University of Udine

Synthesis of CNOT minimal quantum circuits with topological constraints through ASP

Carla Piazza[1] Riccardo Romanello[1]

[1] UNIVERSITY OF UDINE, ITALY

December 10, 2023

Our Aim - Act I

- Quantum Algorithms are usually described through unitary matrices
- Unitary matrices describe the quantum gates we have to apply to physical qubits
- Only a finite set $\mathcal{B}$ of quantum gates can be manufactured
- Synthesis is the problem of expressing a generic unitary matrix in terms of $\mathcal{B}$

Our Aim - Act II

- When dealing with real world quantum computers, there are constraints to take into account
- One of them is the Qubit Topology, which restrict the set of available operations
- The specific problem we tackle is minimizing the number of CNOT gates, dealing with topological constraints.

Results

- We propose an ASP encodings to solve the CNOT minimization problem
- When solving the problem, we also take into account topological constraints
- We test the model with some random generated matrices
- We compare the results with an ASP model that does not take into account topology [1]

Plan of the Talk

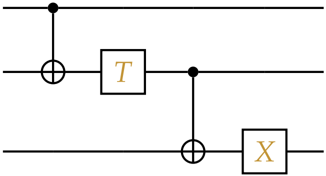
- Unitary matrices and Quantum Gates
- Clifford+T
- {CNOT, T} circuits
- Problem statement
- ASP model
- Results

Introduction to Unitary Matrices

- Quantum Computing only allows reversible operations
- Quantum states are described through vectors inside $\mathbb{C}^{2^n}$ for some n
- Such states can be manipulated only using unitary matrices
- Let $U \in \mathbb{C}^{2^n \times 2^n}$. Then U is unitary if and only if $UU^\dagger = I$

Unitaries and Gates

- In the quantum circuit formalism, qubits are manipulated through quantum gates
- Unitary matrices have a 1 to 1 correspondence to quantum gates



Universal Set of Gates

- Only a finite set of gates can be manufactured in real world quantum computers
- The rest of the unitaries must be synthesised in terms of such gates
- A set of gates $\mathcal{B}$ is called *universal* if it can synthesise any unitary U

Clifford+T

- The most adopted universal set of gates is Clifford+T
- It contains three single-qubit gates and a two-qubit gate

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix} \qquad T = \begin{pmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{pmatrix}$$

$$CNOT = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$$

- CNOT$(a, b) = (a, a \oplus b)$ where a is called *control* and b is the *target*.

CNOT minimization problem

- Consider a circuit composed only by CNOT and T gates — a {CNOT, T} circuit
- It describes some particular function g that acts on the input qubits
- The problem is to find a circuit with the same action g on the input, with the *minimum* number of CNOT gates.

Phase Polynomial Representation

Lemma 1

The action of a {CNOT, T} circuit on the initial state $|x_1, x_2, \cdots x_n\rangle$ *has the form:*

$$|x_1, x_2, \cdots x_n\rangle \mapsto e^{i\frac{\pi}{4}p(x_1,x_2,\cdots,x_n)}\,|g(x_1, x_2, \cdots x_n)\rangle$$

with $p(x_1, x_2, \cdots, x_n)$ *defined as:*

$$p(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{k} (c_i \; mod \; 8) f_i(x_1, x_2, \cdots x_n)$$

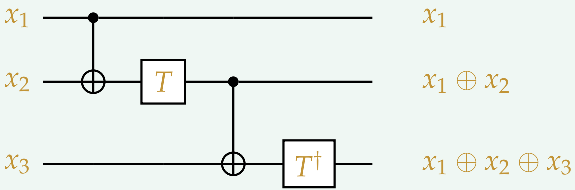
where $g : \mathbb{B}^n \to \mathbb{B}^n$ *is a linear reversible function and* p *is a linear combination of linear boolean functions* $f_i : \mathbb{B}^n \to \mathbb{B}$.

