# Use of Simulated Annealing in Quantum Circuit Synthesis

Manoj Rajagopalan

17 Jun 2002

## Outline

- **Overview**
- Simulated Annealing: the idea used
- Implementation
- Results
- Conclusions and Future Work

## Quantum Circuit Synthesis

- Input-output transformation specified
- Objective: Gates and their arrangement into a circuit to achieve this transformation
- Methods:
  - Factorization <vshende> (Cybenko)
  - Enumeration <gmathew, akprasad> (exhaustive, B&B)
  - Genetic Algorithms <smaddipa> (Williams+, Yabuki+)
  - Simulated Annealing <rmanoj> ( )

## Synthesis by SA

- Choose whole circuit every time
- Use equivalent transformations (optimization)
- Incremental modification (ends of circuit)
  - Computationally efficient!
  - But is it good enough?
- Hey, what is Simulated Annealing?

## Optimization Problems

- State variables:
  - Reflect the system state
  - May not be directly modified
- Control variables (degrees of freedom):
  - Affect the state
  - Directly modified
- Constraints on state and control (equality/inequality)
- Objective: Solve for optimum of scalar objective function

## Optimization Heuristics

- Search space is too large for exhaustive enumeration, too complex to visualize
- Pick random solutions and evaluate performance index (PI)
- Filter good/best solution(s)
- Perturb these and re-evaluate PI
- Incorporate means to avoid local minima

## Simulated Annealing: Basic Idea

- Objective: minimize scalar function subject to given constraints
- Select one initial solution and evaluate cost
- Perturb the solution and calculate new cost
- Improvement in cost?
  - Yes: Copy perturbed solution to initial solution
  - No: Probabilistically accept perturbed solution (to avoid local minima)

## Quantum Circuit Synthesis as an Optimization Problem

- Select type, number and location of gates (control)
- Evaluate equivalent unitary operator (state)
- Constraint: Operator == given unitary
- Objective: Minimize number of gates

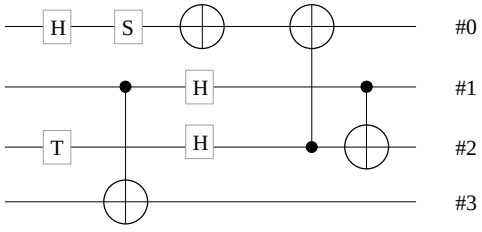
## Outline

- Overview
- **Simulated Annealing: the idea used**
  - **Incremental perturbation**
- Implementation
- Results
- Conclusions and Future Work

## SA: Quantum Circuit Synthesis

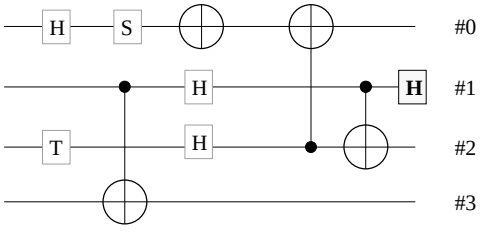
- Assume given operator can be synthesized
- Number of qubits known for given operator
- Choose entire circuits in each perturbation?
  - How many gates?
  - What location?
  - Need to 'multiply' all gates to get equivalent operator each time!
- Alternative: Incremental modification (at ends of circuit each time): NOP, ADD, REM, REP
- Qubits handled independently

## SA: Incremental Perturbation




**ADD**: Hadamard Gate to #1

## SA: Incremental Perturbation



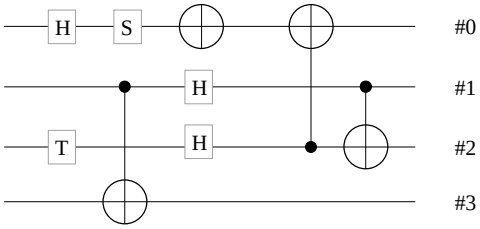

**ADD**: Hadamard Gate to #1

## SA: Incremental Perturbation

- Equivalent Operator =

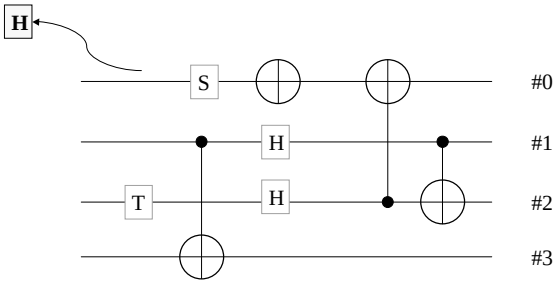
$$\begin{pmatrix} I \\ \otimes \\ H \\ \otimes \\ I \\ \otimes \\ I \end{pmatrix} \times \left[ \textit{Original} \qquad \textit{Operator} \right]$$

