Evolving Quantum Circuits with Genetic Algorithms and through Exhaustive space search
Outline

- Technology
  - Constraints
- Design of new quantum primitives
- Evolutionary and Frame based Generator
- Exhaustive Search
- Results
- Future directions
- Possible projects - Homework
Technology for Quantum Computing
What size of (binary) Quantum Computers can be build in year 2003?

- 7 bits
What size of (binary) Quantum Computers can be build in year 2003?

- **7 bits**

- dicarboxylcyclopentadienyl (perfluorobutadien-2-yl) iron ($C_{11}H_5F_5O_2Fe$)
- Each fluorine and 2 Carbones can be used for computation
5 qubit 215 Hz Q. Processor

(Vandersypen, Steffen, Breyta Yannoni, Cleve, and Chuang, 2000)

- The Molecule

- Quantum Circuit

$T_2 > 0.3 \text{ sec} ; \sim 200 \text{ gates}$

Source: IBM, Hot Chips Conference, 2000
Qbits, ¬ bits

• In binary quantum logic, the notation for the superposition is $\alpha|0\rangle + \beta|1\rangle$.

• These intermediate states cannot be distinguished, rather a measurement will yield that the qubit is in one of the basis states, $|0\rangle$, or $|1\rangle$.

• The probability that a measurement of a qudit yields state $|0\rangle$ is $|\alpha|^2$, and the probability is $|\beta|^2$ for state $|1\rangle$. The sum of these probabilities is one. The absolute values are required, since in general, $\alpha$ and $\beta$ are complex quantities.
Quantum Circuit Synthesis is Technology dependent
Constraints based on the properties of physical implementation of quantum circuits (technology constraints, gate costs)

We would like to assume that any two quantum wires can interact, but we are limited by the realization (layout) constraints.

Structure of atomic bonds in the molecule determines neighborhoods in the circuit.

This is similar to restricted routing in FPGA layout - link between logic and layout synthesis known from CMOS design now appears in quantum.
Decoherence plays an important role → minimization circuit length

Minimization of cascade’s width but each bit counts (more critical than in reversible synthesis)

At first, we will be interested only in the so-called “permutation circuits” - their unitary quantum matrices are permutation matrices

One solution to layout constraint problem in quantum NMR computers is to take into account in logic synthesis phase only those gate and their placements that are technology-realizable
• Even if conceptually we use higher complexity gates, ultimately we have to build from 2-qubit gates.

• Another possibility is to assume only primitives for the future algorithms are only 2-qubit gates then the optimal circuit will be the shortest.

• Bottom line is that basic gate in quantum logic is a 2*2 (2-qubit gate).

• 3*3 Toffoli, Fredkin, de Vos, Kerntopf, Margolus are not directly realizable as a primitive.
Molecule - Driven Layout and Logic Synthesis

Allowed gate neighborhood for 2 q-bit gates
Quantum wires A and C are not neighbors.

This is a result of our ESOP minimizer program, but this is not realizable in NMR for the above molecule, because there is no connection between A and C, for instance, in the molecule.
So we have to modify the schematics as follows:

```
a b c d

 Costs 3
 Feynman gates
```
So we have to modify the schematics as follows:
Design of new (complex) quantum gates and their costs
Design a Toffoli Gate from 2-qbit quantum primitives

\[
\begin{align*}
V & \quad V^+ \quad \text{are root square of NOT and its hermitian (complex) conjugate such as } V^*V = \text{NOT} \\
& \\
V : \quad \frac{(1+i)}{2} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix} \\
C_V: \quad \text{q-bit 2 unchanged unless q-bit 1 equals to 1}
\end{align*}
\]
Example: Optimal Solution to Miller Function

\[(AC \oplus BC \oplus AB) \oplus (A \oplus B) = AC' \oplus AB \oplus BC'\]

\[(AC \oplus BC \oplus AB) \oplus (A \oplus C) = AC \oplus AB' \oplus B'C\]

Cost = 1 Toffoli + 4 Feynman gates

Cost in Gates: 
\[4 \times 1 + 5 = 9\]
2-qubit quantum realization of Miller Gate

Cost in Gates: \( 9 \times 1 = 9 \)

Cost in Gates: \( 7 \times 1 = 7 \)

Cost in Gates: \( 7 \times 1 = 7 \)
Fredkin Gate build from Toffoli and Feynman gates

