Evolving Quantum Circuits to Implement Stochastic and Deterministic Cellular Automata Rules

Shailendra Bhandari, Sebastian Overskott, Ioannis Adamopoulos, Pedro Lind, Sergiy Denysov and Stefano Nichele

March 28, 2022

Evolving quantum circuits to implement stochastic and deterministic cellular automata rules

Shailendra Bhandari[1,2,3], Sebastian Overskott[1,2,3], Ioannis Adamopoulos[1,2,3], Pedro G. Lind[1,2,3], Sergiy Denysov[1,3], and Stefano Nichele[1,2,3,4,5]

[1] Department of Computer Science, OsloMet – Oslo Metropolitan University, P.O. Box 4 St. Olavs plass, N-0130 Oslo, Norway
[2] *AI Lab* – OsloMet Artificial Intelligence Lab, Pilestredet 52, N-0166 Oslo, Norway
[3] *NordSTAR* – Nordic Center for Sustainable and Trustworthy AI Research, Pilestredet 52, N-0166 Oslo, Norway
[4] Department of Holistic Systems, Simula Metropolitan Center for Digital Engineering, Pilestredet 52, N-0166 Oslo, Norway
[5] Department of Computer Science and Communication, Østfold University College, B R A Veien 4, N-1757 Halden, Norway

Abstract. The aim of this work is to generate specific rules of deterministic and stochastic cellular automata (CA) using the set of five quantum gates, which is known to generate any quantum circuit. To build such quantum circuits, we use an evolutionary algorithm, based in mutations, which allows the optimization of quantum gate types and their connectivity. The fitness function of the evolutionary algorithm aims at minimizing the difference between the output of the quantum circuit and the CA rule. We also inspect the differences observed when changing the number of gates and the mutation rate. We benchmark our methods with stochastic as well as deterministic CA rules, and briefly discuss the possible extensions their quantum "cousins" may enable.

Keywords: Quantum circuits · Critical behavior · Evolutionary algorithms · Stochastic Cellular Automata

1 Scope and motivation

Quantum computing and cellular automata are two important topics in modern computer science, often approached independently from each other. Cellular automata (CA) are classical discrete models for reproducing complex processes of extended systems of coupled elements, based in simple rules which map the present state of each element and its direct neighbours into the next state of each element. Quantum computing (QC) is a field in its infancy aiming at solving the limitations of classical computation, when dealing with problems which in practice are not computable due to their complexity.

Underlying the difference between classical and quantum algorithms are their respective elementary components, the bit and qubit respectively. While bits can have only two possible states, 0 or 1, qubits can have an infinite number of possible states *if* their state is not measured. If it is, they will "collapse" to the classical pair of possibilities, 0 or 1, and one can know beforehand the probability to observe each one of these final measured

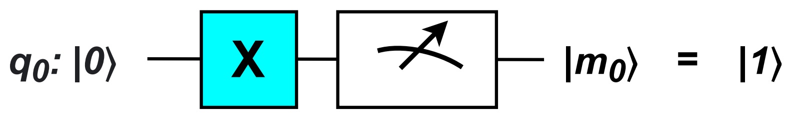

Fig. 1. Example of a quantum circuit [9]

states. The gates used to build quantum circuits are also different from the usual classical gates (e.g. NOT, AND, OR), since they can operate on qubits, and only after measurement, retrieve an on- or an off-state with a specific probability. Consequently, they may be good candidates for mimicking so-called stochastic cellular automata, discrete models which define the cellular automaton rules with an associated probability. In this paper we introduce a framework to build specific quantum circuits, whose classical counterpart reproduces the behavior of a CA rule. We illustrate the framework for some specific deterministic and stochastic CA rules and argue that it can be, in principle, apply to any other rule.