## SA: Incremental Perturbation




**REM**ove: Hadamard Gate from #0

## SA: Incremental Perturbation



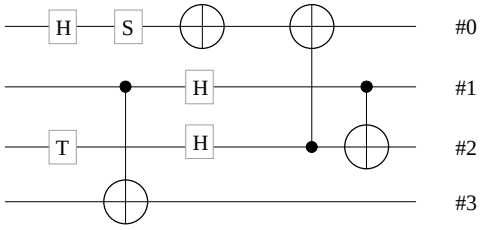

**REM**ove: Hadamard Gate from #0

## SA: Incremental Perturbation

- Equivalent Operator =

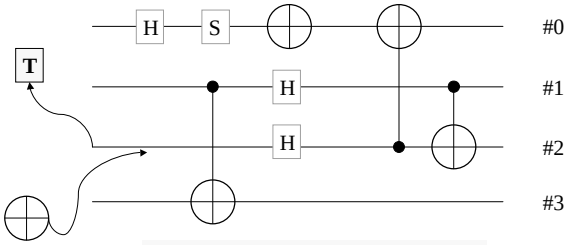
$$\left[ \textit{Original} \qquad \textit{Operator} \right] \times \begin{pmatrix} H^{\uparrow} \\ \otimes \\ I \\ \otimes \\ I \\ \otimes \\ I \end{pmatrix}$$

## SA: Incremental Perturbation




**REP**lace: T Gate on #2 with X Gate

## SA: Incremental Perturbation



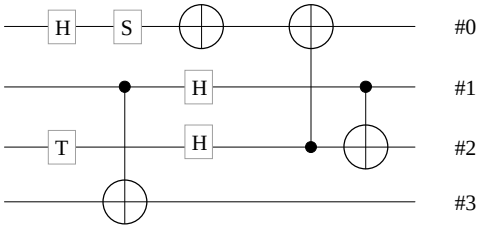

**REP**lace: T Gate on #2 with X Gate

## SA: Incremental Perturbation

- Equivalent Operator =

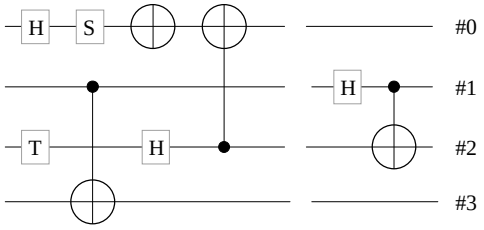
$$\left[ \textit{Original} \qquad \textit{Operator} \right] \times \begin{pmatrix} I \\ \otimes \\ I \\ \otimes \\ T^{\uparrow} \\ \otimes \\ I \end{pmatrix} \times \begin{pmatrix} I \\ \otimes \\ I \\ \otimes \\ X \\ \otimes \\ I \end{pmatrix}$$

## SA: Incremental Perturbation



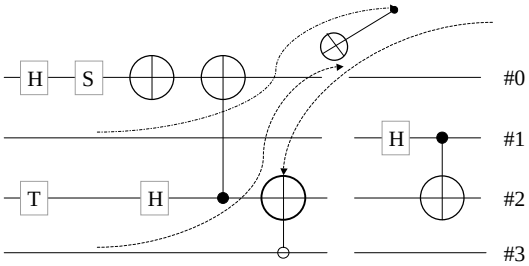

**REP**lace: CNOT on #1-3 with CNOT on #3-2

## SA: Incremental Perturbation




**REP**lace: CNOT on #1-3 with CNOT on #3-2

## SA: Incremental Perturbation




**REP**lace: CNOT on #1-3 with CNOT on #3-2

## SA: Incremental Perturbation

- Equivalent Operator =

$$\begin{pmatrix} I \\ \otimes \\ C \\ \otimes \\ NOT \\ \otimes \\ I \end{pmatrix} \times \begin{pmatrix} I \\ \otimes \\ H \\ \otimes \\ I \\ \otimes \\ I \end{pmatrix} \times \begin{pmatrix} I \\ \otimes \\ I \\ \otimes \\ NOT \\ \otimes \\ C \end{pmatrix} \times \begin{pmatrix} I \\ \otimes \\ C^{\uparrow} \\ \otimes \\ I \\ \otimes \\ NOT^{\uparrow} \end{pmatrix} \times \begin{pmatrix} I \\ \otimes \\ H^{\uparrow} \\ \otimes \\ I \\ \otimes \\ I \end{pmatrix} \times \begin{pmatrix} I \\ \otimes \\ C^{\uparrow} \\ \otimes \\ NOT^{\uparrow} \\ \otimes \\ I \end{pmatrix} \times \left[ \textit{Original} \quad \textit{Op} \right]$$

