LONDON'S GLOBAL UNIVERSITY
Department of Computer Science

Dynamic Circuit Mapping Algorithms for Networked Quantum Computers

Natasha Valentina Santoso

23223788

Supervisors:

Abhishek Agarwal (NPL), Lachlan Lindoy (NPL)

A project report presented in partial fulfillment of the degree

MSc Quantum Technologies

Department of Physics and Astronomy

University College London

August 2024

Declaration

I, Natasha Valentina Santoso, declare that the thesis has been composed by myself and that the work has not be submitted for any other degree or professional qualification. I confirm that the work submitted is my own. The report may be freely copied and distributed provided the source is explicitly acknowledged.

[illegible]

Natasha Valentina Santoso	22 August 2024
Signature	*Date*

Table of Contents

List of Tables

List of Algorithms

Acronyms

DAG Directed Acyclic Graph. xii, xvi, xvii, xxi, 35, 36

DQC Distributed quantum computing. 1

ECR Echoed Cross-Resonance. 3

ISA Instruction Set Architecture. 6, 7

QBN Qubit Interaction Neighborhood. xii, xiv

QPI Qubit Pair Interaction. v, xii, xiii, xiv, 35

QPU Quantum Processing Unit. 1, 3, ix, 40

3 | Results and Analysis

3.1 Configurations

The evaluation is configured as follows:

1. Metrics: The evaluation focuses on two main metrics, the total number of additional swap gates (g) and the resulting circuit depth (d) after transpilation.
2. Experiment Software: The algorithm is implemented in Python version 3.10.12 with IBM Qiskit framework. The code is available on Github `https://github.com/natashaval/qubit-mapping-distributed-qc`.
3. Experiment Hardware: The tests were conducted on a standard personal computer equipped with an Intel i7-8700 CPU and 16 GB of RAM.
4. Layout: The algorithms were tested using 20 qubits across the layouts detailed in Table 3.1. The layout names follow the format: *<layout name>*_ *<number of qubits>*_ *<number of groups>*. Each layout is a symmetric coupling graph, enabling bidirectional operation of two-qubit gates between any pair of connected physical qubits. Coupling graphs are illustrated in Appendix B.

Table 3.1: Number of qubits and groups for layouts

Layout	**Number of qubits**	**Number of groups**
Full	20	1
Full	10	2
Grid	9	2
Ring	10	2
Full	7	3
Grid	8	3
Ring	7	3
Full	5	4
Grid	6	4
Ring	5	4
T Horizontal	5	4
T Vertical	5	4
Line	1	20

5. Benchmarks: The benchmarks listed in Table 3.2 are taken from MQTBench [1]. The quantum circuits are composed of native gates from the Qiskit library and are represented in the QASM 2.0 language.

Table 3.2: Circuit size and circuit depth of benchmark algorithms for 5, 10, and 15 qubits

Algorithm	**n = 5**		**n = 10**		**n = 15**	
	gate	depth	gate	depth	gate	depth
ghz	7	7	12	12	17	17
dj	36	11	79	17	118	22
graphstate	50	22	100	26	150	29
qft	71	38	270	78	591	118
wstate	73	45	163	90	253	135
qftentangled	78	42	282	82	608	122
vqe	83	21	168	26	253	31
qaoa	95	31	190	34	285	34
realamprandom	130	37	335	57	615	77
twolocalrandom	130	37	335	57	615	77
su2random	150	41	375	61	675	81
qnn	154	58	459	108	914	158
portfolioqaoa	195	72	615	132	1260	192
random	223	97	646	155	1992	412
portfoliovqe	310	107	1145	217	2505	327

3.2 Evaluation

The data is processed from the table in the appendix E.

3.2.1 Distribution of Additional Swap Gates and Circuit Depth

The box plot in Figure 3.1 compares the average number of additional swap gates required by three different strategies, namely `BasicSwap`, `SabreSwap`, and `LookaheadSwap`, across various layout configurations. `BasicSwap` [2] strategy employs a brute force approach, adding one or more swap gates whenever a connectivity constraint is encountered, iterating through each layer until the circuit is compatible. The `SabreSwap` strategy uses a heuristic search to minimize the number of swap gates and reduce circuit depth [3], [4]. Meanwhile, the `LookaheadSwap` strategy combines elements from both algorithms (Algorithm 1 and 2) that has been described in the previous section. A lower number of swap gates or fewer circuit depth indicates better strategy performance.

