Qubit Allocation

Marcos Yukio Siraichi[1]

Advisor: Fernando Magno Quintão Pereira[1]
Co-advisors: Vinícius Fernandes dos Santos[1], Caroline Colange[2]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

[2]Inria – Université de Rennes – CNRS – IRISA
Rennes – France

Abstract. *The availability of the first prototypes of quantum computers, in 2016, with free access through the cloud, brought much enthusiasm to the research community. However, programming such machines is difficult. One core challenge is the so called "qubit allocation problem". This problem consists in mapping the virtual qubits that make up a logical quantum program onto the physical qubits that exist in the target quantum architecture. To deal with this challenge, we have proposed one of the first algorithms to solve qubit allocation. This algorithm, together with its ensuing formulations, is today available in the Enfield compiler—a concrete product of this work. Our first paper in this field, titled "Qubit Allocation", has inspired much research, and our latest qubit allocation design, called "Bounded Mapping Tree", stands out today as one of the most effective qubit allocators in the world.*

1. Introduction

In recent years, the advance of technology has enabled the development of bigger-scale practical quantum machines, making them accessible not only for field researchers, but also for the general public. One such example of publicly available quantum prototype is the IBM Quantum Experience platform[1]. Given the computational possibilities that quantum computers are known to have, these first prototypes have elicited much enthusiasm. Theoretically, these quantum devices can be used for accelerating the computation of problems that are impractical in classical computers, such as integer factorization—a core step in many implementations of cryptographic routines. However, implementing algorithms in a quantum computer is not a simple endeavour. One key obstacle to the implementation of these algorithms is the challenge of mapping a logical quantum circuit into the physical quantum architecture.

In this context, an important architectural detail that must be taken into consideration is the coupling graph of the given quantum machine. It defines the machine's inter-connection between two information storage units, the so called *quantum bits* or *qubits* for short. Qubit connections enable one of the fundamental operations of quantum computers: to correlate two different qubits. The coupling graph poses a problem to developers because different architectures allow different qubits to correlate; hence, it is impossible for some programs to be executed in a given quantum computer without any modification.

[1]`https://quantum-computing.ibm.com/`

There are some specific operations that allow us to re-map logic qubits onto physical qubits. Such operations enable the implementation of any quantum program on a given architecture with a connected coupling graph. Yet, as one might notice, these operations enlarge the program, increasing runtime and noise introduced to the system. Even though error correction algorithms have been studied to reduce this noise, they demand too many qubits. Given that current quantum machines have few qubits, using error correction is presently impractical. Therefore, it is important to design and implement compilers that translated logical quantum circuits into physical quantum machines using as few transformations as possible.

Theoretical Contributions of this Work. In this dissertation, we have combined graph theory and compilation techniques to solve the aforementioned problem, henceforth called *Qubit Allocation*. In the effort to solve this problem, we have designed and implemented three different solutions to it:

1. a dynamic programming exponential exact algorithm to serve as baseline for smaller instances;
2. a fast and straight-forward algorithm that achieved comparable results in smaller architectures; and
3. a polynomial parameterized algorithm that outperformed (quality-wise) all of the state-of-the-art algorithms for bigger architectures.

In addition to the implementation of this algorithm, this dissertation also brought other, more basic, theoretical contributions, namely:

- A formal definition of the qubit-allocation problem.
- A demonstration that qubit-allocation, and its many variants, are NP-complete problems—a result of ours cited by several other researchers since publication.

The Products of this Dissertation. As a by-product of this dissertation, we also developed an entirely new openly available compiler called Enfield [2]. Enfield translates OpenQASM, a well-known quantum programming language, into one of the several quantum architectures proposed by IBM. In addition to our own algorithms, Enfield also implements seven other solutions to qubit allocation, which are useful for comparisons. Several papers have been published as direct consequence of this work:

- Best paper finalist at "International Symposium on Code Generation and Optimization" (CGO'18–Qualis A2) [Siraichi et al. 2018]. This paper, titled "Qubit Allocation" was the first to examine the theoretical properties of the namesake problem, including NP-completeness. It counts 39 citations accumulated in 19 months—only one of them from our research group.
- Published the "Bounded Mapping Tree" algorithm in the Conference of Object-Oriented Programming, Systems, Languages & Applications (OOPSLA'19–Qualis A1) [Siraichi et al. 2019]. In this paper we established approximation bounds to qubit allocation, and developed what, to the best of our knowledge, is one of the most effective qubit allocators available today;
- Second best tool in CBSoft 2018 Tools Session, with the paper "Enfield: An OpenQASM Compiler" [Siraichi and Tonetti 2018].

