Scalable Quantum Circuit Mapping for Scalable Quantum Computing Architectures

Feres Ben Fraj
Technical University Munich
fares01benfredj@gmail.com

Victor Matheke
Technical University Munich
victor.matheke@tum.de

Abstract

Quantum computing is poised to solve problems intractable for classical systems, but scalability challenges persist. This report examines the Route-Forcing algorithm [1], a novel quantum circuit mapping technique for scalable architectures. By optimizing qubit routing with attraction forces, the algorithm significantly reduces compilation time and circuit depth compared to state-of-the-art methods like SABRE, at the cost of a slight increase in SWAP gates. Furthermore, this report proposes an enhancement to the Route-Forcing algorithm, termed Sub-Architecture-Based Route-Forcing, to better address scalability in large and modular quantum systems.

1 Introduction

Quantum computers hold the promise of revolutionizing diverse fields by solving problems beyond the capabilities of classical systems. However, one critical bottleneck prevents current quantum processors from achieving their full potential: inefficient quantum circuit mapping.

The importance of quantum circuit mapping grows as architectures scale to accommodate larger numbers of qubits. Future quantum systems are expected to transition from monolithic designs to modular and multi-core architectures, as shown in Figure 1, where inter-core connections with lower fidelity introduce additional complexities. Inefficient mapping not only increases execution time but also exacerbates error rates, undermining the benefits of quantum computation.

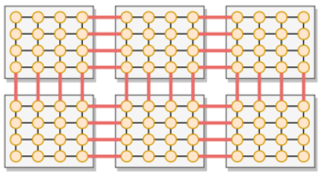
Figure 1: Multi-core architecture with short-range coupling links between cores. Black edges represent intra-core links with high fidelity, and red edges represent inter-core links with lower fidelity.

2 Background

Quantum circuit mapping is essential to ensure that algorithms comply with the physical constraints of quantum processors. It ensures that all two-qubit gates can be executed on neighboring physical qubits, aligning the quantum circuit with the processor's topology and involves two main stages:

- **Initial Placement:** Logical qubits are mapped to physical qubits to maximize direct interactions.
- **Routing of Qubits:** SWAP gates are added to connect non-neighboring qubits, enabling the execution of all operations.

This mapping problem is NP-complete, making it infeasible to find optimal solutions for large circuits. Existing methods like SABRE address this with heuristic approaches to minimize circuit depth and SWAP gates. However, these methods struggle to scale effectively for large qubit counts and modular architectures.

In this context, our work introduces *Route-Forcing*, an approach designed to overcome these limitations by focusing on efficient mapping for next-generation quantum systems. The following sections detail its design and evaluation.

3 System Design

The first task in the compilation process is to generate Directed Acyclic Graph (DAG) [2]. This Graph indicates the dependencies of operations, by creating directed edges from each operation op1 to every other operation op2 that uses the result of op1. The layer, in which an operation is found, therefore reflects the number of dependencies this operation has (meaning: the number of operations that need to be executed before this one). An operation op2 can therefore be executed once all operations op1, with an arrow from op1 to op2, have been executed. Creating a DAG from a circuit is a common concept in quantum computing and will therefore not be regarded any further in this paper.

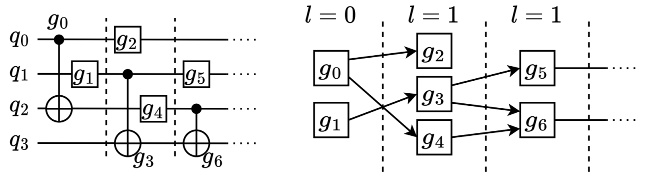

Figure 2: A DAG on the right generated from the circuit on the left.

The second task in the compilation process is to create an initial placement (mapping) of all virtual qubits onto physical ones. This placement is arbitrary, Route-Forcing can handle any mapping. The simplest mapping is a random mapping, where each virtual qubit gets placed into a random, available physical qubit.

