Cut Searcher MIQCP Solver Problem Statement

Wei Tang
Princeton University
weit@princeton.edu

Abstract

1 Background

A quantum circuit can be represented by a directed acyclic graph (DAG) as shown in Figure 1. There are three types of vertices in a DAG representation of a quantum circuit. The first is input vertices, represented by green circles, with names written as text in the circles. They are the input qubits into a quantum circuit. The second is operation vertices, represented by blue circles. They take qubit inputs and output changed qubit state based on the exact operation they represent, which is written as text in the circles. In quantum computing, an operation vertex always has the same number of input and output edges. In our specific case, all the blue vertices always take exactly two inputs and have two outputs. The last type is the output vertices, represented by red circles. They are the final output of the quantum circuit, and each input qubit vertex has exactly one corresponding output qubit vertex.

The edges in the DAG represent the flow of a qubit from its input vertex, through some operation vertices, eventually reach its output vertex. An important feature of a DAG representation of quantum circuit is that a qubit input vertex never splits. It always has exactly one path from its input vertex to its output vertex. For example, Figure 1 has 12 input vertices and also exactly 12 output vertices. This is the non-cloning theory in quantum computing.

2 Task

The problem with quantum computing hardware nowadays is that they can only handle very few number of qubit inputs. We need to cut a large quantum circuit into smaller fragments and run them separately. Figure 2 shows an example to cut a circuit. After cutting on those two edges, qubit qreg[5] and qreg[9] are separated from the rest of the circuit, and the circuit is split into two clusters.

In order to determine how good a particular cut is on a circuit, I define two variables K and d. K is the number of edges being cut in each of the cluster resulted. In the example of Figure 2,

$$\begin{aligned} K_0 &= 2 \\ K_1 &= 2 \end{aligned}$$

Note that if the circuit is only cut into two clusters, the two clusters always have the same K. However, this is generally not true if the circuit is cut into more than two clusters.

The other variable d represents the number of qubits required for each of the cluster resulted. In the example,

$$\begin{aligned} d_0 &= 2 \\ d_1 &= 12 \end{aligned}$$

This is because cluster 0 now only contains 2 qubit input vertices. They go through one single operation vertex and will output (the two new output vertices created in this cutting are omitted in