Moreover, due to its particular features, beyond the common behavior of classic bits and digital gates, there is still no straightforward procedure to build the quantum circuit for one specific computation or algorithm using a minimum number of gates. In this paper we show how evolutionary algorithms can be useful to address this drawback to build optimal quantum circuits. Evolutionary algorithms (EA) are a group of heuristic algorithms that solves problems in a Darwinian way. In short, they start out with a widespread population of possible solution for a problem. The solutions in the population that yielded best results when tested against a target is used to create a new population: the next generation of solution. This process continues until a solution meets the target criteria, based in a so-called fitness function (FF) which plays the role of a cost function. We will focus on a sub-category of EA called genetic algorithms (GA) [2], which built upon the idea of the solution of the algorithm to be chromosomes, compose of genes [3]. In our case the chromosome represents a quantum circuit while a gene represents a quantum gate.

Finally, the framework also enables to explore new contexts of complex behavior. In particular, we focus in critical behavior. We introduce a quantum circuit, which when reduced to a purely classical architecture - with all its components being measured and therefore without superposing their states - retrieves the usual behavior of a critical CA. Having such circuit, and with the sufficiently large number of qubits, one is able to explore the outcome of an evolving critical CA when "switching-off" the full monitoring of its qubits and leaving their states to superpose in time.

A quantum circuit involves a collection of one or more qubits which are initialized to state $|0\rangle$. Figure 1 shows an example of a quantum circuit involving one qubit. The gate which acts on the qubit is represented by a square and in our case, it is the X gate which flips the state of the qubit from 0 to 1. Each horizontal application of gates for a qubit is called a wire and each wire has a depth which is the number of gates applied on that qubit. In our example we have one wire with depth 1. The measurement symbol at

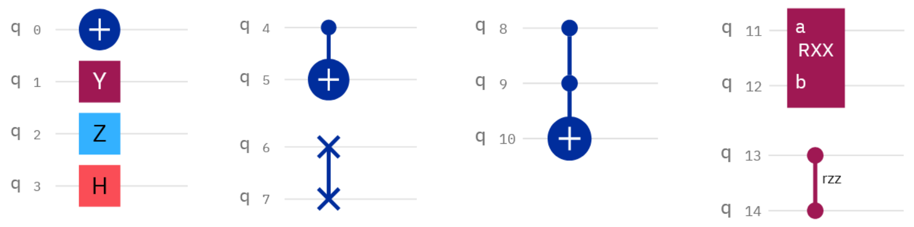

Fig. 2. A overview of quantum gates available to the algorithm: X-gate (q0), Y-gate (q2), Z-gate (q3), CNOT (control: q4, target: q5), swap-gate (q6, q7), Toffoli-gate (control: q8 and q9; target:q10), RXX-gate (q11, q12), RZZ-gate (q13, q14) [1].

the end is not a gate. It is the component collapsing the state of the qubit into either 0 or 1. The time progress in a quantum circuit goes from the left to the right [9].

2 The genetic algorithm to evolve quantum circuits

GA have been used to evolve quantum circuits with success earlier [5, 7, 11]. Lukac and Perkowski uses the unitary matrix representation of the gates and the identity matrix for connections/wires. This way we can use Kronecker products (tensor products) and matrix products to calculate the complete circuit as a matrix. This also makes it possible to represent the FF as a matrix to calculate the error. As we see in [5] and [2] the quantum gates are genes in the algorithm. They never changes the properties of the gates themselves, but move, swap, delete gates from the circuit as genetic operators. It is also a possibility to add new gates as a mutation. The gates has been encoded in several ways. Yabuki and Iba [11] encodes the gates with a letter set of four digits 0,1,2,3, and assign each gate a three letter codon i.e. 231. Here the first letter describes type of gate, and the second and third indicate what qubit will be operated. By using a table we can look up the gate type and placement in the circuit. Here we will focus in one particular set of genes (quantum gates) and compare the results with two different fitness functions. We will use the following set of quantum gates: Hadamard, three Pauli gates (X, Y and Z), Cnot, Toffoli, swap, RZZ, and RXX.