## Outline

- Overview
- Simulated Annealing: the idea used
- **Implementation**
  - **Data Structures**
  - **Algorithm**
  - **Annealer Configuration**
  - **Hardware and Software Platforms**
- Results
- Conclusions and Future Work

## Data Structures

- Class *qgate*: Quantum gates & operators
- Qubit: *list<qgate>*
- Circuit: Array of qubits
- *qgate* instance for CNOTs duplicated on control and target qubits

## SA Algorithm

- Initial circuit = empty
- Initial operator = I
- For each qubit
  - For head and tail of qubit, each
    - Choose one out of 4 moves: NOP, ADD, REM, REP in a non-conflicting manner.
    - Choose gates required for these, if applicable
- Evaluate new operator, guarding special cases
- Calculate (frobenius) norm of deviation from given unitary

## SA Algorithm

- Out of a few (10) such moves, choose move with minimum deviation norm
- Is this below tolerance ($10^{-6}$) ?
  - Yes: Synthesis complete! Return.
  - No: Is this 'cost' better than that previously accepted?
    - Yes: Accept this move into circuit
    - No: Accept this move with probability $e^{(-\Delta\, cost\, /\, T)}$

## SA Algorithm

- Repeat this procedure for a few (10) trials.
- Change temperature according to schedule.
- Iterate whole procedure till temperature lower limit is reached.

## Annealer Configuration

- Moves:
  - 4 equiprobable changes at each end of qubit: NOP, ADD, REM, REP
- 1 or 10 moves per trial
- 1 or 10 trials per iteration
- 1001 iterations:
  - Tstart = 1
  - Tend = 0.001
  - Temperature schedule = linear with step 0.001
- Objective: Simply minimize deviation norm (no circuit size reduction yet) (tolerance = $10^{-6}$)

## Platforms

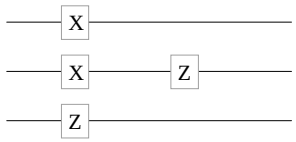
- proton.eecs.umich.edu
- AMD Athlon @ 1194 MHz 256kB cache
- Debian linux (kernel v2.4.18)
- Coded in C++
- g++ 2.95.4 with –O3 optimization
- Timing and peak memory tracking using `getrusage()`

## Outline

- Overview
- Simulated Annealing: the idea used
- Implementation
- **Results**
  - **Single qubit circuits**
  - **Circuits with CNOT gates**
- Conclusions and Future Work

## Results: simple {H,X,Z} circuits

- TEST I
  - Randomly generated, 3 qubits, 30 gates
  - Optimal equivalent:



## Results(1/2): TEST I

| | Using {H, X, Z} | | Using {H, X} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 4 | 0 | 6 | 0.09 |
| Max | 22 | 1.57 | 72 | 1.95 |
| Avg | 9 | 0.33 | 26 | 0.76 |
| | **97**% success rate | | **61**% success rate | |

## Results(2/2): TEST I

| | Using {H, Z} | | Using {H, X, S} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 6 | 0 | 5 | 0.11 |
| Max | 47 | 1.84 | 23 | 1.71 |
| Avg | 10 | 0.35 | 12 | 0.66 |
| | **17**% success rate | | **54**% success rate | |

## Results: Simple {H, X, Z} circuits

- TEST II
  - Randomly generated
  - 5 qubits, 300 gates

## Results(1/3): TEST II

| | Using {H, X, Z} | | Using {H, X, S} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 6 | 0.02 | 7 | 0.06 |
| Max | 74 | 3.54 | 127 | 15.11 |
| Avg | 17 | 0.60 | 27 | 5.53 |
| | **100**% success rate | | **82**% success rate | |

## Results(2/3): TEST II

| | Using {H, Z} | | Using {H, X} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 8 | 0.06 | 10 | 0.05 |
| Max | 58 | 11.54 | 178 | 13.98 |
| Avg | 23 | 1.82 | 52 | 3.48 |
| | **99**% success rate | | **81**% success rate | |

