Project 2

Evolving Fitness function for QC synthesis

GOAL: Using GA evolve a fitness function specific to the quantum circuit synthesis to be used in another GA.

The use of GA or any automated methods for logic synthesis (binary, multivalued, fuzzy, quantum, etc) requires an evaluation function representing the fitness calculation. In our approach we are using GA to evolve quantum circuits and especially we want to evolve the cheapest possible implementation of bigger universal gates such as Toffoli, Fredkin, Kerntopf, de Vos and others using only 1 and 2 qbit gates The reason for this search is the fact in quantum technologies the biggest possible unitary gate can have maximum of two wires (see slides). Consequently evolving circuits/gates from one and 2 wire primitives is very important for future application for quantum computing such as building quantum computers working with permutation logic.

During the synthesis there are various criteria we try to minimize. These are especially the error, the cost of the circuit and the size. The error is the measure defining the difference between the matrix representing the circuit and the final circuit, by comparing each elements of both matrices. The cost and the size are evaluated together. In general we are using the following cost variants:

a. 1 wire gate has a cost of 0
b. 2 wire gate has a cost of 1
c. and no gate with more than 2 wires are used.

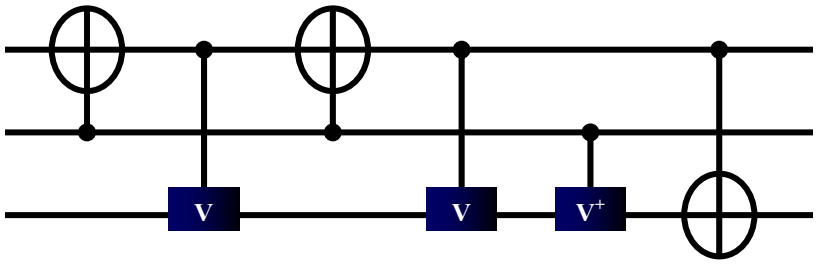
The cost function is the total sum of costs of all gates in the circuit. Then for the normalization the minimal cost that was found for this gate is divided by $Cost_{this}$. For example the cost function for Fredkin gate can be:

$$Cost = 5/Cost_{this} \quad (1)$$

with $Cost_{this}$ is the cost of the currently evaluated circuit and 5 is the above mentioned minimal cost for this circuit. Currently in our approach we are using this composite fitness function:

$$F = \alpha(1 - \frac{Error}{Max\_error}) + \beta\frac{1}{Cost} \quad (2)$$

with F is the fitness, α and β are scaling factors for respectively the error and the cost part, Cost is the cost of the circuit and in general the 1/Cost is replaced by an expression similar to (1). The left-hand part of the equation is the error evaluation and Error/Max_error scales the error in the interval [0,1] so that consequently the error is inversely proportional to the fitness. The problem of this fitness function appears when one analyzes closer a single circuit evaluation. This means that according to our method we are creating circuits by assembling parallel blocks each of them containing one or more quantum gates in parallel as shown on the figure below.



0.7	**0.39**	**0.33**	**0.76**	**0.18**	**1**
0.7	**0.4**	**0.4**	**0.8**	**0.3**	**0**
1	**2**	**3**	**4**	**5**	**6**

Fitness

Error

Cost

This means that the circuit based on the Cnot from the first block has a cost of 1 and an error of 0.7. This is only an example so the values might not be correct. Consequently it is obvious from this illustration that the synthesis of an arbitrary quantum circuit will be subject to very high variations of the fitness function making the search for the optimal solution very hard.

The presented here fitness function was our early and consequently does not reflect the complexity of the problem space we are dealing with. The task in this project is to use a GA to evolve a better fitness function. To do this you will be provided with software and its documentation. Your task is to modify the GA so as able to encode your desired function and encode it. A good starting point is to take the proposed Fitness function and to evolve parameters such as α and β. The fitness evaluation will be based upon the calculation of the fitness value for a set of given circuits. We advise to use your imagination and encode more different functions and evolve them and finally compare which one fits the best. An example is to evolve a fitness function based on an exponential form:

$$F = \exp^{(-error-\cos t_{this})}$$

This can be achieved through an encoding of different functions as specific expression of the genotype.

The criterion for evaluation a fitness function:

- Statistically the function will have to increase the fitness function during evaluation per segments of a circuit well designed.
- Consequently you will have to test each time your function over the provided set of gates and assigning fitness to it according to its result.

As already mentioned the evaluation will be done over a set of provided gates. It means that for each evolved fitness function you will have to evaluate all circuits segment by segment and assign the fitness according to the linearity of these values. For example the fitness function from the table will have a pretty low fitness because it is fluctuating inside the circuit. The ideal fitness function should have for one circuit values such as 0, 0.2, 0.4, 07, 09, 1. Of course this is only an example but the goal is to have a function hiding the nonlinearities in the circuit and representing only the steps toward a good solution

Expected results:

- Report what functions you have been evolving and the reasons for
- Graphical representation of data in the for of how many circuits are well described by each evolved function
- Discussion on how such a function should be
- Discussion on you results