Quantum Circuit Synthesis using MILP and SMT Encodings

Jakub Havlík (xhavlij00@stud.fit.vutbr.cz)
Supervisor: doc. Ing. Ondřej Lengál, Ph.D

Excel@FIT 2026

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Motivation

The field of quantum computing is at a constant state of growth and many advancements have been achieved in the recent years. However, achieving the quantum advantage is not only about the rise of quantum computers, but also efficient quantum circuits.

- Quantum algorithms are hard to design – automated synthesis is a key step to creating efficient implementations (small, resource efficient circuits)
- Even with hundreds of qubits, a poorly optimized circuit renders the hardware's power useless
- Quantum architecture search (QAS) methods enable the discovery of new, novel implementations for various problems
- In the NISQ era of quantum computers, architectures differ by the native gate set, a key input for synthesis methods – we can achieve portability of quantum algorithms.
- Huge circuits introduce noise – the smaller the circuit, the higher the achievable fidelity

Problem Specification

The problem of quantum circuit synthesis, more specifically QAS, can be reduced to the following schema:

- Input Circuit specification C, cost function f, gate set G
- Output A quantum circuit, which only consists of gates from G and minimizes f

More specifically, $C = \{(v_i, t_i) \mid v \in \{|0\rangle, |1\rangle\}^{\otimes q}, t \in \mathbb{C}^{2^q}, i \in \{0, 1, \ldots, 2^q\}\}$, where q is the number of qubits, G is a universal gate set (e.g. Clifford+T), and f is usually circuit depth, T-count, or overall gate count.

Circuit Encoding

A quantum circuit can be encoded as a sequence of layers. A schema of a layer at a depth d consists of the sets of input vectors I^d and output vectors I^{d+1}, and a set of selection variables $Sel_d = \{s_{g,n,d} \mid g \in G_1, n \in \{0, 1, \ldots, 2^q\}\} \cup \ldots$ Each of the selection variables determines the gate, and the affected qubits. These variables constrain the application of the gate to all the input vectors by mapping it to the output vectors. Each of the layers assumes that only one gate is active for that specific layer.

Our current encodings are solved by a portfolio of SMT solvers (QF_LIA, QF_NRA), and MILP solvers.

Various representations of complex numbers

The representation of complex numbers significantly affects the performance of encoding and equivalence checking, since they handle gate operations and equivalence checks differently. Our methods support two different representations.

Algebraic representation of complex numbers using five-tuples (a, b, c, d, k):

$$\mathbb{D}[\omega] = \left\{ \frac{1}{\sqrt{2^k}} \cdot (a + b\omega + c\omega^2 + d\omega^3) \,\middle|\, k, a, b, c, d \in \mathbb{Z}, \omega = e^{\frac{j\pi}{4}} \right\} \quad (1)$$

The other, more common, representation is the basic $a + bj$, where $a, b \in \mathbb{R}$.

Equivalence Encoding

In equivalence checking, we distinguish exact and approximate synthesis. Our experiments show only the results of exact equivalence. The following equivalences have to hold for each pair of $(x, y), x \in I_{i,j}^{dmax}, y \in T_{i,j}, i \in \{0, \ldots, |T|\}, j \in \{0, \ldots, 2^q\}$, where T is a set of all the target vectors from C.

In the case of exact equivalence of five-tuples, the five-tuples have to be rescaled by the specific formula:

$$x = 2^{\lfloor \frac{\Delta k}{2} \rfloor} \cdot \begin{cases} M \text{ (iff } \Delta k \text{ is odd)} \\ I_4 \text{ (iff } \Delta k \text{ is even)} \end{cases} \odot y \quad (2)$$

where $x, y \in \mathbb{D}[\omega], x_k > y_k, \Delta k = x_k - y_k$. $\odot$ represents multiplication of a matrix with the vector of coefficients (a, b, c, d). M is a custom 4×4 rescaling matrix, and I_4 is the identity matrix with the same dimensions.

