Multi-Objective Synthesis of Quantum Circuits Using Genetic Programming

Moein Sarvaghad-Moghaddam[1], Philipp Niemann[2,3], and Rolf Drechsler[2,3]

[1] Young Researchers and Elite Club, Mashhad Branch,
Islamic Azad University, Mashhad, Iran
[2] Group of Computer Architecture, University of Bremen, Bremen, Germany
[3] Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
moeinsarvaghad@mshdiau.ac.ir {pniemann,drechsler}@uni-bremen.de

Abstract. With the emergence of more and more powerful quantum computers, synthesis of quantum circuits that realize a given quantum functionality on those devices has become an important research topic. As quantum algorithms often contain a substantial Boolean component, many synthesis approaches focus on reversible circuits. While some of these methods can be applied on rather large functions, they often yield circuits that are far from being optimal. Aiming at better solutions, evolutionary algorithms can be used as possible alternatives to above methods. However, while previous work in this area clearly demonstrated the potential of this direction, it often focuses on a single optimization objective and employs cost functions that are not very well suited for quantum-technological implementations of the resulting circuits.
In this paper, we propose a framework for multi-objective synthesis of quantum circuits based on Genetic Programming that puts a focus on quantum-specific aspects and can be tuned towards several relevant/related cost metrics. A preliminary evaluation indicates that the proposed approach is competitive to previous ones. In some cases, the generated circuits even improve over existing results on all optimization objectives simultaneously, even though the latter were found by specifically targeting a single objective.

1 Introduction and Related Work

Quantum computing [10] is resulted by combining quantum mechanics and classical information theory which can lead to powerful (and something strange) effects like superposition or phase shifts that can be exploited for asymptotically faster algorithms for several important problems. To actually conduct a complex quantum algorithm on a quantum computer it has to be synthesized from a high-level description to a quantum circuit composed of elementary quantum gates that are supported by the specific device. As many quantum algorithms contain a substantial Boolean component, many synthesis approaches focus on so-called reversible circuits that realize these components in a quantum-compatible way.

In fact, the design and optimization of reversible and quantum circuits is a hard problem, due to the high dimension of the search space (the functionality

is given in terms of exponentially large *transformation matrices*), the large number of possible gates and so on. For synthesis of reversible and quantum circuits, various methods have been presented including matrix decomposition methods [13], search-based methods using graph theory (DFS and BFS) [4], cycle-based approaches [12], methods based on decision diagrams [16] or two-level representations (e.g., Exclusive Sum of Products, ESOP [9]) etc. The synthesis of reversible logic circuits using above methods are not optimal in terms of commonly applied cost metrics like quantum cost (QC) or gate count (GC) as they stick into local minima. An optimal synthesis method for reversible circuits has been proposed by Shende et al. [14]. However, it only works for small circuits and fails to provide optimal solutions for larger circuits.

For achieving good solutions for larger circuits in reasonable computation time, evolutionary algorithms can be used as possible alternatives to above methods. In these methods, using evolutionary algorithms and especially Genetic Algorithms (GA), the goal has been defined as achieving the desired transformation matrix fully or with an acceptable percentage of difference. In [7,11], a simple GA has been used to design quantum circuits. Although these methods search the large space of the solutions in the problem, they are very general and do not consider the cost of the circuit. Other approaches use alternative optimization techniques like Particle Swarm Optimization (PSO, see e.g. [3]) or Ant Colony Optimization (ACO, see e.g. [6]). Recently, Abubakar et al. [1] presented a synthesis method based on Genetic Programming (GP) a subfield of evolutionary computing in which computer programs evolve to solve the studied problem. While the approach yields quite appealing results, it is strongly focused on and limited to reversible circuits.

In this work, we also use GP, but in contrast to [1], our used circuit model is completely different and we put a focus on quantum-specific aspects like the consideration of equivalence up to global-phase as well as cost metrics that are more suited for currently considered quantum technologies. Another innovation of this paper is the use of a two-step fitness function that in the first step evaluates the accuracy of the circuit, before the cost of the circuit is considered.

The rest of this paper is structured as follows: in Section 2, some background information about quantum computing and genetic programming is given. The proposed method is then presented in Section 3. Section 4, discusses some preliminary results, before Section 6 concludes the paper.

2 Background

2.1 Quantum Computation and Circuits

Quantum computation is based on qubits, i.e. two-level quantum systems whose state $|\phi\rangle$ can be described as a superposition of the basis states $|0\rangle$ and $1\rangle$: $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ for complex-valued *amplitudes* α, β with $|\alpha|^2 + |\beta|^2 = 1$.