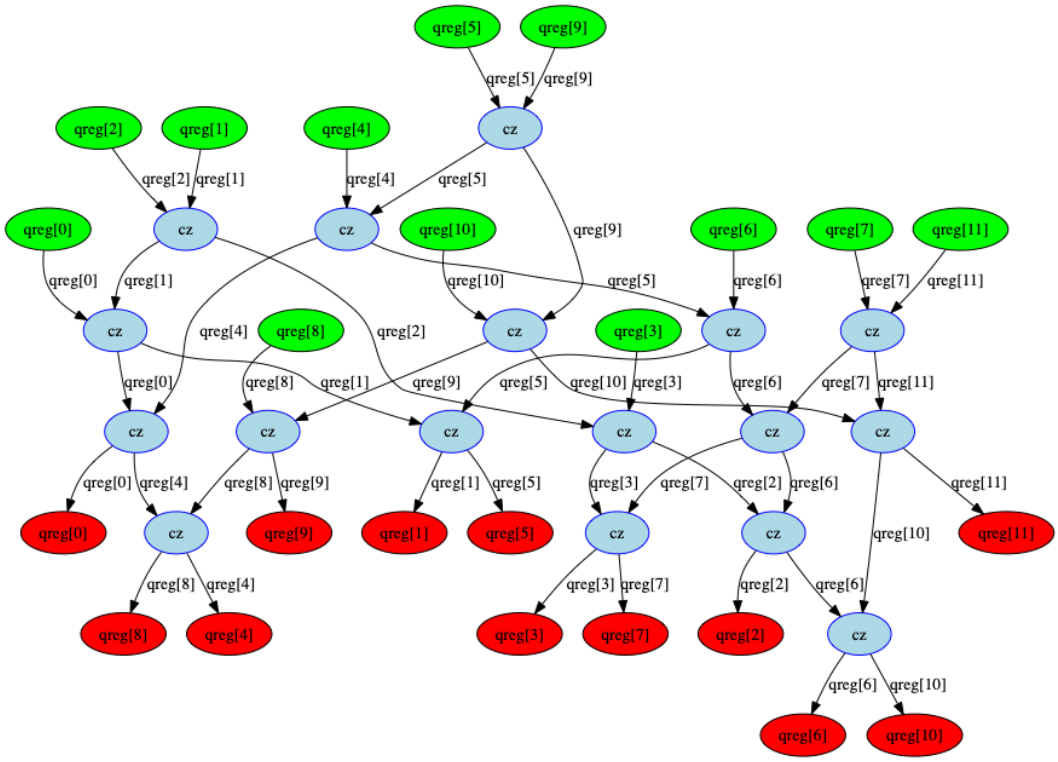

Figure 1: A directed acyclic graph (DAG) representation of a quantum circuit.

the drawing). Cluster 1 requires the original 10 other input qubits. However, note that it also needs 2 more input qubits (the two new input vertices created in this cutting are omitted in the drawing) to go into the two 'cz' operation vertices.

The overall memory requirement M to run and store the results of the clusters of a cutting is defined as follow:

$$M = \sum_{i=0}^{r-1} 2^{d_i} \times 8^{K_i} \tag{1}$$

where r is the number of clusters cut into, assumed given. And each d_i must be smaller than the maximum number of qubits allowed by our hardware, D, assumed given. The objective is to minimize the memory requirement M.

3 Problem Statement

Given a DAG representation of a quantum circuit G, maximum number of qubits allowed D, and number of clusters r. Cut some edges in G to split into r clusters. Define K_i to be the number of edges being cut in cluster i. Define d_i to be the number of qubits required to run cluster i. Find an optimal set of edges to cut to minimize

$$M = \sum_{i=0}^{r-1} f(d_i, K_i) \tag{2}$$

where

$$f(d_i, K_i) = \begin{cases} 2^{d_i} \times 8^{K_i}, & \text{if } d_i \leq D \\ \infty, & \text{otherwise} \end{cases}$$

4 MIQCP Formulation

The formulation here is inspired by [1].

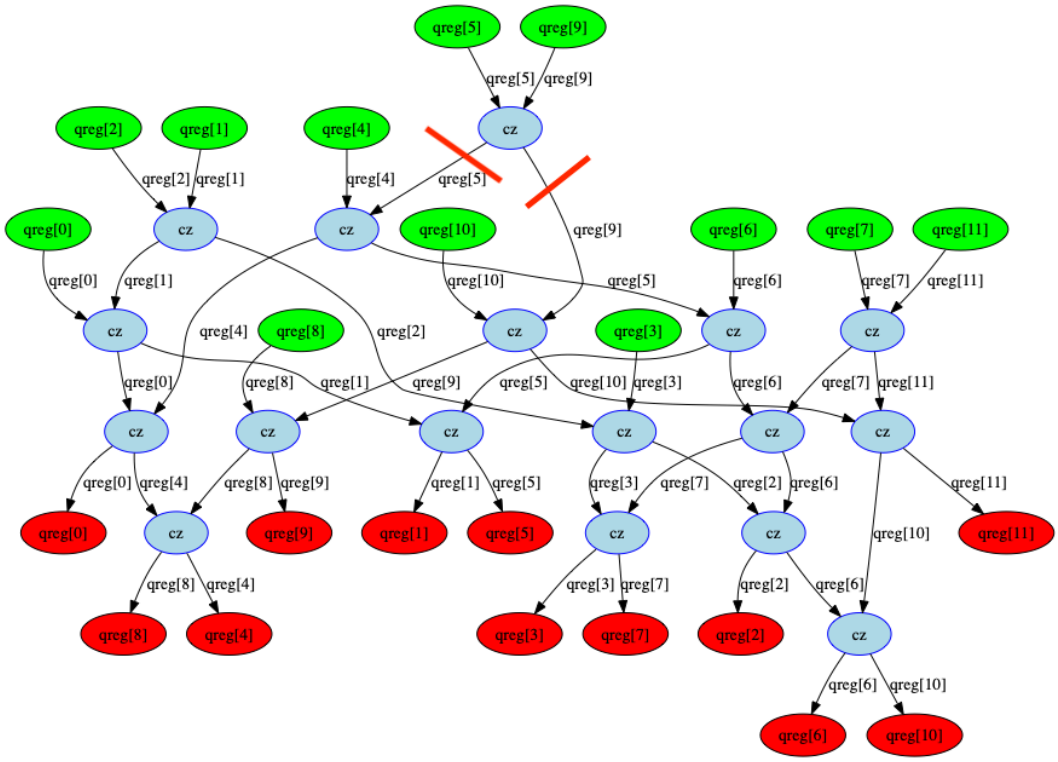

Figure 2: A bad way to cut a quantum circuit.