Moreover, if we include a global phase in the equivalence check, one of eight possible rotations of the number is selected before the rescaling, since the phase in $\mathbb{D}[\omega]$ is limited to $\omega^m, m \in \{0, \ldots, 7\}$.

On the other hand, the classic $a + bj$ representation simply checks $x = e^{j\phi} \cdot y$, where $x, y \in \mathbb{C}, |e^{j\phi}| = 1$. In the case of checking identity, $e^{j\phi}$ is reduced to 1.

Repeat-Until-Success (RUS) circuits

- Non-deterministic approximations of single-qubit unitaries
- RUS circuits show promising advancements in error correction codes, the implementation of quantum neural networks, and fault-tolerant quantum architectures

Figure 1 shows the full **repeat-until-success protocol** with 1 target qubit and $q-1$ ancilla qubits. After measuring the ancillae, the result either implements the target operation $T|\varphi\rangle$, or one of failure operations $R|\varphi\rangle$, to which we need to apply a cheap recovery operation $R^\dagger$. Each of the measurement outcomes implements a different operation on the target qubit. In general, measuring all ancillae as $|0\rangle$ is treated as a success.

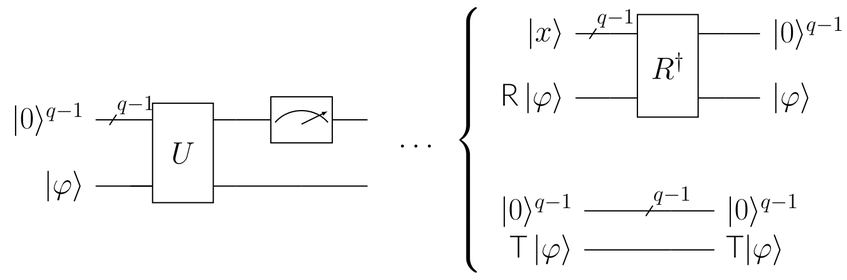

Figure 1. Full Repeat-Until-Success (RUS) protocol.

Our observations show that the RUS matrices have a very specific form and that the full RUS protocol can not be synthesized by only using $|0\rangle$, and $|1\rangle$ input states on the target qubit, and another, entangled $|+\rangle$ input state has to be introduced. This observation lead to revealing inconsistencies regarding the target operations implemented by the RUS circuits in [2].

Figure 2 shows an example of a synthesized RUS circuit that implements the operation $\frac{I+2jZ}{\sqrt{5}}$ on the target qubit. Our circuit, which minimizes gate count, shows that the state-of-the-art RUS circuits from [2] can be further optimized by our approach, achieving exact equivalence and correct implementation by smaller circuits.

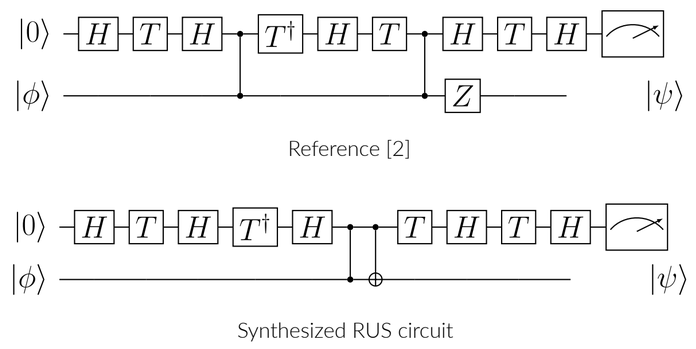

Figure 2. RUS implementations of $\frac{I+2jZ}{\sqrt{5}}$

Furthermore, Table 1 shows advancements in comparison to the reference paper [2].

	$\frac{I+j\sqrt{2}X}{\sqrt{3}}$	$\frac{I+2jZ}{\sqrt{5}}$	$R_X(arccos(\frac{3}{5}))$	$\frac{2X+\sqrt{2}Y+Z}{\sqrt{7}}$
Reference	7	12	9	13
SMT/MILP	7	11	7	11

Table 1. Gate count comparison between our approach and the reference paper for circuits implementing the specified operation.