## Results(3/3): TEST II

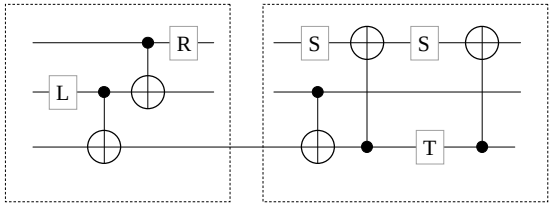
| | Using {H, S} | | Using {H, T} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 12 | 0.46 | 28 | 19.86 |
| Max | 94 | 17.71 | 28 | 19.86 |
| Avg | 29 | 4.23 | 28 | 19.86 |
| | **81**% success rate | | **1** % success rate | |

## {H, X, Z}: Conclusions

- Easily synthesized
- Optimal equivalents detected
- Average number of gates is impressive!
- Versatility of annealer: wide variety of gate libraries
- Fast!

## Results: Circuits with CNOTs

- Brassard's teleportation circuit
  - Careful with the gates (especially S and T) !



## Results(1/3): Send Circuit

| | Using {L, CNOT, R} | | Using {CNOT, H, $S_{NC}$} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 4 | 0 | 21 | 1.1 |
| Max | 213 | 99.68 | 199 | 84.17 |
| Avg | 53 | 11.23 | 70 | 23.17 |
| | **95** % success rate | | **8** % success rate | |

## Results (2/3): Send Circuit

| | Using {CNOT, H, X} | | Using {CNOT, H, Z} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 20 | 0.75 | 8 | 0.06 |
| Max | 301 | 100.38 | 357 | 107.00 |
| Avg | 125 | 31.43 | 162 | 38.46 |
| | **51** % success rate | | **45** % success rate | |

## Results(3/3): Send Circuit

| | Using {CNOT, H, X, $S_{NC}$, $T_{NC}$} | | Using {R, S, T, L, X, CNOT} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | - | - | | |
| Max | - | - | | |
| Avg | - | - | | |
| | **0** % success rate | | ? % success rate | |

## Send Circuit: Conclusions

- Difficult to synthesize with overspecified gate library
- Using 10 trials per iteration instead of the usual 1 improves chances of getting an equivalent circuit and may even detect optimal one but takes much more time and may show worse average performance

## Results(1/2): Receiver Circuit

| | Using {S, CNOT, T} | | Using {CNOT, H, $S_{NC}$} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 4 | 0 | 4 | 0 |
| Max | 20 | 0.52 | 16 | 0.72 |
| Avg | 5 | 0.05 | 6 | 0.11 |
| | **100** % success rate<br>**83**% find optimum | | **100** % success rate<br>**68**% find optimum | |

## Results(2/2): Receiver Circuit

| | Using {$S_{NC}$, CNOT, $T_{NC}$, X, H} | | Using {R, S, T, L, X, CNOT} | |
|---|---|---|---|---|
| | # gates | Time(s) | # gates | Time(s) |
| Min | 3 | 0 | | |
| Max | 16 | 1.19 | | |
| Avg | 5 | 0.22 | | |
| | **100** % success rate<br>**44**% find optimum | | ? % success rate<br>? % find optimum | |

## Receiver Circuit: Conclusions

- A lot easier to synthesize than the send circuit
- Variety of gate libraries can be used
- Overspecified gate library not a problem
- Annealer finds optimum quite often.

## Teleportation circuit: Previous Work

- Williams and Gray
  - Objective: minimize discrepancy, sum of absolute value of matrix of differences
  - Send and receive minimal circuits have 4 gates each. Achieved. 3 using N&C gates
- Yabuki and Iba
  - Receive circuit needs minimum of 3 gates. We get 4 using given library and 3 using N&C gates

## Outline

- Overview
- Simulated Annealing: the idea used
- Implementation
- Results
- **Conclusions and Future Work**

## Overall Conclusions

- Annealer is versatile over a range of discrete gate sets using a very simple configuration
- Incremental perturbation works very well (Igor rules!)

## Future Work

- Optimizations
  - Compose single qubit gates into one operator
- Ideas
  - Optimal synthesis: include # gates in PI
  - Circuits equivalent upto a nonzero global phase
  - Annealer configurations (types and probabilities of moves, temperature schedule) based on plots of quality of solution against iterations
  - More challenging problems: Toffoli gates (increased interqubit interaction) , single qubit rotations (continuous values)