2QANIM: 2QAN Compiler Improved

S. S. Kahani

January 22, 2023

Abstract

TODO

1 Introduction

While we are far from a general-purpose quantum computer [**?**] that speeds up solving a vast range of problems, early real-world application could still be happening anytime soon by focusing on specific problems. Therefore, the importance of specific-purpose compilation is undeniable.

Moreover, it is guessed that the early applications of quantum computers will include but not be limited to discrete optimizations and physical simulations. [**?**] This means that studying the compilation of k-local Hamiltonians is a meaningful choice as they play a significant role in the optimizations (like in QAOA [**?**] and VQE [**?**]) and undoubtedly in the physical simulations as well. [**?**]

TODO related work

This work is based on the 2QAN compiler [**?**] which is a hardware-aware compiler for 2-local Hamiltonians. The main contribution of this work is to improve the compilation of 2QAN by taking errors into account and introducing a new compilation algorithm.

TODO classical compilers

Here we introduce a hardware-aware compiler, starts from a 2-local Hamiltonian, defined in Definition 1, allocates qubits and generates intermediate gates that are studied in Section **??** and finally transpile them to a lower-level gate-set. This compiler aims to reduce error in a reasonable compile-time, Therefore a detailed description of the assumed error model is given in Section 4.

2 Basic Concepts

Because of the exponential growth of the number of gates in the compilation of a k-local term [**?**], setting $k = 2$ is a reasonable choice for NISQ devices. This simplification not only helps us by reducing the number of gates, also enables us to use graph theory to define and analyze the problem, which is not easily possible for higher values of k.

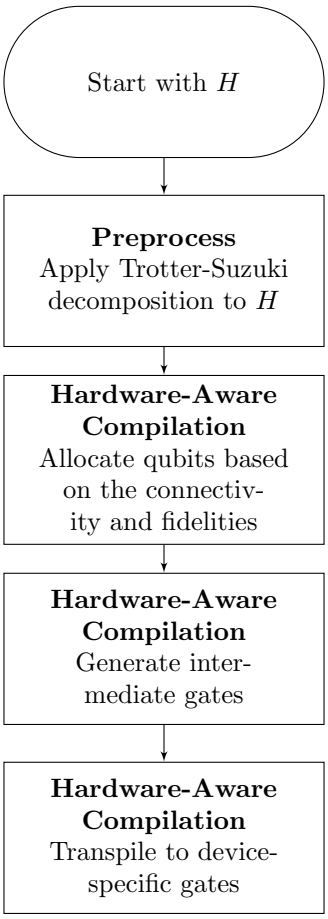


Figure 1: The overview of the compilation process in 2QANIM

Here we introduce our notation for the Hamiltonian.

Definition 1 (2-local Hamiltonian)**.** *Defining an N-qubit Hilbert space $\mathcal{H} := \otimes_{i=1}^{N} \mathcal{H}_{\rangle}$, a 2-local Hamiltonian H acting on it will be like*

$$H := \sum_{i=1}^{k} h_i \tag{1}$$

where each h_i is a 2-local term, acting nontrivially on two specific qubits.

Definition 2 (Graph representation of a Hamiltonian)**.** *We will define the graph $G_H(V_H, E_H)$ to represent the Hamiltonian H, in a way that for each term h_i in H, we have an edge e_i between the two qubits it acts on.*

The very first step toward the compilation of a 2-local Hamiltonian is to apply the first-order Trotter-Suzuki decomposition [**?**] to it. This decomposition

is a way to approximate the time evolution of a Hamiltonian by a sequence of gates.

Definition 3 (Trotter-Suzuki decomposition)**.** *The first order Trotter-Suzuki decomposition, decomposes a Hamiltonian H, into a set of two-qubit gates g_is which associated with an edge like e_i.*

$$\textit{Target Gates} := \{(g_i, e_i)\} \tag{2}$$

The key feature of this decomposition (only the first order) is that the resulting gates can be applied in any order. This arbitrariness is not hold for a general quantum circuit and it may cause an advantage for a problem-specific compiler.

3 Two Qubit Gates

Here we mention the two-qubit gates that are used as an intermediate gates in the process of compilations.

3.1 SWAP and Bridge Gates

In a situation that a mapping is already applied, the main limitation is the connectivity of the device, therefore where two qubits are not directly connected, we need to employ a chain of gates, to apply the target gate on them.

The SWAP gate is well-known, but it is not the only gate that can be used for this purpose. The bridge gates that are introduced in [**?**] for a simple case, can be expanded and employed for this purpose.

The bridge gate is an equivalent expansion of a CNOT gate, applied on two qubits that are connected through a chain of qubits.