[2] `http://cuda.dcc.ufmg.br/enfield/`

2. Qubit Allocation – Theoretical Background

Quantum programs are made of qubits and reversible quantum gates, which receive qubits as inputs, and produce qubits as outputs. Figure 1 shows a quantum circuit, which implements two boolean functions. This circuit has four qubits: a_0, a_1, b_0 and b_1, which are represented as horizontal lines. It uses four different types of gates to operate on these qubits: H, T, $T^\dagger$ and CNOT, where CNOT_{ab} is depicted with a dot on qubit a and $\oplus$ on qubit b. Gates change the *state* of qubits. How exactly this happens is immaterial to this presentation. It suffices to know that each one of these gates represents an operation of matrix multiplication. The final state of each qubit, such as r_0 and r_1 on the right side of Figure 1, is determined by the result of these multiplications.

The only aspect of consequence in the context of this work is the placement of CNOT gates. CNOT gates matter due to *architectural constraints*. Quantum computers based on superconducting qubit technology are made of solid-state circuits that only allow CNOT interactions between qubits that are physically connected [Devoret et al. 2004, Koch et al. 2007]. As an example, Figure 2 (a) shows the *coupling graph* of the IBM qx2 computer [Devitt 2016]. The coupling graph determines which qubits can communicate. We define the coupling graph in terms of CNOT gates as follows:

Definition 1 (Coupling Graph) *Given a quantum architecture A with a set Q of qubits, its coupling graph is a directed graph $G_q = (Q, E_q), E_q \subseteq Q \times Q$. The edge $(q_i, q_j) \in E_q$ if, and only if, $\text{CNOT}_{q_1 q_2}$ is valid in A.*

Qubit Allocation – An Informal Overview. CNOT relations between qubits (henceforth referred to as pseudo qubits) in a quantum circuit need to be mapped to the coupling graph. For instance, in Figure 1, we have that the pseudo qubit a_0 controls b_0 and b_1, i.e. $\text{CNOT}_{a_0 b_0}$ and $\text{CNOT}_{a_0 b_1}$. When allocating pseudo qubits onto the coupling graph, we would like to enable such control relations. However, *perfect mappings* that enable all the control relations in a quantum circuit are not always possible, as Example 1 illustrates.

Example 1 *It is not possible to map the control circuit of Figure 1 onto the coupling graph of Figure 2 (a). Figure 2 (b) represents the control relations in that circuit. This graph contains two nodes of in-degree two, which have no equivalent in Figure 2 (a).*

The qubit allocation problem, which Definition 2 states, asks for a mapping between pseudos and qubits in the coupling graph (henceforth referred to as physical qubits) that respects the control relations. If a perfect mappings is not possible, then we must resort to *circuit transformations* to solve the problem.

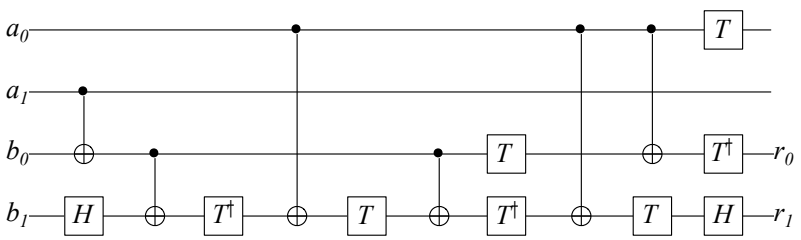

Figure 1. Example of Quantum Program.

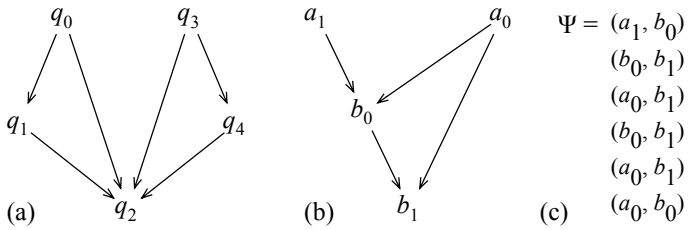

Figure 2. (a) The coupling graph of the IBM qx2 computer. (b) Interactions between qubits of the circuit seen in Figure 1. (c) Dependences that have created these interactions.

Definition 2 (The Qubit Assignment Problem) ***Input:*** *a coupling graph $G_q = (Q, E_q)$, plus a list $\Psi = (P \times P)^n, n \geq 1$ of n control relations between pseudo qubits.* ***Output:*** *yes, if there is a mapping between pseudo and physical qubits that respects the control relations in Ψ.*

Circuit Transformations. A transformation is a combination of gates that we can insert into a quantum circuit to emulate the semantics of non-existing CNOT relations. Figure 3 describes two of these transformations. As Figure 3 shows, a CNOT reversal allows the mapping of "backward" edges on the coupling graph, at the cost of extra gates. A CNOT swap allows the migration of pseudo qubits across physical qubits. Figure 4 outlines a solution to qubit allocation for the program in Figure 1 using two CNOT reversals.

