2QANIM: 2QAN Compiler Improved

S. S. Kahani

February 27, 2023

Abstract

TODO

1 Introduction

While we are far from a general-purpose quantum computer [3] that speeds up solving a vast range of problems, early real-world application could still be happening anytime soon by focusing on specific problems. [6] 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 [2] and VQE [5]) and undoubtedly in the physical simulations as well. [**?**]

We may count the famous Trotter-Suzuki decomposition (see 3) as the first effort in compilation of Hamiltonians. But other works does not necessarily rely on that like [1]. TODO related work

This work is based on the 2QAN compiler [4] 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

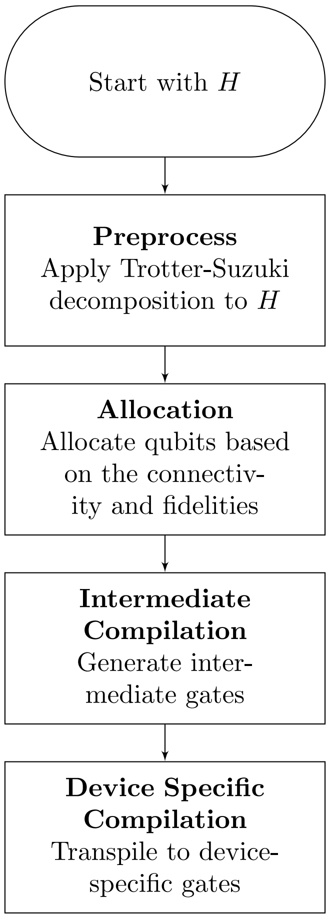

Start with H
Preprocess
Apply Trotter-Suzuki decomposition to H
Allocation
Allocate qubits based on the connectivity and fidelities
Intermediate Compilation
Generate intermediate gates
Device Specific Compilation
Transpile to device-specific gates

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

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.

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 [7] 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 Intermediate Gate Set

In order to breakdown the process of compilation, which is surely a complex task, we use an intermediate description of the circuit, in terms of the intermediate gate set.

This gate set is presumed to not be directly available on any device, but rather, it is easy to generate them relatively efficient on a quantum device. It is not wondering that we assume that the intermediate circuit must obey the topology of the device.

While the implementation of one-qubit gates is not challenging, it is convenient to have all of the one-qubit gates, in the set, but for two qubit gates, many restrictions must be applied.

The most important two-qubit gate that is often a part of any gate set is the CNOT gate. And because of the topology of the device, the CNOT might not be available on all pairs of qubits. Therefore we often add SWAP gate to make it possible all the way through the circuit, but optimality of the circuit is questionable in this case. Here we introduce a new gate, which is a generalization of the CNOT gate, which is shown to be more efficient than applying a SWAP gate.

3.1 Introducing Bridge Gates

For a simple case, that we have three qubits, called a, b, c, and we want to apply a CNOT gate on (a, c), but the connectivity only allows us to apply a CNOT gate on (a, b) and (b, c), the first solution would be to use a SWAP gate, which is shown in figure 3.1 (7 gates, depth of 7) While another approach is to use a bridge gate, which is shown in figure 3.1 (4 gates, depth of 4).

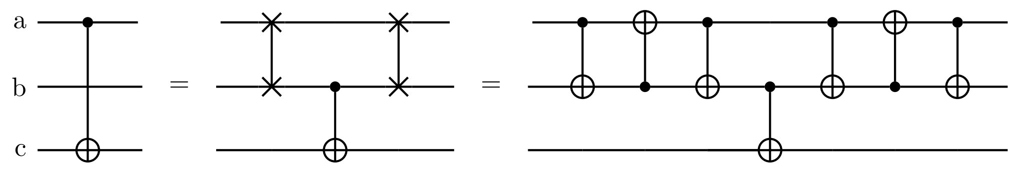


Figure 2: Applying a CNOT gate on (a, c) using a SWAP gate

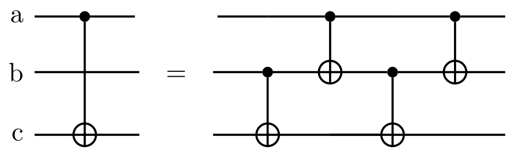


Figure 3: Applying a CNOT gate on (a, c) using a bridge gate

This example, could be generalized and the bridge gate must be generalized as well. Hereby we define the generalized version of the bridge gate and we show the optimality of the bridge gate in terms of the number of CNOT gates and the depth of the circuit.

Definition 4. *Generalized Bridge Gate For an even n*

$$\begin{aligned}\mathrm{Bridge}(1,n) = &\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i+1,i)\mathrm{CNOT}(n-i+1,n-i)\\ &\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i,i+1)\mathrm{CNOT}(n-i,n-i+1)\\ &\mathrm{Bridge}(n/2-1,n/2)\\ &\left(\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i+1,i)\mathrm{CNOT}(n-i+1,n-i)\right)^{\dagger}\\ &\left(\prod_{i=1}^{n/2-1} \mathrm{CNOT}(i,i+1)\mathrm{CNOT}(n-i,n-i+1)\right)^{\dagger}\end{aligned}$$

but for an odd n, it is a little bit different.