Cost in Gates: $2 + 5 = 7$

$b \oplus c \oplus ab \oplus a'c = ac \oplus ba'$

$c \oplus a(b \oplus c) = c \oplus ab \oplus ac = ca' \oplus ac$
Transforms

Cost in Gates: \( 7 \times 1 = 7 \)

Cost in Gates: \( 5 \times 1 = 5 \)
Evolutionary and Frame-based gate generators
Genetic Algorithm

- **A set of elements being modified according to evolutionary rules:**
  - Selection (based on the fitness function)
  - Crossing Over
  - Mutation
  - Replication

- **These operators are made in generation steps**

- **Process stops when the solution is found**

- **Important in GA**
  - **Encoding of the elements/individuals**
    - Complex with a lot of parameters
    - Simple, task specific no parameters
  - **Fitness function**
    - Simple
    - Including layout specific constraints
    - Cost of gates
Circuit Encoding

- **Kronecker product** ($\otimes$)
- **Matrix product** ($\otimes$)

Circuit matrix representation

- $\oplus$ - Kronecker product
- $\otimes$ - Matrix product

4 / PWCCNOT / P / PFHH / P / PFF / P
GA for quantum circuit synthesis

- Set of elements: randomly generated q-circuits encoded in string representation

5PWSWWPPPHWCPPPWSWWWP
Example

Pauli X gate

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

Hadamard gate

\[
\frac{1}{\sqrt{2}}
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\]

XOR or CNOT

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

Wire

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

\[|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}\]

\[|1\rangle \rightarrow (|0\rangle - |1\rangle)/\sqrt{2}\]

\[|j\rangle|k\rangle \rightarrow |j\rangle|j \oplus k\rangle\]
Evaluation
Calculation

\[
\begin{pmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 \\
\end{pmatrix}
\times
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
=
\begin{pmatrix}
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & -1 & 0 & 0 \\
0 & 0 & 1 & -1 \\
\end{pmatrix}
\]

\[
\frac{1}{\sqrt{2}}
\begin{pmatrix}
1 & 1 \\
1 & -1 \\
\end{pmatrix}
\oplus
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
=
\frac{1}{\sqrt{2}}
\begin{pmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 \\
\end{pmatrix}
\]
Operations

Mutation

2 /P Cn /P /P H X /P /P Y W /P 2

2 /P W X /P /P H X /P /P Y W /P 2

Crossing over

2 /P X Y /P /P X Y /P /P /P

2 /P Cn /P /P H X /P /P Y W /P 2

2 /P X Y /P /P H X /P /P Y W /P 2
## Overview

<table>
<thead>
<tr>
<th>Mutation</th>
<th>Gates Blocks</th>
<th>Position (block/circuit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Over*</td>
<td>Segments</td>
<td>Experimental (unitary matrices)</td>
</tr>
<tr>
<td>Reproduction</td>
<td>Circuits</td>
<td>Best gates Best Circuits</td>
</tr>
</tbody>
</table>

* - for circuits having only same number of I/O
GA’s settings

- SUS, Roulette wheel
- Fitness:

\[
F_i = \frac{1}{1 + error_i} - \Lambda_i
\]

\[
error = \sum_{i=1}^{n} \sum_{j=1}^{2^n} |U_{ij} - S_{ij}| \quad S, U \in U(2^n)
\]

Goal:

Fitness:

\[
\sqrt[4]{2} \begin{bmatrix}
0 & 0 & 1 & 1 \\
0 & 0 & 1 & -1 \\
1 & 1 & 0 & 0 \\
-1 & 1 & 0 & 0
\end{bmatrix}
\]

\[
\sqrt[4]{2} \begin{bmatrix}
0 & 0 & 1 & 1 \\
0 & 0 & 1 & -1 \\
1 & 1 & 0 & 0 \\
1 & -1 & 0 & 0
\end{bmatrix}
\]

\[
\Lambda_i = 0
\]

\[
F_i = \frac{1}{1 + 2/N_i} \approx 0.88
\]
Frame-based search starting from Peres gate

- **Peres gate** - the **cheapest** 3-qubit gate

Adding Feynman gates on all possible pairs of wires on which Feynman is realizable

\[ a \oplus b \oplus ab \oplus c = (a+b) \oplus c \]

\[ C = 0 \Rightarrow (A+b) \]

\[ C = 1 \Rightarrow (a+b)' = a'b' \]
Other frame search examples

\[ A = a \oplus c \]
\[ B = a \oplus b \]
\[ C = ab \oplus ac \oplus bc \]