Phase Polynomial Representation

- Each circuit has its phase polynomial representation, uniquely defined by g, f_i, c_i for $i = 1, 2, \ldots k$.
- g can be written as a $n \times n$ boolean matrix G
- Each f_i can be expressed as a boolean row vector F_i

Example

Example 2



Its phase polynomial representation is the following:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad F_1 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix} \qquad F_2 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

Topological Constraints

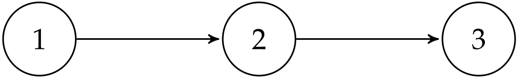
- In real world Quantum Computers, qubits are connected to each other according to some topology S
- CNOT gates can be applied only to pairs of qubits that are connected in S
- How do we encode the topology?

It's Graph Time

- We encode S as a graph (V_S, E_S)
- $V_S = 1, 2, \cdots, n$ is the set of nodes, where n is the number of qubit
- E_S is a set of directed edges
- The set E_S introduces the following constraint on the set of *legal* operations:

$$\text{CNOT}(i, j) \text{ can be applied if and only if } (i, j) \in E_S$$

Example of a Three Qubit Topology



- This examples depict a topology of a three qubit quantum device $S = \{V_S = \{1, 2, 3\}, E_S = \{(1, 2), (2, 3)\}\}$
- Only CNOT$(1, 2)$ and CNOT$(2, 3)$ are allowed
- Notice that notions like reachability becomes important when introducing this constraint
- If node i cannot reach node j in S, then CNOT(i, j) is not implementable

Problem Statement

Since the number of T gates in the input circuit is supposed to be optimal, the problem we want to solve is the following:

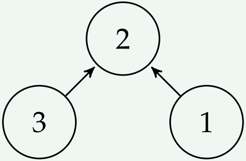
- **INPUT**: $G, S, F_1, F_2, \cdots F_k$
- **OUTPUT**: a sequence of CNOT gates to be applied such that the final behaviour of the circuit is the one described by G.

The constraints we must fulfill are the following:

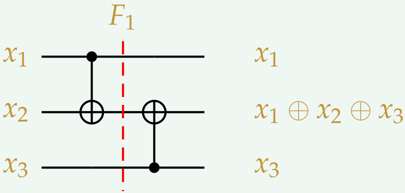
- We want to apply the minimum number of CNOT gates
- Every CNOT gates must be legal according to S
- For each F_i with $i \in \{1, 2, \cdots, k\}$, there must exist a moment during the computation in which a row of G is exactly F_i.

Example 3

Let S, G, F_1 be defined as follows:



$$G = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad F_1 = (1\,1\,0)$$



It is Solving Time

We solved the problem through Answer Set Programming. The model we propose is a DAG based one:

- We want to produce a DAG $\mathcal{G} = (V, E)$
- $V = \{x_1, x_2, \cdots, x_n\}$ where x_i will have to match the i-th row of G
- We will see in the next slides how the set E is built
- We want to build the set E of minimum size l
- Each edge in E must comply with the rules introduced by S

Solving with a DAG

- A CNOT with control x_j and target x_i is represented by a node x_i with two incoming edges:
  1. One edge comes from the closer node labelled x_i
  2. The other from the closer node labelled x_j

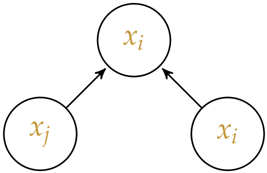


Figure 1: Generic DAG node describing CNOT(x_j, x_i).

Why is it a DAG

- The type of node induces a *layering* of the nodes
- Leafs are at layer 0
- A node at layer j can only have incoming edges from nodes in lower layers
- One layer can contain more than one node
- Let j be the current layer, it can contain at most one node labelled x_i
- Each internal node is a CNOT gate

Constraints

- After l layers, it must hold that

$$\text{VAL}_l(x_i) = G_i, \forall i \in \{1, 2, \cdots, n\}$$

Where

$$\begin{aligned}
\text{VAL}_0(x_i) &= x_i && \forall i \in \{1, 2, \cdots, n\} \\
\text{VAL}_t(x_i) &= \text{VAL}_{t-1}(x_i) && \text{if } \nexists x_k \mid (x_k, x_i, t) \in E \\
\text{VAL}_t(x_i) &= \text{VAL}_{t-1}(x_i) \oplus \text{VAL}_{t-1}(x_k) && \text{if } \exists x_k \mid (x_k, x_i, t) \in E
\end{aligned}$$