Since cutting edges to split G into r clusters is essentially to decide which operation vertices should belong to the same cluster, I define the following variables (the green and red vertices can be ignored to simplify our formulation, hence vertex refers only to the blue operation vertices):

$$v(i,c) = \begin{cases} 1, & \text{if vertex } i \text{ is in cluster } c \\ 0, & \text{otherwise} \end{cases}$$

$$w(i) = \begin{cases} 0, & \text{if vertex } i \text{ has 0 input qubits in } G \\ 1, & \text{if vertex } i \text{ has 1 input qubit in } G \\ 2, & \text{if vertex } i \text{ has 2 input qubit in } G \end{cases}$$

$$ce(i,c) = \begin{cases} 1, & \text{if edge } i \text{ has one and only one of its vertices in cluster } c \\ 0, & \text{otherwise} \end{cases}$$

I present the constraints required next.

4.1 Every vertex exists in one and only one cluster

This is easy and simply represented by the sum of v.

$$\sum_{c=0}^{r-1} v(i,c) = 1, \forall i \in |V| \tag{3}$$

4.2 An edge is cut if it has one and only one of its vertices in a cluster

The constraint on variable ce is defined as:

$$ce(i,c) = v(a,c) \oplus v(b,c), \forall i \in |V|, \forall c \in r \tag{4}$$

where edge i is an edge from a to b. Translate this constraint into linear constraints:

$$ce(i,c) \leq v(a,c) + v(b,c) \tag{5}$$
$$ce(i,c) \geq v(a,c) - v(b,c) \tag{6}$$
$$ce(i,c) \geq v(b,c) - v(a,c) \tag{7}$$
$$ce(i,c) \leq 2 - v(a,c) - v(b,c) \tag{8}$$

Table 1 details the explanation for how this works.

v(a,c)	v(b,c)	constraints on ce(i,c)	ce(i,c)
0	0	$\leq 0, \geq 0, \leq 2$	0
0	1	$\leq 1, \geq -1, \geq 1$	1
1	0	$\leq 1, \geq 1, \geq -1$	1
1	1	$\leq 2, \geq 0, \leq 0$	0

Table 1: Explanation of linear constraint translation for XOR

4.3 Symmetry Breaking

Clustering problems often have an inherent symmetry which is due to the fact that the labels of clusters are arbitrary. This means that for each solution, there are $r!$ equivalent solutions that only differ by the cluster labels. To break such symmetry, I assign label 0 to the cluster that contains vertex 0, assign labels to other clusters in the increasing order of their cluster sizes. This translates to the following:

$$v(0,0) = 1 \tag{9}$$
$$\sum_{i=0}^{|V|} v(i,c) \leq \sum_{i=0}^{|V|} v(i,c+1), \forall c \in 1, ..., r-2 \tag{10}$$

However, this constraint does not break the symmetry where multiple optimal solutions exist that assign vertex 0 to different clusters.

4.4 Objective Function

The variable K_c in the objective function is easy to model with our defined variable ce.

$$K_c = \sum_{i=0}^{|E|} ce(i,c), \forall c \in 0, ..., r-1 \tag{11}$$

Note that the variable d_c in the objective function is the sum of

1. number of original green input vertices
2. number of incoming edges being cut in the cluster

Hence

$$d_c = \sum_{i=0}^{|V|} w_i \times v(i,c) + \sum_{i=0}^{|E|} ce(i,c) \times v(b,c), \forall c \in 0, ..., r-1 \tag{12}$$

where edge i is an edge from a to b. Each term in the summation of M can then be approximated by a piecewise linear function. The infinity when $d_i > D$ can be approximated by a hard coded infinity value.

To prevent the problem of float overflow, objective function is manually scaled down by a factor of 2^{10}.

$$M = \sum_{i=0}^{r-1} f(d_i, K_i) \tag{13}$$

where

$$f(d_i, K_i) = \begin{cases} 2^{d_i/10} \times 8^{K_i/10}, & \text{if } d_i \leq D \\ \infty, & \text{otherwise} \end{cases}$$

5 Benchmark Results

To test the performance of MIQCP solver, I also implemented a randomized cut searcher using graph contraction and an exhaustive searcher that is extended from the randomized searcher.