Cost in Gates: 5*1 = 5

\[ A = a \oplus c \]
\[ B = a'b \oplus a'c \oplus bc \]
\[ C = ab \oplus ac \oplus bc \]

Cost in Gates: 5*1 = 5
Exhaustive Search
Searching all gates in a very limited space of permutation $10^{15}$-$10^{18}$

- Up to 7 segments circuits
- 3 I/O circuits
- Comparing to gates such as Toffoli, Fredkin, de Vos, Kerntopf, etc.
Idea: to look for all possible equivalent gates in a certain category

Using specified gates in different technologies

Find the minimal possible cost of the gate

Exhaustive gate search
Results
Unitary gate search examples

No starting set restriction

\[
\begin{bmatrix}
1 & 1 \\
1 & -1 \\
\end{bmatrix}
\]
Generations: 10
Mutation rate: 0.3

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]
Generations: 20
Mutation rate: 0.3
Other gates search

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Generations: 100
Mutation rate: 0.4
Random circuit search

EPR producing circuit
Generations: 450
Mutation: 0.3

4 /PWWHW/P /PWWF/P /PWSW/P /PWFW/P /PSWW/P /PFWW/P /PSWW/P /PWSW/P
Random circuit search

"Send" circuit

Generations: 150
Mutation: 0.3

3 /PWWH/P /PWF/P /PFW/P /PHWW/P
Examples for Toffoli
## Experimental results

<table>
<thead>
<tr>
<th>Number of inputs per q-gate</th>
<th>Number of generations</th>
<th>pM</th>
<th>pC</th>
<th>Real time (average 20 runs)</th>
<th>pM &lt; 0.2 Number of generations</th>
<th>Real time (average 20 runs)</th>
<th>Population size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - input</td>
<td>&lt;50</td>
<td>0.4</td>
<td>0.6</td>
<td>&lt; 30 seconds</td>
<td>&lt;100</td>
<td>&lt; 1 minute</td>
<td>50</td>
</tr>
<tr>
<td>2 - inputs</td>
<td>&lt;50</td>
<td>0.6</td>
<td>0.4</td>
<td>&lt; 30 seconds</td>
<td>&lt;100</td>
<td>&lt; 1 minute</td>
<td>50</td>
</tr>
<tr>
<td>3 - inputs</td>
<td>50 - 200</td>
<td>0.6</td>
<td>0.6</td>
<td>&lt;1 minute</td>
<td>&lt;200</td>
<td>&lt; 3 minutes</td>
<td>60</td>
</tr>
</tbody>
</table>
## Experimental results (cont.)

<table>
<thead>
<tr>
<th>Problem searched</th>
<th>Solution found F1/F6</th>
<th>Time of search in number of generations F1/F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toffoli</td>
<td>YES/YES</td>
<td>&lt;2000 / &lt;50000</td>
</tr>
<tr>
<td>Fredkin</td>
<td>YES/YES</td>
<td>&lt;1000 / &lt;75000</td>
</tr>
<tr>
<td>Margolus</td>
<td>YES/NO</td>
<td>&lt;1000 / &lt;100000</td>
</tr>
</tbody>
</table>
Generalized Feynman Gate

\[ f_1 \]

A \rightarrow f_1 \rightarrow B \oplus Q \rightarrow P
Generalized Ternary 3*3 Toffoli Gate
Generalized 4*4 Fredkin Gate
Generalized 4*4 Kerntopf Gate
\( A_1 \oplus f_{n-1}(A_1, \ldots, A_{n-1}) \)
A_{n-1} f_{n-1} (A_1, \ldots, A_{n-1}) + A_n f_{n-1} (A_1, \ldots, A_{n-1})

\begin{align*}
\text{Generalized n*n Fredkin Gate} \\
A_1 \\
A_{n-2} \\
A_{n-1} \\
A_n
\end{align*}
Future Perspectives
Population → Crossover → Mutation → Evaluation → Replication → Population

New Generation → Population

Genotype: I.e Polarity of GRM

Phenotype: Logic expression

Evaluation → Circuit

Darwinian evolution
Lamarckian optimization

- An alternative approach to this problem can be the use of Lamarckian approach to the GA.
  - When a solution is found, the genotype is modified in order to be more precise for a given term-wise polarity set and the given function.
  - Consequently the search for this individual will induce smallest search space
Baldwinian learning