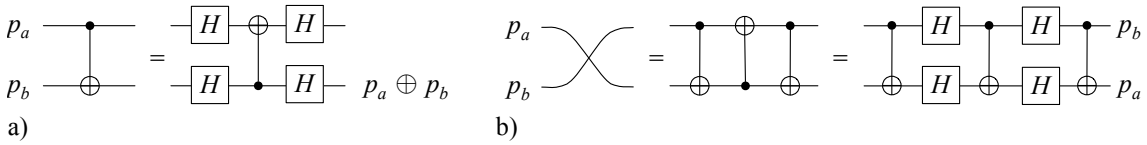

Figure 3. Two types of transformations. (a) Reversal: Emulation of a virtual CNOT between p_a and p_b. (b) Swap: exchanges two pseudo qubits p_a and p_b.

A particular instance of qubit allocation might have several different solutions. The quality of a solution is given by its *cost*, which we measure as the number of gates necessary to implement it. The main **contribution** of the research effort that resulted in this dissertation was the design and implementation of different algorithms to solve this optimization problem. Our algorithms yield quantum circuits with less gates and shorter latency—the minimum number of gates from the beginning until the end of the circuit.

3. Selected Results

The dissertation contains a plethora of experiments that compare different versions of our qubit allocators with state-of-the-art algorithms. In this section, we chose to show results involving seven different algorithms: `ibm`, `jku`, `chw`, and `sbr`; plus the two variations of BMT [Siraichi et al. 2019]: `bmtS` and `bmtF` and `wpm` [Siraichi et al. 2018]. The latter three algorithms are contributions of the dissertation. We compare these algorithms in terms of compilation time, cost of quantum circuits and speed of quantum circuits. We conducted experiments in two different quantum architectures, using an ensemble of 158 programs typically used to compare compilers for quantum programming languages.

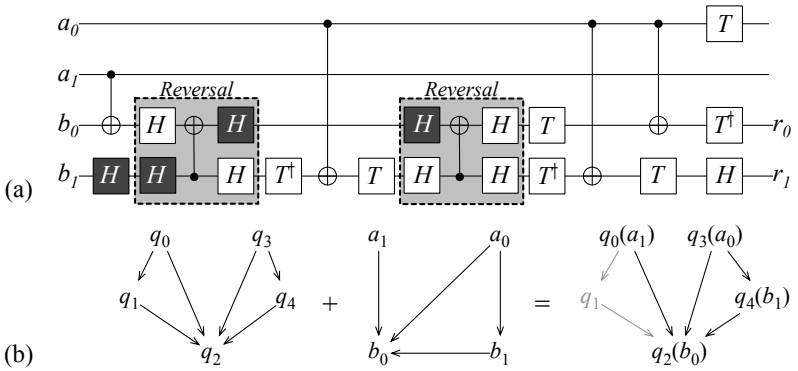

Figure 4. (a) CNOT reversals, marked as grey boxes, invert the direction of $\text{CNOT}_{b_0 b_1}$. (b) Solution to qubit allocation that maps the logical circuit from Figure 1 onto the coupling graph from Figure 2-a using the two CNOT reversals.

Metrics. The **cost** directly addresses the fact that different gates introduce a different amount of noise. Since there are only small quantum computers publicly available, we measured the latency through the quantum circuit **depth** (related to the computation time). Finally, the **compilation time** can be easily measured by executing the algorithm.

Figure 5 summarizes the comparison between the different qubit allocators when targeting the 20-qubit IBM Tokyo computer architecture. As the figure shows, the two algorithms of our design, `bmtS` and `bmtF` could outperform all the other allocators, in terms of average cost per benchmark, and in terms of absolute cost (when adding the number of gates in all of them). The good quality of the code produced by our allocators come with a price: the two variations of BMT have the highest compilation time. In practice, we are spending more compilation time to generate quantum circuits that are considerably better that what could be produced by other qubit allocators.

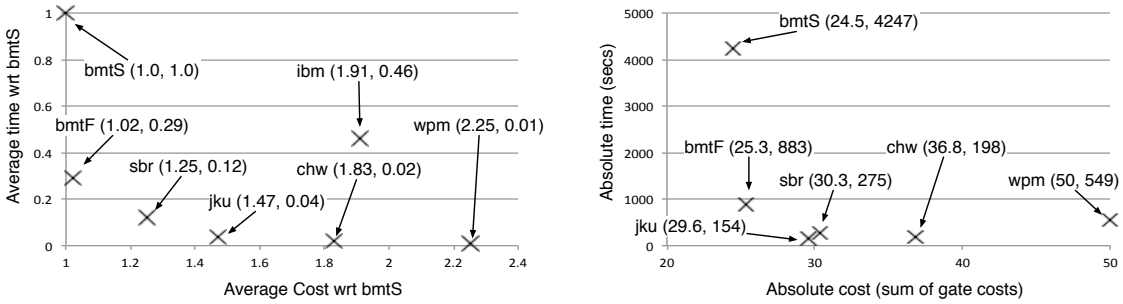

Figure 5. Compile Time vs Cost for all algorithms w.r.t. `bmtS` in the IBM Tokyo quantum architecture. The smaller (in both axis), the better.