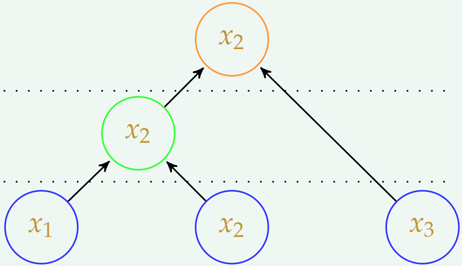
- Moreover, it must be true that:

$$\forall F_i \; \exists t \leq l \; \exists x_j \mid \text{VAL}_t(x_j) = F_i$$

DAG example

Example 4

Consider the matrix from Example 3. The generated DAG $\mathcal{G}$, with the minimum number of node is the following:



Let $G \in \{0,1\}^{n \times n}$ and $F_i \in \{0,1\}^n$ for $i = 1, 2, \cdots k$.

- We used two predicates $\text{G}(i,j,b)$ and $\text{F}(i,j,b)$ to encode G and F respectively
- We ecndoded the graph S with a predicate $\text{S}(i,j)$
- $\text{NODE}(i)$ holds for every $i = 1, 2, \cdots, n$
- $\text{LAYER}(i)$ holds for $i = 1, 2, \cdots, l$
- $\text{XOR\_NODE}(I,J,L)$ holds iff at layer L, there is a node labelled x_I which is the result of the XOR between x_I and x_J — CNOT(x_j, x_i)
- $\text{XOR\_NODE}(I,J,L)$ can hold iff $(j,i) \in E_S$
- $\text{VALUE}(Z,I,Y)$ holds if and only if $x_Y \in \text{VAL}_I(x_Z)$

Testing Generation

For each $n \in \{4, 5, 6, 7, 8\}$:

- we generated 10 different random tests
- For each test:
  1. we created an $n \times n$ boolean matrix—G
  2. we picked a number k between 1 and n—the number of F_i
  3. we generated k different n boolean vectors—the set of F_i

What about S in the tests

In the tests,we used the following S:

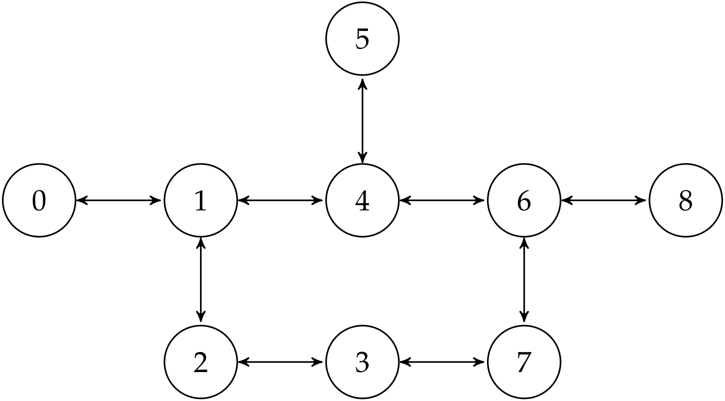


Figure 2: Guadalupe Quantum Computer topology reduced to 8-qubit.

Testing Algorithm

For each test case G, S, $\{F_i\}$:

1. we initialized a counter l to 1
2. we run the model with input G, $S, \{F_i\}, l$ to see if the instance was solvable with l layers
3. if true, we moved to the next example
4. if false, we increased l and got back to step 2

Results

n	avg time (seconds)
4	0.023
5	0.702
6	20.212
7	45.548
8	98.721

(a) Results without the topology.

n	avg time (seconds)
4	0.020
5	0.85
6	60.357
7	200.948
8	> 500

(b) Results with the topology.

Figure 3: Test results for the model with and without topology.

Conclusions and Future Works

Conclusions:

- We proposed an ASP models to minimize the number of CNOT gates in a {CNOT, T} circuit
- In doing so, we took into account also the underlying qubit topology
- We run a batch of tests to see the method efficiency.

Future Works:

- We want to optimize the model to make it faster
- We want to investigate also constraint programming approaches
- We think that synthesis problem can be tackled as the sum of small optimization subproblems

[1] **Carla Piazza, Riccardo Romanello, and Robert Wille.**
An asp approach for the synthesis of cnot minimal quantum circuits.
volume 3428, 2023.