Experimental Results

Figure 3 shows the performance of our implemented methods compared to the tool Quokka# [3]. The benchmarks consist of nearly 500 randomly generated circuits up to 6 qubits. The size of the circuits ranges from two to 18 gates.

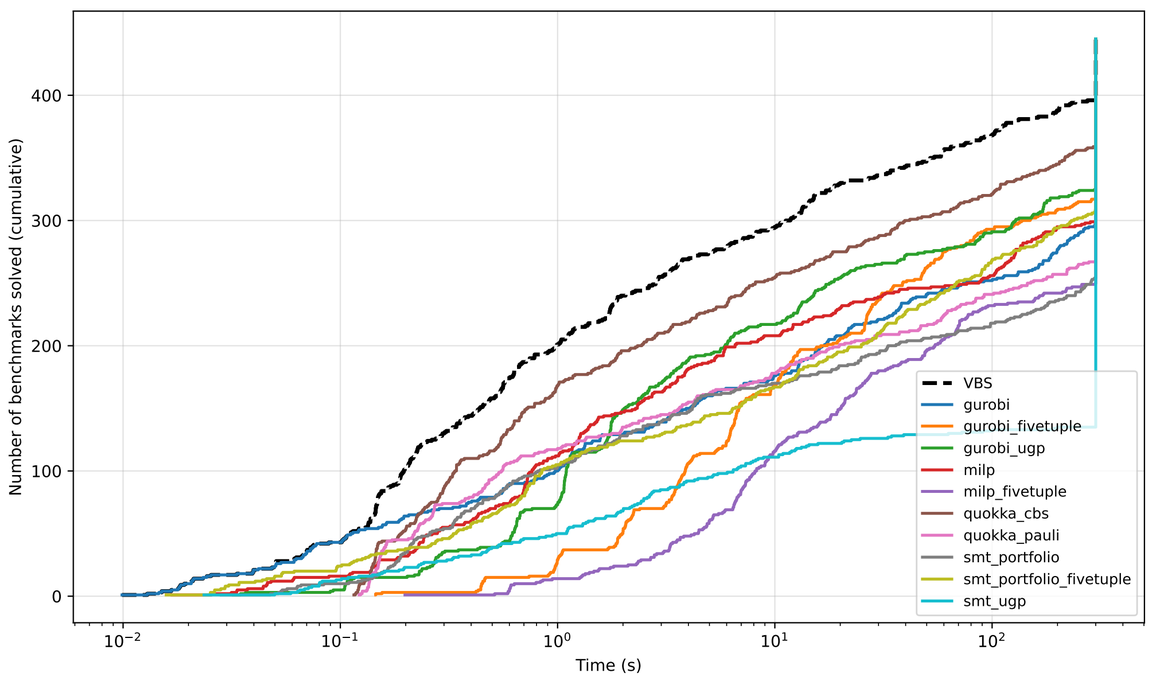

Figure 3. Comparison of our implemented methods and the tool Quokka# [3]

References

[1] Nielsen, M. A. and Chuang, I. L. *Quantum Computation and Quantum Information 10th Anniversary Edition*. Cambridge University Press, 2010. ISBN 978-1-107-00217-3.

[2] Paetznick, A. and Svore, K. M. Repeat-until-success: non-deterministic decomposition of single-qubit unitaries. *Quantum Inf. Comput.*, 2014, vol. 14, 15-16, p. 1277–1301. Available at: https://doi.org/10.26421/QIC14.15-16-2.

[3] Zak, D.; Mei, J.; Lagniez, J. and Laarman, A. Reducing Quantum Circuit Synthesis to #SAT. In: Banda, M. G. de la, ed. *31st International Conference on Principles and Practice of Constraint Programming, CP 2025, Glasgow, Scotland, August 10-15, 2025*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, vol. 340, p. 38:1–38:21. LIPIcs. Available at: https://doi.org/10.4230/LIPIcs.CP.2025.38.