Finally, the third task in the compilation process is the actual Route-Forcing algorithm. This step takes in the DAG and operates on the mapping created before, until all operations have been executed. This iterative process is reflected by these steps:

1. Execute all operations from the DAG that can currently be executed, and remove them from the DAG (remember: All dependencies must have been executed before, and all operands must be in adjacent physical qubits)
2. Look at all upcoming operations in the DAG and find all possible SWAP operations to prepare for each operation
   (a) Find the physical qubits for all operands of this operation. Calculate the direction vector between these physical qubits.
   (b) Look at all edges connected to these physical qubits. Calculate the swap coefficient for each of these edges based on: The attraction force, the temporal distance to the operation, the fidelity of the link and a randomness factor
3. Sort all possible SWAP operations by their coefficient and execute them in descending order, as long as they are possible
4. Continue at 1. as long as the DAG is not empty

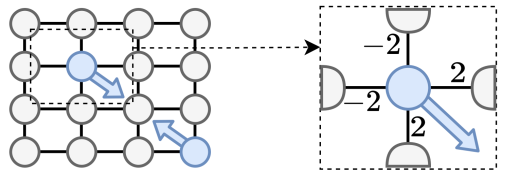

Figure 3: The qubits marked in blue represent two operands. Route-Forcing will attempt to find the best two SWAP operations to bring the operands physically closer together. The attraction force for edges pointing towards the other operand are therefore positive.

The attraction force (required in step 2.b) is a coefficient that is calculated for a edge between two physical qubits. A larger attraction force indicates an edge that is closer to the direction vector between the two operands of an operation. An example of this can be seen in Figure [3]. The attraction force is therefore modelled as the dot product between the edge direction, and the vector between the operands' qubits.

4 Evaluation

The results in Figure [4] highlight the performance of the proposed *Route-Forcing* algorithm compared to SABRE.

As the figure shows, the execution time ratio is consistently above 1, demonstrating that *Route-Forcing* achieves faster compilation times compared to SABRE—a primary objective of this work.

Significant reductions in circuit depth are observed which arises from *Route-Forcing*'s ability to add multiple parallel SWAP gates per iteration, unlike SABRE, which adds only one SWAP gate per iteration. Although this reduces the total number of SWAPs for SABRE compared to Route-Forcing, it often leads to increased circuit depth.

Notably, the speedup in compilation time and depth ratio improves with larger system sizes, suggesting that Route-Forcing's advantages become more pronounced as quantum architectures scale. Additionally, the SWAP ratio maintains a consistent average, indicating that transitioning to larger devices does not diminish Route-Forcing's performance relative to SABRE.

However, the SWAP ratio indicates that Route-Forcing does not consistently reduce the number of SWAP gates added compared to SABRE, as the ratios for SWAPs in Figure 4 are generally around or below 1. This suggests that the algorithm focuses more on optimizing depth and execution time rather than minimizing the total number of SWAPs.

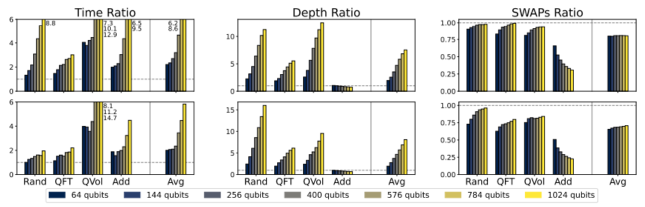

Figure 4: Performance ratio (the higher, the better, since it is computed as SABRE/Route-Forcing) of the three evaluated metrics (execution time, circuit depth, and added gates) across all tested circuits (Random Circuits, Quantum Fourier Transform, Quantum Volume, and Cuccaro Adder), and all computer sizes (from 64 to 1024 qubits in a square grid topology). The top row uses the basic heuristic, and the lower row uses the decay heuristic. Results are averaged over 50 trials.

The proposed *Route-Forcing* algorithm demonstrates significant improvements in quantum circuit mapping, achieving an average compilation time speedup of 3.7× compared to state-of-the-art scalable techniques. Additionally, it reduces the mapped circuit depth by 4.7×, although this comes at the cost of introducing 1.3× more SWAP gates. These results highlight the algorithm's effectiveness in optimizing execution time and circuit depth while maintaining manageable SWAP overhead. Furthermore, an important advantage of *Route-Forcing* lies in its ability to optimize qubit placement, particularly in multi-core quantum architectures where inter-core transfers are required to move quantum information between cores. By minimizing the need for inter-core transfers through better qubit placement, *Route-Forcing* reduces latency and error rates, enabling more direct interactions within the same core. This optimization enhances the fidelity of quantum operations, making *Route-Forcing* not only a scalable solution but also a reliable one for next-generation quantum computing systems.