Any quantum operation can be represented by a unitary transformation matrix U, i.e. a complex-valued matrix of dimension $2^n \times 2^n$ where n denotes the

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & e^{\pi i/4} \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \frac{1+i}{2}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) H (b) T (c) NOT (d) V (e) SWAP

Fig. 1: Basic Quantum Operations

number of qubits in the considered quantum system. Commonly used basic operations (termed as quantum gates) are shown in Fig. 1. Note that all these gates operate on a single target qubit (except for the SWAP gate which operates on two qubits). In order to enable more powerful computations, all gates can also be used in a controlled fashion, i.e. they are only applied to the target qubit(s) if a set of control qubits is in a predefined state ($|1\rangle$ for positive controls or $|0\rangle$ for negative controls).

A cascade of such operations/gates $G = g_1 \dots g_d$ forms a quantum circuit. The corresponding transformation matrix is computed as the matrix product of the matrices of the individual gates (in reversed order). Two quantum operations whose matrices only differ by a scalar factor $e^{i\phi}$ (i.e. a global phase shift by ϕ) cannot be distinguished physically and are, thus, considered equivalent.

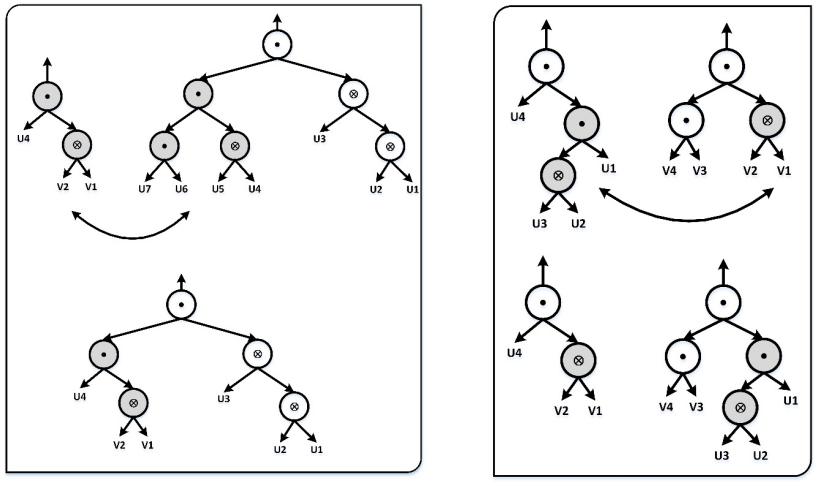
2.2 Genetic Programming

Genetic Programming (GP) [5] plays the role of an evolutionary algorithm distinctively functioning on a varying-sized chromosome, generally in a tree structure. The populations of computer programs are genetically developed through the Darwinian tenet of natural choice and hereditary processes. Individuals in a GP population act as programs in a hierarchical tree structure, comprised of primitives such as functions and terminals defined in the problem field.

GP commences by means of a preliminary population of programs randomly created in most cases. Each individual in this population is, therefore, assessed via a predefined problem-specific fitness function. The fitness value signifies the competence of the individual to resolve the problem. Selection utilizes the fitness value in order to recognize the individuals which will replicate and pair off to yield the following generation. Mutation and crossover simulate the recombination process. These operators intend to decompose the features of parent individuals to breed distinctive offspring individuals. To this end, as illustrated in Fig. 2 crossover swaps sub-trees between the parents' chromosome, while mutation randomly replaces a sub-tree in the parent's chromosome. The creative process is reiterated up to when a concluding circumstance is fulfilled.

3 Proposed Approach

In this section, we present detailed information about the proposed multi-objective synthesis method for quantum circuit synthesis based on Genetic Programming.



(a) Mutation (b) Crossover

Fig. 2: Genetic operators.

3.1 Genetic Programming Specifics

In order to apply GP for quantum circuit synthesis, we require a way to represent circuits as chromosomes. However, as the operations (gates) in quantum circuits can have multiple inputs, the standard representation as a (binary) tree can hardly be employed here. In contrast to [1], where the internal nodes represent multiplication and all leaves correspond to individual gates, we rather use degenerated trees where each node represents an individual gate and has a single successor (such that the tree depth is one less than the tree size).