In order to address our optimization problem, the performance of the proposed approach has been accessed by running the evolutionary algorithm on a simulator belonging to the IBM Q experience initiative (Qiskit)[6]. For more information about the module and instructions on how to use it, please visit the Github repository https://github.com/Overskott/Quevo. The project was created and done in python version 3.8 with Qiskit version 0.34.1 and scipy.special 1.7.1.

The code developed for this paper was done in python, and resulted in a python module called quantum_circuit_evolver. The module consists of three classes: `Chromosome`, `Generation` and `Circuit`. The `Chromosome`-class is responsible for handling the series of integers by storing them as a list. The class also handles the creation of random series, the list of angles needed by some of the gates, mutation of the series, and other list

[6] https://qiskit.org

Neighbors	Stoch. CA Prob.	Rule90	Rule110	Rand. Prob. 1	Rand. Prob. 2	Rand. Prob. 3
[0, 0, 0]	0.394221	0	0	0.6364	0.4778	0.1988
[0, 0, 1]	0.094721	1	1	0.6603	0.5604	0.4701
[0, 1, 0]	0.239492	0	1	0.5261	0.8528	0.9836
[0, 1, 1]	0.408455	1	1	0.1748	0.4818	0.7115
[1, 0, 0]	0	1	0	0.8820	0.3143	0.6616
[1, 0, 1]	0.730203	0	1	0.3371	0.3464	0.1218
[1, 1, 0]	0.915034	1	1	0.0340	0.0678	0.1328
[1, 1, 1]	1	0	0	0.4444	0.9124	0.7306

Table 1. The deterministic and stochastic CAs considered in this paper. For the stochastic cases, the values indicate the probability of an update of value 1 for the middle cells in the triad-neighborhood. For the deterministic cases, the valus indicate the exact update imposed.

related functions. It takes a list of the desired gate types as a parameter on construction, and automatically creates the tables needed for parsing. The `Generation`-class stored a generation of chromosomes, the fitness associated with each, methods for running and retrieving fitness for two different fitness functions, and functions for printing. Last, the `Circuit`-class is handling the parsing from string and generating a Qiskit quantum circuit, the simulations of the circuit, measurements, and visualisation of the circuit.

At the beginning, the certain number of chromosomes is assigned with the number of quantum gates per circuit and subsequently, the initial number of chromosomes are generated. The evolutionary iteration starts and proceeds by identifying the best fit chromosomes, evolving the current chromosomes into a new one and inserting this best as the first element of the new population. This evolution cycle continues depending on the desired number of generations, and every time finds the best chromosome in the generation to be a parent for the next generation. When a new parent is generated at each evolution step, every chromosome undergoes a very simple mutation process so as to get the best mutated chromosomes. Mutation can happens either by replacing a gates from the pool of gates in the chromosome with a randomly generated new one, or replacing the chromosome so as to generate four best parents. The mutation process is probabilistic and when the chromosomes get mutated, the four best chromosomes are left unchanged as "elites" the rest of the chromosomes are evolved with a probabilistic selection where each of the current circuit has a probability to become a parent that is proportional to its fitness. We consider one-dimensional CAs composed of cells with two possible states, 0 or 1 (Boolean CA), which are updated according to one out of 256 possible rules matching each one of the eight neighborhoods ([0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0] and [1, 1, 1]) to an update of the middle cell. Table 1 shows the matching in all cases considered in this paper.