The idea behind a randomized searcher is that it randomly contracts the graph edges until only a given number of clusters of vertices remain in the circuit graph. The remaining edges that were not selected to contract become the edges we want to cut. An exhaustive searcher simply iterates through all possible orders of contraction.

The original problem statement assumes that number of cluster r is given. In order to find the best number of clusters, I repeated the cut searchers for a reasonable range of clusters and output the best solution.

Figure 3 shows the comparison of using randomized searcher with different number of trials against MIQCP solver. The circuit examples used are square Google quantum supremacy circuits with a fixed depth of 8. The circuit sizes range from 2×2 to 6×6, which has number of qubits ranging from 4 to 36. Left panel shows that MIQCP solver always finds an equally good if not better objective value than randomized searchers. Randomized searchers approach the performance of MIQCP solver with more and more trials, but never exceed.

The right panel shows the run time comparison. MIQCP solver is thousands times faster than randomized searcher with $1e^5$ trials and still achieves better, in fact, optimal objectives.

The objective values found are verified against exhaustive searcher for up to 2×3 supremacy circuits. Since the exhaustive searcher suffers from factorial scaling in terms of edges in the circuit graph, a 3×3 supremacy circuit require about 450 days to run and was aborted.

References

[1] Behrouz Babaki, Dries Van Daele, Bram Weytjens, and Tias Guns. A branch-and-cut algorithm for constrained graph clustering. 2017.

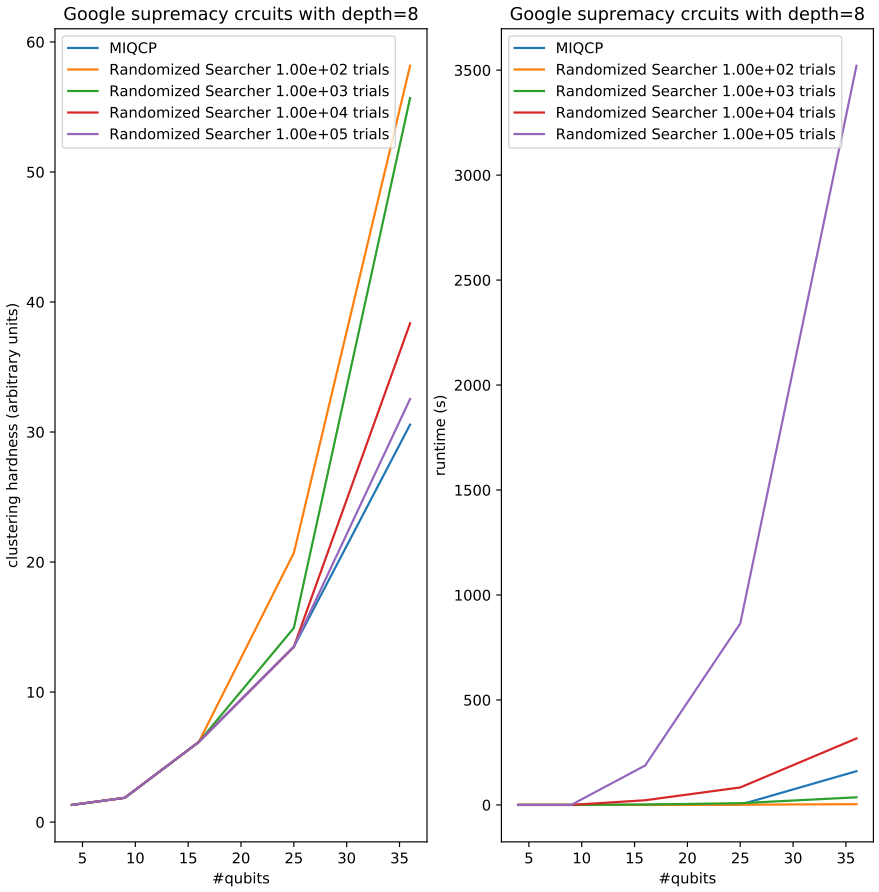

Figure 3: Cut searcher benchmark results against randomized edge contraction.