In each round of the algorithm, a new population (child population) is generated from the previous population (parent population). As in [1], the individuals that serve as parents are selected based on the roulette wheel method.[4] Consequently, the expected number of children for each individual is based on its rank in the population [2], i.e. an individual with high fitness will have more children. As genetic operations, we employ

- *mutation*, i.e., the first k gates of a (single) parent circuit are taken and a randomly generated circuit is appended, and
- *crossover*, i.e., two parent circuits are split into two parts and the first part of one circuit is combined with the second part of the other—yielding two offsprings.

Moreover, the concept of Dynamic Maximum Tree Depth is used [2]. In this method, a dynamic limit is applied on the depth of the trees allowed in the

[4] This method acts as if a roulette with random pointers is spun, and each individual owns a portion of the roulette which corresponds to its expected number of children.

population. At first, i.e. when generating the initial population, it is set to a low value but increased whenever needed to accommodate a new best individual. For the generation of the initial population we use the Ramped Half-and-Half method, i.e. half of the trees have the full maximum depth while the remaining trees have a (randomly chosen) smaller depth.

3.2 Proposed Fitness Function

In order to survey the fitness of a chromosome, i.e. its similarity with the given unitary matrix to be synthesized, the following two-step fitness function is used. At first, the similarity of the matrix U^c of each chromosome C with the target unitary matrix S (of dimension $2^n \times 2^n$) is calculated as follows:

$$Fitness_1(C) = penalty \cdot \sum\nolimits_{i,j=1}^{2^n} |U^c_{ij} - S_{ij}| \tag{1}$$

Also, in this step, a penalty value is used to put more significance on the accuracy of the solutions. Of course, Eqn. (1) does not consider global-phase equivalence between the target function and the obtained circuit. In order to evaluate this, we compute

$$correctness(C) = |\operatorname{tr}(S^\dagger U^c)| \cdot 2^{-n} \tag{2}$$

where tr denotes the trace operator and $S^\dagger$ denotes the conjugate transpose, i.e. the multiplicative inverse, of S. If S and U^c differ only by a complex phase factor, we have $S^\dagger U = \phi I$ for some complex number ϕ with $|\phi| = 1$. As a result, $|\operatorname{tr}(S^\dagger U^c)| = |2^n \phi|$ such that Eq. (2) will evaluate to 1 in this case.

Thus, if Eqn. (1) is equal to zero or Eqn. (2) equals to one, then the chromosome realizes the desired functionality and in the next step other optimization criteria quantum cost, circuit depth and nearest neighbor cost are considered:

$$Fitness_2(C) = 1 + k_1 \cdot QuantumCost(C) + k_2 \cdot Gates(C) + k_3 \cdot NNCost(C) \tag{3}$$

Here, the coefficients k_1, k_2, and k_3 can have arbitrary values between zero and one according to the significance of the respective cost metric. While quantum cost (QC) were originally computed based on realizations in terms of the NCV gate library (consisting of NOT, controlled-NOT and controlled-V gates) and are only suited for reversible circuits, nowadays other metrics have become more relevant for quantum circuits, e.g. T-count or T-depth. These are based on the assumption that the circuits are realized in the fault-tolerant Clifford+T library where the high cost of T gates dominates the overall execution cost. In contrast, the pure gate count (GC)—apparently inspired by conventional circuit realizations—does not have much significance for quantum circuits as the execution time of individual gates can differ significantly. Nearest neighbor cost reflect the fact that multi-qubit operations (e.g. a controlled NOT) can typically only be applied on adjacent physical qubits. Such topological constraints (so-called *nearest neighbor constraints*) apply essentially to all currently investigated technologies for quantum computation.

Table 1: Experimental Evaluation

Benchmark	Method [1]		Best Result from [8]		Proposed Approach	
	GC	QC	GC	QC	GC	QC
3_17	5	11	6	12	3	10
4_49	12	28	14	28	9	26
hwb5	24	102	38	80	30	80
nth_prime4_inc	11	26	14	26	8	24
nth_prime5_inc	28	96	29	91	24	82
4b15g_3	14	33	15	33	14	33

4 Preliminary Evaluation

In order to evaluate the principal capabilities of the proposed method, it has been implemented using MATLAB software and the GPLAB toolbox created for genetic programming by Sara Silva [15]. In order to have a baseline for comparison, we applied our method to all benchmarks from [1] (ranging from three to five qubits) and also used the same cost metrics w.r.t. quantum cost. For the coefficients k_1, k_2, k_3 in Eq. (3) we used 0.8, 0.6, and 0.4, respectively, in order to put the main focus on quantum cost while also taking into account gate count and nearest neighbor costs in a reasonable way. We started with a population size of 50 individuals and successively increased the population size up to 500 individuals if no satisfying solution was found.