To assess the performance of the quantum circuits we derive, we measure only the first qubit (q0) and consider two different fitness functions to compare the measured probability Q(ω) of an initial state ω, computed from the chromosome, and the corresponding desired probability $P(\omega)$. The first one is the sum of the absolute difference

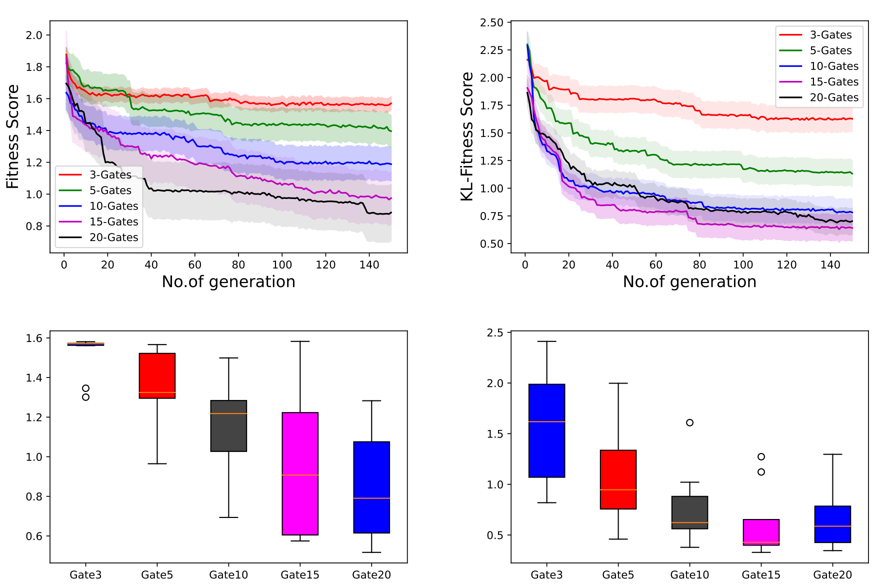

Fig. 3. (Top) The fitness scores as a function of the number of generations, for **different number of gates**. (Bottom) Number of gates vs. the best fitness scores. In each case we show the result for (left) the absolute sum of differences, Eq. (1), and (right) Kullback-Leibler divergence, Eq. (2). The fitness scores of each gates for the box plots are the best fitness scores per run and the fitness scores for the lower two plots are the average fitness scores of 10 runs.

between each P(ω) and corresponding Q(ω):

$$F = \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)| \,. \tag{1}$$

This gives a possible fitness value between 0 and 8. The second type of fitness function that is implemented is the Kullback-Leibler (KL) divergence, which measures the difference between two probability distributions and has been used in other works [6] [4] as a fitness function as well:

$$D_{KL}(P||Q) = \sum_{\omega \in \Omega} P(\omega) log\Big(\frac{P(\omega)}{Q(\omega)}\Big) \,. \tag{2}$$

In our case those distributions are discrete and each one of them has eight different values which are related to the eight different initial states of the three qubits. If the distributions completely match then both F and D_{KL} fitness functions are zero.

3 Three different "flavours" of quantum cellular automata

We will consider three different types of quantum cellular automata. We will start with a stochastic *critical* cellular automaton and then illustrate the robustness of our framework