The bridge gate is defined as follows:

Definition 4. *Bridge Gate*

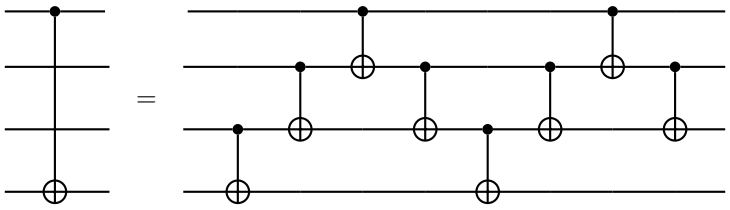


Figure 2: The bridge gate

$$\textit{Bridge}(i,j) := \left(\prod_{k=i+2, k<j}^{\leftarrow} CNOT(k-1,k) \prod_{k=j, k>i}^{\leftarrow} CNOT(k-1,k) \right)^2$$

As far as we know, the family of bridge gates with more than one qubit in the chain is not studied yet, and neither implemented in any quantum compiler. It can be easily shown that for a simple case like the one in the figure **??** that we need to swap and return the qubits, using bridge gate can reduce both depth and number of gates. DOUBT

3.2 Unifying Gates

TODO We can combine two-qubit gates...

4 Error Model

Here we introduce a model to estimate errors of a circuit on a device. In this model, we take the average infidelity of a circuit as a measure of the error. In order to calculate the average (in)fidelity, we consider two different source of errors.

- **Decoherence**: When a qubit is left alone for some amount of time, it will decay to a random state. The average fidelity of the qubit can be written as:
$$F_{\text{decoherence}}(t) = \exp\left(-\frac{t}{\tau_{\text{decoherence}}}\right) \tag{3}$$
- **Gate Errors**: Any gate, specially the two-qubit gates, are not perfect. We can assign an average fidelity to each gate.

As for any parallel or sequential processes, the average fidelity of the circuit is the product of the average fidelity of each process, we use $-\log \mathcal{F}$, as a positive measure of error with additivity.

Therefore, a device is defined as a tuple of the following:

Definition 5 (Device)**.**

$$\textit{Device} := (w_0, G_d(V_d, E_d, w_{\mathcal{F}}))$$

where

$$\begin{cases} w_0 = \tau_{\textit{2-qubit gate time}}/\tau_{\textit{decoherence}} \\ w_{\mathcal{F}}(e) = -\log\left(\bar{\mathcal{F}}_{\textit{2-qubit gate on edge } e}\right) \end{cases}$$

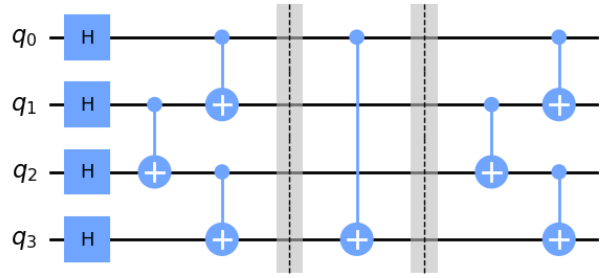
5 Problem Formulation

5.1 NP-Hardness

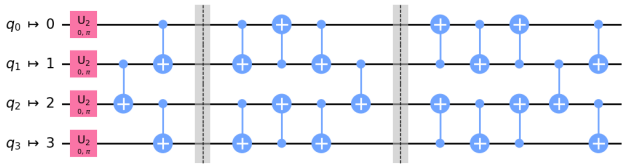
Anyhow we define the problem, it is inevitable to have a subproblem like below

Problem 1 (Qubit Allocation)**.** *Given a graph that represents a Hamiltonian $G_H(V_H, E_H)$, find an allocation $\phi : V_H \to V_d$ such the maximum number of edges in E_H are mapped to E_d.*

a)



b)



c)

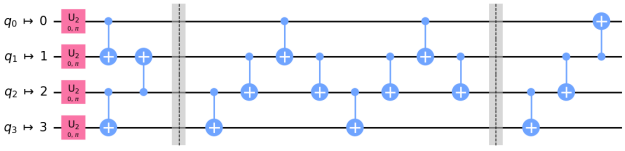


Figure 3: a) The original circuit, consisting of some local operations, then a far away CNOT and then some local operations. b) The circuit after transpiling using Qiskit. c) The circuit after transpiling using bridge gate as an intermediate gate.

We may use any other more complex criteria, such as minimizing estimated circuit error or minimizing the number of gates in the implemented circuits. But even in the simplest case, the problem is NP-hard, as the graph isomorphism problem can be reduced to it [**?**].

But yet, this argument could be easily misinterepted into the fact that it is impossible to address the exact compilation algorithm in any case. It states that for an abitrary device graph, the problem is NP-hard. If we just add a simple constraint that the device graph has a bounded degree, then the problem is solvable in polynomial time. [**?**]

5.2 General Treatment

6 Algorithms

6.1 2QAN

6.2 A*

6.3 Resource Allocation