Theorem 1. *Assume that we decomposed a CNOT between the first and the last qubits in a chain, in terms of local circuit. This circuit must carry information from the first bit to the last and reverse. For the forward information flow, the information is an X gate placed on the first qubit, will travel along the circuit, so*

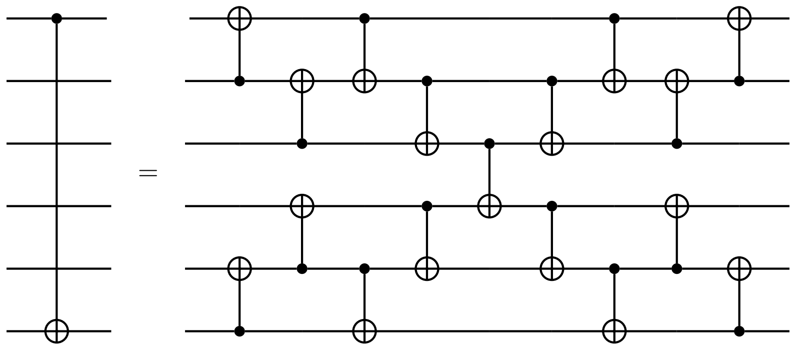


Figure 4: The bridge gate for $n = 6$

we need to have a consequential n CNOT gates to carry this information. The same argument is valid for a backward series of gates. Ignoring the first and the last qubit, for each qubit this argument is also valid that, the gate in those series must not be the first neither the last CNOT gate applied on the qubit. Therefore, at least we need to have two series, each has to have a range of CNOT gates, before and after. One could easily manage to find out that the simplest structure would be similar (in terms of complexity) to the proposed bridge gate.

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.

3.2 Unifying Gates

TODO We can combine two-qubit gates...

4 Error Model

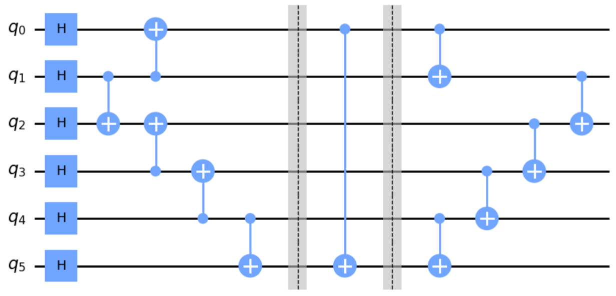
Here we introduce a model to estimate errors of a circuit on a device. In this model, we take the average infidelity of a circuit as a measure of the error. In order to calculate the average (in)fidelity, we consider two different source of errors.

- **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:

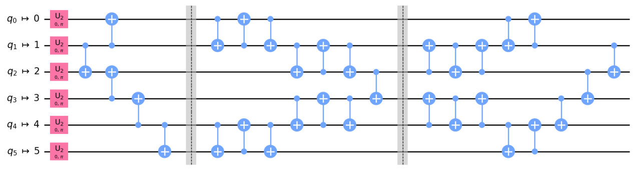
$$F_{\text{decoherence}}(t) = \exp\left(-\frac{t}{\tau_{\text{decoherence}}}\right) \tag{3}$$

- **Gate Errors**: Any gate, specially the two-qubit gates, are not perfect. We can assign an average fidelity to each gate.

a)



b)



c)

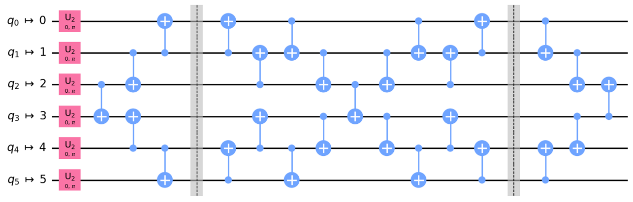


Figure 5: a) The original circuit, consisting of some local operations, then a far away CNOT and then some local operations. b) The circuit after transpiling using Qiskit. c) The circuit after transpiling using bridge gate as an intermediate gate.

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)**.**

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

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.*

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 misintepreted 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

References

[1] Earl Campbell. “Random Compiler for Fast Hamiltonian Simulation”. In: *Physical Review Letters* 123.7 (Aug. 2019), p. 070503. DOI: `10.1103/PhysRevLett.123.070503`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.123.070503` (visited on 01/22/2023).

[2] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm.* Tech. rep. arXiv:1411.4028 [quant-ph] type: article. arXiv, Nov. 2014. DOI: `10.48550/arXiv.1411.4028`. URL: `http://arxiv.org/abs/1411.4028` (visited on 01/22/2023).

[3] Matt Langione et al. "Where will quantum computers create value—and when". In: *Boston Consulting Group* (2019), p. 19.

[4] Lingling Lao and Dan E. Browne. *2QAN: A quantum compiler for 2-local qubit Hamiltonian simulation algorithms.* Nov. 7, 2021. arXiv: `2108.02099`.

[5] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". en. In: *Nature Communications* 5.1 (July 2014), p. 4213. ISSN: 2041-1723. DOI: `10.1038/ncomms5213`. URL: `https://www.nature.com/articles/ncomms5213` (visited on 01/22/2023).

[6] John Preskill. "Quantum computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[7] Hale F Trotter. "On the product of semi-groups of operators". In: *Proceedings of the American Mathematical Society* 10.4 (1959), pp. 545–551.