- Genotype
  - (polarity₁, fitness₁)
  - (polarityᵣ, fitnessᵣ)

- Heuristics
- Phenotype
  - GRM₁,₁
  - GRM₁,n
  - GRMᵣ,₁
  - GRMᵣ,n

- Min Cost
- Min Cost

- Learning Polarity
- Learning Product terms
Projects and Homework
General directions

• All results from a GA should be averaged over 20 runs
• All details, files, precisions can be either asked directly or by email on lukac242@netscape.net
• Exceptionally good or bad results should be denoted apart
• All steps during exploration and experimentation should be precisely written down in a log book
Statistical analysis of non linearity in synthesized circuits

- During synthesis fitness of circuits is highly non linear and non proportional to the distance from the final gate
- Goal: analyze a set of known gates (provided) and make a statistical analysis on the changes of the fitness function of the gates
- Finally establishing a table of results where the known gates will be represented as curves of fitness function
Statistical analysis of non linearity in synthesized circuits

Example

<table>
<thead>
<tr>
<th>0.7</th>
<th>0.39</th>
<th>0.33</th>
<th>0.76</th>
<th>0.18</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>0.4</td>
<td>0.4</td>
<td>0.8</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

- Fitness
- Error
- Cost
Inversely to the classical approach the goal is to synthesize a set of parameters fitting on the non-linearity present in the fitness function evaluating quantum circuits.

Parameters evolved can be either taken from already existing fitness function or a completely new fitness function can be evolved.
Evolving Fitness function for QC synthesis

- Example

\[ f = \alpha \left( \frac{1}{\text{error}} \right) \Pi \beta(\text{Cost}) \]

- Circuit

- Fitness

- Encode parameters \( \alpha, \beta, \Pi \) into chromosome. \( \Pi \) is arbitrary function such as +, -, *, /, etc.

- Evolve them according to your own selected strategy

- You can choose any fitness function you want but minimum 2 parameters
We want to test how will a GA with Pareto optimal evaluation evolve new quantum circuits.

Minimal parameters are the size of the circuit and the error as a measure of the distance from the goal. More parameters can be used as cost, complexity, etc.

Use ranking method to select the best individuals to the next generation.
Evolving circuit according to more than one parameter.

Ranking is a method allowing to evaluate circuits on the numbers of won tournament.

Comparing all parameters one to one:

\[ \text{if } \sum_i (X_0 - X_j) \geq i \quad \text{Rank}_0 +, \quad \forall X = [x_0, ..., x_i] \]

\[ \text{else Rank}_0 \]

Fitness assigning on the number of won fights.

All individuals are compared to all other in the population.

Mutation, Crossover as in standard GA.

Project 3
Exploring GA for QC by modifying parameters and settings

- By modifying parameters of the GA for QC synthesis it is possible to modify the behavior of the algorithm.
- Here the goal is to explore in a methodic way combinations of parameters in the GA and observe the different behavior (deception, premature convergence).
- Parameters to be modified can be all such as mutation or crossover probability, population size, selection criteria, etc.
Exploring GA for QC by modifying parameters and settings

- Parameters to modify
  - Probabilities
    - Mutation
    - Crossover
    - Selection threshold
  - Selection pressure
    - Elitism
    - Roulette wheel
    - Universal Stochastic Sampling
    - Tournament
    - Threshold
  - Mixed generations
    - Comparing children with parents
  - Genetic operators
    - Mutation
      - Normal, bitwise, inversion,
    - Crossover
      - Normal, Double, Multi-point, Multi-parent

- Best n for replication

- Population using Elitism

- Population using SSU

- Individuals with Fitness > \( \lambda \)

- Population using Tournament

- Population using Threshold
Exhaustive search of Q-circuits

- Works by enumerating all possibilities of gates
  - Permutations
  - Wire allocations
  - Def: number of wires, number of blocks

- Goal: Make an extensive research for specified gates using this software.

- Classification according to different criteria:
  - Distance from the goal
  - Complexity
  - Cost
Using GAq

- A console interface
  - Input: file
  - Output: file
- All commands through the input file
- Available Monday on the class Web Page
  - Download the zipped package
    - Contents: exe file, source file, documentation and user’s manual

!!!!GOOD LUCK!!!!