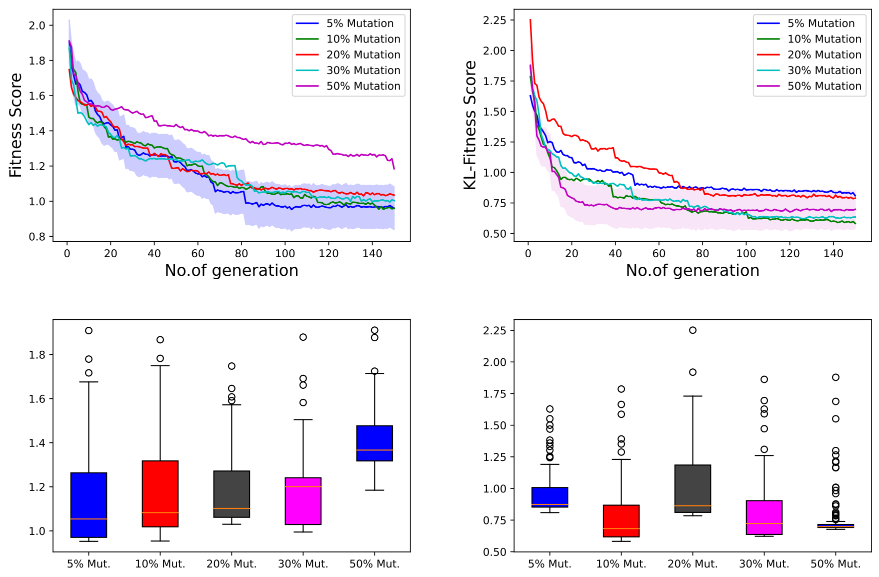

Fig. 4. (Top) The fitness scores as a function of the number of generations, for **different mutation rates**. (Bottom) Number of gates vs. the best fitness scores. In each case we show the result for (left) the sum of absolute differences (Eq. (1)), and (right) the KL divergence.

with more general stochastic CA (random updates) and a few deterministic rules, namely rule 90 and 110 [10]. In all cases, the updates are shown in Tab. 1. We fix a maximum number of chromossomes $N_c = 20$, a maximum number of generations $N_g = 150$, and each simulation is repeated for $N_{ic} = 10$ initial conditions randomly chosen.

3.1 The quantum cousin of a stochastic critical CA

The authors in [8] have evolved a stochastic cellular automata model in order to reach criticality which is a property of dynamical systems that gives them the possibility to do robust computations. For each triad-pattern a probability has been calculated through genetic algorithm for the central cell to have state 1. These probabilities are shown in Table [1], second column.

We start by considering the number of gates used, evolving quantum CAs with 3, 5, 10, 15 and 20 gates. The mutation probability is fixed to 10 percent.

Figure 3 shows the result for both fitness functions above. It is clear from the figure that the fitness score improves with increasing the number of gates until 15 gates and the gradual increase is seen for 20 gates. Therefore for experiment 2 and 3, the number of gates used is 15. Notice that the fitness score is optimum for 15 gates in the case of KL fitness function while the fitness score is optimum for 20 gates in the absolute difference of probabilities fitness function. Moreover, while the sum of absolute

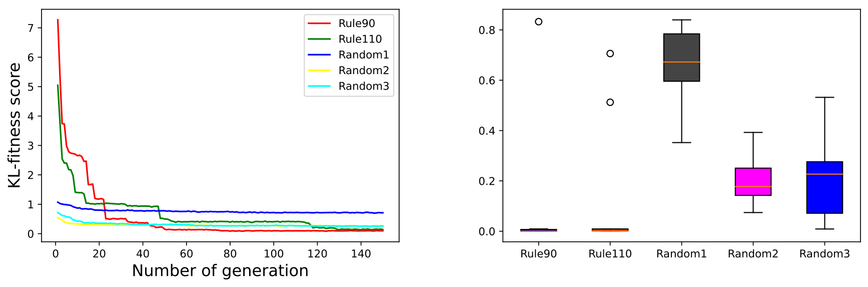

Fig. 5. (Left) Fitness scores for different sets of probabilities against the number of generations for KL-fitness function. (Right) Best fitness scores per runs for KL-fitness function for different sets of probabilities. The fitness scores of each gates are the average fitness scores of 10 runs.

differences performs better for the cases with lowest number of gates, while KL fitness function is better suited when the number of gates increases.

Next we explore how the fitness changes when changing the mutation probability. The goal is to test the impact of mutation over the fitness function at different number of generations. We fix the number of gates to 15 and select mutation probabilities of 5%, 10%, 20%, 30% and 50%.

Figure 4 (left) shows the comparison of the fitness scores with the number of generations at different percentage of mutation rate for the absolute difference of probabilities fitness function, while Fig. 4 (right) shows the comparison of the fitness scores with the number of generations at different percentage of mutation rate for KL-fitness function. In case of KL divergence fitness function we can see the gradual improvement in results with increase in percentage of the mutation rate, however similar conclusion cannot be drawn in other fitness score as the results are random.