Figure 6 provides an in-depth comparison between BMT and Sabre (`sbr`) [Li et al. 2019] in two quantum architectures from IBM: the 16-qubit Albatross, and the 20-qubit Tokyo. Sabre is regarded as one of the best qubit allocators available today. BMT tends to outperform Sabre in the larger architecture, Tokyo, which has a larger diameter per qubit relation, however falls short in Albatross, the smaller system. Addressing this issue, we generated random coupling graphs with different number of qubits and diameter. This result indicates that the denser the coupling graph, the better the results of our algorithms when compared with state-of-the-art approaches.

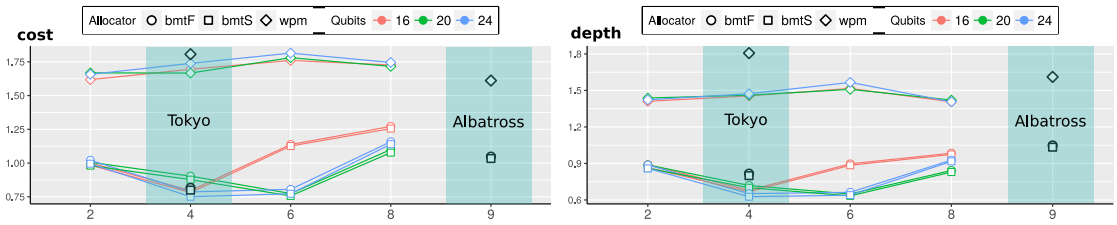

Figure 6. Cost (left) and Depth (right) results by different diameter of coupling graphs w.r.t `sbr`. The smaller (cost and depth), the better. Line colors refer to the number of qubits in the coupling graph. Point shapes represent different allocators. Single black points represents the overall results for each architecture (Albatross and Tokyo), as well as their place in the diameter axis.

4. Conclusion

This MSc dissertation has formalized the qubit allocation problem, a core step in the compilation of quantum programming languages into physical computer architectures. In addition to this formalization, we have provided a novel modeling of such a problem. This modeling led to efficient solutions that outperformed existing state-of-the-art qubit allocators. On heavily connected quantum architectures, such as IBM Tokyo, BMT—our most efficient design—found programs that are, on average, 25% cheaper and 40% faster than the state-of-the-art algorithm `sbr`. Analyzing our struggles and accomplishments with hindsight, we believe that the most important contribution of this work was the observation that the qubit allocation problem can be solved by combining the solution to multiple instances of two NP-complete graph problems: the subgraph isomorphism problem, and the token swapping problem. We hope that this observation will foster new implementations of qubit allocators that can be even more efficient than the solutions that we have found in the course of this work.

References

Devitt, S. J. (2016). Performing quantum computing experiments in the cloud. *Phys. Rev. A*, 94(3):032329.

Devoret, M. H., Wallraff, A., and Martinis, J. M. (2004). Superconducting qubits: A short review. *arXiv*, cond-mat/0411174:1–41.

Koch, J., Yu, T. M., Gambetta, J., Houck, A. A., Schuster, D. I., Majer, J., Blais, A., Devoret, M. H., Girvin, S. M., and Schoelkopf, R. J. (2007). Charge-insensitive qubit design derived from the cooper pair box. *Phys. Rev. A*, 76(1):04319.

Li, G., Ding, Y., and Xie, Y. (2019). Tackling the qubit mapping problem for nisq-era quantum devices. In *ASPLOS*, page 1001–1014. ACM.

Siraichi, M. Y., Santos, V. F. d., Collange, C., and Pereira, F. M. Q. (2019). Qubit allocation as a combination of subgraph isomorphism and token swapping. In *OOPSLA*, pages 120:1–120:29. ACM.

Siraichi, M. Y., Santos, V. F. d., Collange, S., and Pereira, F. M. Q. (2018). Qubit allocation. In *Inter. Symp. on Code Generation and Optimization*, page 113–125. ACM.

Siraichi, M. Y. and Tonetti, C. (2018). Enfield: An OpenQASM compiler. In *Congresso Brasileiro de Software, Sessão de Ferramentas*, Bento Gonçalves, RS, Brazil. SBC.