5 Hybrid Approach: Sub-Architecture-Based Route-Forcing

While the proposed Route-Forcing algorithm has achieved remarkable advancements in execution time and circuit depth, it still has room for optimization, especially as quantum architectures scale. The introduction of more complex, modular, or multi-core systems poses challenges that demand even more scalable solutions.

To address these challenges, we propose an enhanced version of Route-Forcing: the Sub-Architecture-Based Route-Forcing algorithm. This approach combines the strengths of Route-Forcing with the concept of dividing large quantum architectures into smaller, manageable sub-architectures. By applying Route-Forcing locally within these sub-architectures and efficiently integrating them, we aim to achieve even greater scalability and optimization.

5.1 Background

The approach of dividing larger topologies into many small sub-architectures has been proposed in [2]. In this paper, an algorithm is introduced to find a near-optimal sub-architecture to use for n qubits. This can be used to split a complex circuit into multiple parts, which can then be mapped onto different parts of the topology, by computing a sub-architecture for each part of the circuit.

The paper focuses on finding near-optimal (as opposed to optimal) sub-architectures by applying heuristics. It first finds all non-isomorphic sub-architectures of a topology with exactly n qubits. A list of candidates (a candidate is a near-optimal sub-architecture) is build up by recursively combining smaller sub-architectures into bigger ones. This is required, since only searching the smallest possible sub-architectures may not result in the best possible solution.

Once all possible candidates have been found, a greedy algorithm is applied to find a set of candidates (of length k) which are considered equally good. Any of these candidates might therefore be chosen as the sub-architecture for an input circuit with n qubits.

For this research idea, we can use this existing algorithm to find near-optimal sub-architectures based on given connnectivity constraints. Each sub-architecture can then "host" sub-circuits on it. The mapping of this sub-circuit onto the sub-architecture is done using Route-Forcing.

5.2 Chances and Risks

The biggest chance of combining the sub-architectures approach with Route-Forcing is an additional speedup of compiler execution time. The biggest problem of mapping techniques for quantum architectures in general is the exponential growth of the search space. While Route-Forcing already combats that by using heuristics instead of an optimal search, using smaller sub-architectures obviously reduces the individual search spaces even more to combat the exponential growth.

Reducing the execution time of the mapping process can either be taken as an advantage as is, or it can in turn be used as budget for more complex mapping techniques (for example by tuning the Route-Forcing parameters). This might then improve the mapping quality while keeping at a steady exeuction time.

Whether this research idea actually leads to a speedup in the compilation process or potentially even to a slowdown must of course be evaluated in an implementation.

There are however two major risks involved in this research idea. This first one is losing the generalization of Route-Forcing. A major selling point of the proposed algorithm is that it works on any quantum architecture out-of-the-box, without needing any tuning or specific knowledge. Our research idea however requires knowledge of the (chosen) sub-architectures of the quantum architecture, which can be quite expensive to compute. Therefore, these would need to be computed once, ahead of time for an architecture. Logically, this would then increase the effort required to get Route-Forcing up and running for a new architecture.

A second risk is the decrease in quality of the generated mapping. Finding optimal sub-architectures has been proven impractical (as described in). Additionally, connecting the different sub-circuits back together afterwards might require a lot of SWAPs. In general, it seems plausible that doing one mapping on the global circuit would lead to a more optimal output than doing these indivual mappings.

All of the listed chances and risks were thought of in theory. Whether they actually hold true in practice must be checked against a working implementation.

5.3 Implementation

Our implementation divides large quantum architectures into smaller sub-architectures and maps each independently using *Route-Forcing*. These locally optimized mappings are then reintegrated into a global mapping that maintains overall coherence. Below is the full pseudocode that outlines how the hybrid algorithm operates:

Algorithm 1 Subarchitecture Route-Forcing (SARF)

```
Require: Quantum Circuit $C$, Topology $T$, Subarchitecture Size $k$
Ensure: Fully Mapped Circuit $C_{mapped}$
  Step 1: Subarchitecture Generation
  /* Generate all possible subarchs of size k */
  $Subarchs \leftarrow generate\_non\_isomorphic\_subarchs(T,k)$
  /* Select the most connected subarchs */
  $BestSubarchs \leftarrow greedy\_select\_subarchs(Subarchs, C.get\_qubits().size())$
  Step 2: Circuit Partitioning
  /* Divide the circuit into subcircuits based on the selected subarchs */
  $Subcircuits, Remaining \leftarrow partition\_circuit(BestSubarchs, C)$
  /* Remaining gates that cannot fit into any subarch are kept for global mapping */
  Step 3: Local Mapping
  $MappedSubcircuits \leftarrow []$
  for each $(Subarch, Subcircuit)$ in $(BestSubarchs, Subcircuits)$ do
      /* Initialize a mapping for the current subarchitecture */
      $Mapping \leftarrow get\_initial\_mapping(Subarch.get\_qubits(), Subcircuit.get\_qubits())$
      while not $Subcircuit.count\_unexecuted\_gates() == 0$ do
          /* Identify gates that can be executed immediately */
          $ExecutableGates \leftarrow Subcircuit.get\_executable\_gates(Subarch, Mapping)$
          Append $ExecutableGates$ to $MappedSubcircuits$
          /* Calculate possible SWAPs to enable unexecuted gates */
          $SwapCosts \leftarrow calculate\_all\_swap\_coefficients(Subarch, Mapping, Subcircuit)$
          if not $SwapCosts.isEmpty()$ then
              /* Select the SWAPs with the highest coefficients */
              $BestSwaps \leftarrow select\_best\_swaps(SwapCosts)$
              /* Update the mapping with the selected SWAPs */
              Update $Mapping$ with $BestSwaps$
          end if
      end while
  end for
  Step 4: Global Integration
  /* Merge all mapped subcircuits */
  $GlobalCircuit \leftarrow integrate\_subcircuits(MappedSubcircuits, Remaining, T)$
  /* Handle any remaining gates that span across subarchitectures */
  Map remaining gates in $Remaining$ using Route-Forcing
      return $GlobalCircuit$ as $C_{mapped}$
```

5.4 Results

Unfortunately, the proposed Route-Forcing is not guaranteed to converge for all possible circuits onto all possible topologies (see [1], pages 5-6). Therefore, the testing capabilities for our algorithm is limited. However, testing it on small topologies has provided valuable insights into its behavior.

Below, we present the results from a 10-qubit setup:

Method	Execution Time (s)	Number of SWAPs
RF	1.508	3
SARF	3.522	2

These results demonstrate the trade-offs of our approach:

- For small architectures, SARF optimizes the number of SWAP gates compared to RF, reducing them from 3 to 2, which is critical for improving quantum circuit efficiency.
- However, SARF requires more than double the execution time of RF, primarily due to the overhead of dynamically generating subarchitectures and mapping them.

At this stage, the research idea does not seem as promising as initially hoped. However, the implementation is incomplete, and more work is needed to fully evaluate its potential. A potential optimization to address this issue would involve **pre-calculating subarchitectures** before execution. By reducing the on-the-fly computation of subarchitectures, the algorithm could achieve faster runtime, particularly for larger setups.

6 Conclusion and Future Work

This paper has summarized the Route-Forcing algorithm, as it was proposed in [1]. Afterwards, the paper has proposed, implemented, and evaluated an original research idea, based on a different paper [2].

Route-Forcing is a quantum circuit mapping technique, created as one of the first scalable compilation techniques. It exhibits much faster compilation times, as well as reduced circuit depth compared to the existing state-of-the-art techniques, at the cost of slightly increased number of generated SWAPs. This makes Route-Forcing an important milestone in the progress of growing quantum computer architectures.

The proposed idea combines the Route-Forcing algorithm with the subdivision approach to improve quantum circuit mapping by splitting larger architectures into smaller sub-architectures and applying Route-Forcing locally. While the implementation demonstrated potential in reducing SWAP gates for small topologies, it struggled with execution time and scalability to larger architectures.

Future work should focus on optimizing this approach to address its current limitations. Key improvements include pre-computation of subarchitectures to eliminate runtime overhead, refining the process of reconnecting subcircuits to enhance scalability and performance, and extending the implementation to evaluate its practicality on larger and more complex topologies.

Although the research idea is not yet fully realized, these enhancements could make it a viable solution for scalable quantum circuit mapping. Further work is required to fully evaluate its potential and identify the most effective optimizations.

References

[1] Pau Escofet, Alejandro Gonzalvo, Eduard Alarcón, Carmen G Almudéver, and Sergi Abadal. Route-forcing: Scalable quantum circuit mapping for scalable quantum computing architectures. *arXiv preprint arXiv:2407.17306*, 2024.

[2] Tom Peham, Lukas Burgholzer, and Robert Wille. On optimal subarchitectures for quantum circuit mapping. *ACM Transactions on Quantum Computing*, 4(4):1–20, 2023.