3.2 The two other flavours: deterministic CA and stochastic CA but non-critical

We end our investigation applying the same framework to deterministic rules as well as to stochastic CAs which do not show critical behavior. Here, we fix the parameters with 15 gates and 10% mutation probability. The desired probabilities for Rule 90, and Rule 110 (the deterministic rule, so the probabilities will of the 8 neighborhoods will be either 0 or 1), and 8 randomly generated probabilities with 3 repetition are used as target probabilities. The target probabilities for each condition are shown in Tab. 1 and results are shown in Fig. 5. The fitness scores for the deterministic CA (Rule 90 and Rule 110) is very good as shown in figure Fig. 5 (left), therefore it is quite successful to run quantum circuit for deterministic CA. For the stochastic CA with randomly generated probabilities with three repetition, the results are indeed promising, with best fitness scores 0.14 and 0.27 for random2 and random 3. While the best fitness scores for random 1 is 0.685 which is not bad either but needs further experiments to be able to tune the evolution to reduce the difference even further. Fig. 5 (right), shows the fitness scores for the different sets of probabilities against the number of generation for two deterministic

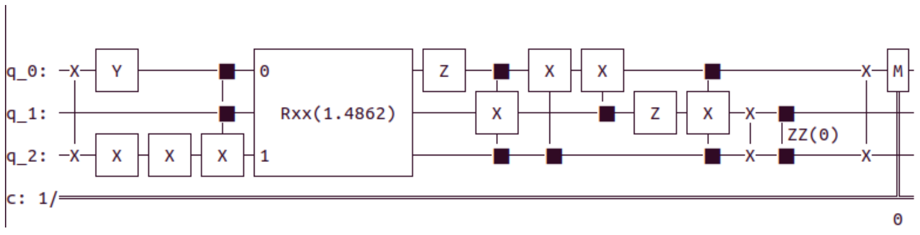

Fig. 6. Visualization of a circuit created by 15 number of gates, 10% mutation probability with KL fitness scores. The circuit is the best produced circuit with fitness score of 0.3404 for critical stochastic CA.

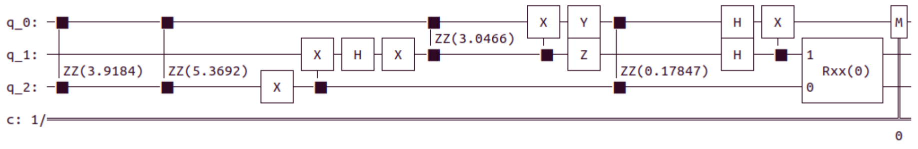

Fig. 7. Visualization of a circuit created by 15 number of gates, 10% mutation probability with KL fitness scores for deterministic CA (Rule 110). The circuit is the best produced circuit with fitness score 0.000921.

CA (Rule 90 and Rule 110) and the stochastic CA with randomly generated probabilities with three repetition (Random1, Random2, and Random3).

Finally, we illustrate the quantum circuits generated with our algorithms. The circuit in Fig. 6 is the result of a run that scored much better than the average runs for critical stochastic CA. It shows a fitness score of 0.3404, using KL fitness function in Eq. (2). Similarly, the circuit in Fig. 7, is the result of the deterministic CA Rule 90. It has a best fitness score of 0.000921, with 15 numbers of gates, 10% mutation probability and the KL fitness function.

4 Discussion and conclusions

In this paper we showed a simple framework to derive quantum circuits reproducing specific CA rules, using genetic algorithms. We showed that the framework is able to evolve quantum circuits towards several types of CA rules, ranging from deterministic rules to stochastic updates.

