), 1–39.

[22] Colin P Williams and Alexander G Gray. 1998. Automated design of quantum circuits. In *NASA International Conference on Quantum Computing and Quantum Communications*. Springer, 113–125.

Received [TO BE FILLED]; revised [TO BE FILLED]; accepted [TO BE FILLED]