Due to page limitations, only a small selection of the results is shown in Table 1 and compared to the corresponding results from [1] and Maslov's benchmark library [8], but we obtained very similar results for all benchmarks. Note that, in order to allow for a fair comparison, a post-processing has been applied to identify certain groups of 2-controlled NOT gates and 1-controlled NOT gates which can be identified as a single gate (the so-called Peres gates) and, thus, lead to small savings in gate count and quantum cost. The results clearly indicate that the proposed method is able to compete with previous work. In some cases, the generated circuits even improve over existing results on all optimization objectives simultaneously, even though the latter were found by specifically targeting a single objective. More precisely, the numbers listed for GC and QC for method [1] as well as [8] might refer to different circuits, while the costs listed for our approach are always realized by a single circuit.

5 Conclusion

In this paper, we proposed a multi-objective synthesis method for quantum circuits based on Genetic Programming. In contrast to previous work, we put a strong focus on quantum-related aspects like global-phase equivalence and more appropriate cost metrics that allow to incorporate technological constraints like nearest neighbor constraints already during synthesis. Another innovation of the

method is the use of two-step fitness function that in the first step, the accuracy of the circuit is evaluated. Then, metrics of quantum cost, circuit depth, nearest neighbor costs are considered. A preliminary evaluation confirmed that the proposed method is competitive to previous methods when applied in their original domain of reversible circuits. For future work, we plan to thoroughly investigate the method's performance and benefits for real quantum benchmarks (e.g. in terms of Clifford+T circuits).

References

1. Abubakar, M.Y., Jung, L.T., Zakaria, N., Younes, A., Abdel-Aty, A.: Reversible circuit synthesis by genetic programming using dynamic gate libraries. Quantum Information Processing **16**(6), 160 (2017)
2. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Intl' Conf. on Genetic Algorithms. pp. 101–111 (1985)
3. Datta, K., Sengupta, I., Rahaman, H.: Particle swarm optimization based reversible circuit synthesis using mixed control toffoli gates. J. Low Power Electronics **9**(3), 363–372 (2013)
4. Kole, D.K., Rahaman, H., Das, D.K., Bhattacharya, B.B.: Optimal reversible logic circuit synthesis based on a hybrid dfs-bfs technique. In: Intl' Symposium on Electronic System Design. pp. 208–212 (Dec 2010)
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
6. Li, M., Zheng, Y., Hsiao, M.S., Huang, C.: Reversible logic synthesis through ant colony optimization. In: DATE. pp. 307–310 (2010)
7. Lukac, M., Perkowski, M., Goi, H., Pivtoraiko, M., Yu, C.H., Chung, K., Jeech, H., Kim, B.G., Kim, Y.D.: Evolutionary approach to quantum and reversible circuits synthesis. Artificial Intelligence Review **20**(3), 361–417 (Dec 2003)
8. Maslov, D.: Reversible logic synthesis benchmarks page (2018), `http://webhome.cs.uvic.ca/~dmaslov/`
9. Mishchenko, A., Perkowski, M.A.: Logic synthesis of reversible wave cascades. In: IWLS. pp. 197–202 (2002)
10. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge Univ. Press (2000)
11. Ruican, C., Udrescu, M., Prodan, L., Vladutiu, M.: Automatic synthesis for quantum circuits using genetic algorithms. In: ICANNGA. pp. 174–183 (2007)
12. Saeedi, M., Zamani, M.S., Sedighi, M., Sasanian, Z.: Synthesis of reversible circuit using cycle-based approach. J. Emerg. Technol. Comput. Syst. **6**(4) (2010)
13. Shende, V.V., Bullock, S.S., Markov, I.L.: Synthesis of quantum-logic circuits. IEEE Trans. on CAD of Integrated Circuits and Systems **25**(6), 1000–1010 (2006)
14. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. on CAD of Integrated Circuits and Systems **22**(6), 710–722 (2003)
15. Silva, S., Almeida, J.: Gplab-a genetic programming toolbox for matlab. In: In Proc. of the Nordic MATLAB Conference (NMC-2003. pp. 273–278 (2005)
16. Wille, R., Drechsler, R.: Bdd-based synthesis of reversible logic for large functions. In: DAC. pp. 270–275 (2009)