An important observation for all the experiments that we did, was that most of the times the fitness functions of the parents that would survive were not the same in the next generations. Their fitness functions in the next generations were sometimes actually worse than those they had in the previous generations. Mutated solutions with better fitness functions would then take their places as parents but still these functions could be worse than those that the first parents had originally. As a result, we did not see a gradually decrease of the value of the best fitness function from generation to generation as we expected or as we wanted but instead, that value had ups and downs. The above phenomenon happened for both types of fitness functions that we used and the reason is the following. Every quantum circuit which corresponds to a chromosome or a solution, is executed multiple times (1000 in our case) with a simulator. The outcome probabilities that are related with the fitness function can not be exactly the same each time we run the same circuit in another generation. If we execute the same circuit another 1000 times for example, we will have slight differences in our results. Even when we increased the number of shots in the simulator (10000 and then 50000) to stabilize the probability value, we did not succeed.

Another thing that we realized, was that for every set of desired probability outcomes, the value of the best fitness function was achieved sooner or later either we applied the mutation rules with different mutation probabilities each time, or we just chose random possible solutions for the same number of generations. As we can see in Fig. 4, this randomly happened very early, when using the sum of absolute differences as fitness function, Eq. (1), while for the KL fitness function, Eq. (2), it does not happen in the same way. One possible reason for this is that the pool of possible solutions is not very big based on the number of gates that we used, although the type of gates that we had were enough to build almost any quantum circuit. As far as concerned the two different fitness functions, we can not really say that one performs better than the other. One small observation though is that the diagrams which corresponds to the KL fitness function have more gentle fluctuations and they reach their final range of values in the early generations.

Moreover, there are some parameters that could be further tested in future investigations, as well as considering additional genetic operators in evolutionary algorithms, e.g. crossover.

References

1. Operations glossary. https://quantum-computing.ibm.com/composer/docs/iqx/operations_glossary, accessed: 2022-3-22
2. Giraldi, G.A., Portugal, R., Thess, R.N.: Genetic algorithms and quantum computation. CoRR **cs.NE/0403003** (2004), http://arxiv.org/abs/cs/0403003
3. Holland, J.H.: Genetic algorithms. Scholarpedia **7**(12), 1482 (2012). https://doi.org/10.4249/scholarpedia.1482, revision #128222
4. Lucas, S.M., Volz, V.: Tile pattern kl-divergence for analysing and evolving game levels. Proceedings of the Genetic and Evolutionary Computation Conference (Jul 2019). https://doi.org/10.1145/3321707.3321781, http://dx.doi.org/10.1145/3321707.3321781
5. Lukac, M., Perkowski, M.: Evolving quantum circuits using genetic algorithm. In: Proceedings 2002 NASA/DoD Conference on Evolvable Hardware. pp. 177–185 (2002). https://doi.org/10.1109/EH.2002.1029883
6. Martín, F., Moreno, L., Garrido, S., Blanco, D.: Kullback-leibler divergence-based differential evolution markov chain filter for global localization of mobile robots. Sensors **15**(9), 23431–23458 (2015). https://doi.org/10.3390/s150923431, https://www.mdpi.com/1424-8220/15/9/23431
7. Mukherjee, D., Chakrabarti, A., Bhattacharjee, D., Choudhury, A.: Synthesis of quantum circuits using genetic algorithm. FULL PAPER International Journal of Recent Trends in Engineering **2** (12 2009)
8. Pontes-Filho, S., Lind, P., Yazidi, A., Zhang, J., Hammer, H., Mello, G.B., Sandvig, I., Tufte, G., Nichele, S.: A neuro-inspired general framework for the evolution of stochastic

dynamical systems: Cellular automata, random Boolean networks and echo state networks towards criticality. Cognitive Neurodynamics **14**(5), 657–674 (2020). https://doi.org/10.1007/s11571-020-09600-x, https://doi.org/10.1007/s11571-020-09600-x
9. Sutor, R.S.: Dancing with Qubits. Packt Publishing (2019)
10. Wolfram, S.: Cellular automata as models of complexity. Nature (London) **311**(5985), 419–424 (1984)
11. Yabuki, T.Y.: Genetic algorithms for quantum circuit design –evolving a simpler teleportation circuit–. In: In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference. pp. 421–425. Morgan Kauffman Publishers (2000)