The `BasicSwap` consistently exhibits the highest number of swap gates across all layouts, with particularly high values exceeding 5000 additional swap gates in layouts such as *ring_10_2*, *t_horizontal_5_4*, *t_vertical_5_4*, and performing worst in the *line_1_20* layout. The large spread and height of the boxes indicate both poor and inconsistent performance. In contrast, `SabreSwap` performs significantly better with the range of around

1000 additional swap gates for all layouts. `LookaheadSwap` performs almost as well as `SabreSwap`, with the best results seen in the *full_ 7_ 3* layout. However, in some cases like *full_ 5_ 4* and *ring_ 5_ 4*, the `LookaheadSwap` strategy has exceptionally low values that are likely outliers due to timeouts, which may skew the perception of its effectiveness in these instances.

Overall, the layouts that show better performance, as indicated by lower swap gate counts, are *grid_ 8_ 3*, *grid_ 6_ 4*, *full_ 10_ 2* and *full_ 7_ 3*. These results suggest that *grid* and *full* layouts are generally more efficient than other configurations. It is also important to note that *full_ 20_ 1* and *line_ 1_ 20* serve as baselines; *full_ 20_ 1* represents a maximally connected coupling map and is expected to deliver the best performance, while *line_ 1_ 20*, being the least connected, is likely to show the poorest performance.

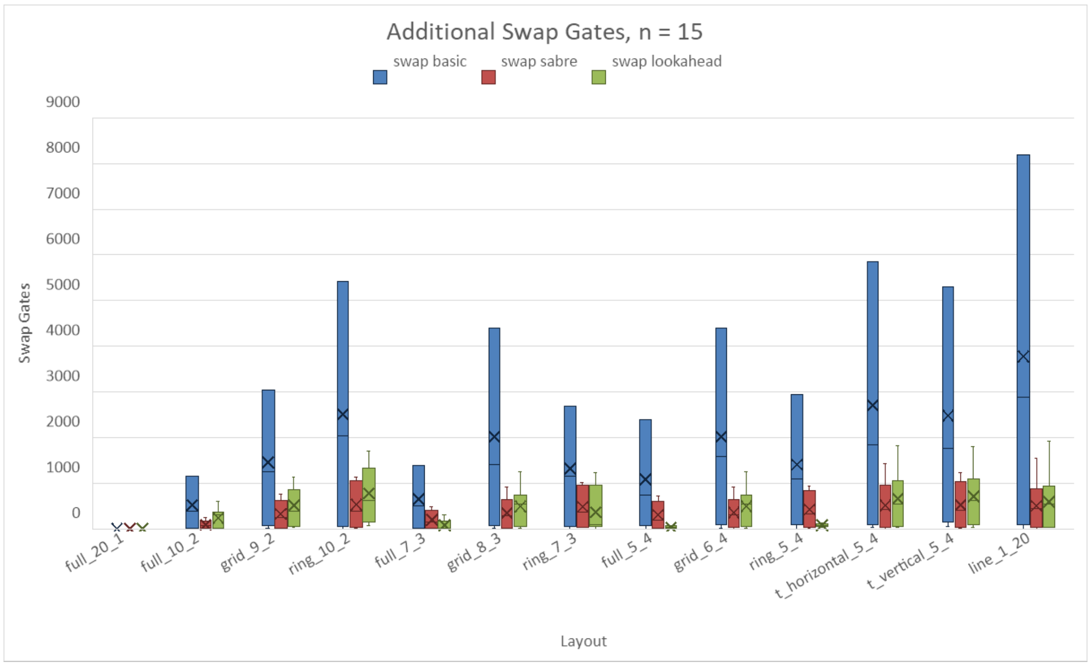


Figure 3.1: Box plot additional swap gates for a circuit size of 15

Similarly, the analysis of quantum circuit depths across various layouts, as shown in Figure 3.2, reveals significant differences in the effectiveness of the three optimization methods. The depths associated with `BasicSwap` consistently result in the highest circuit depths often exceeding 1000, and in some cases approach 2000 or more in layouts like *line_ 1_ 20*, *t_ horizontal_ 5_ 4*, and *ring_ 10_ 2*. In contrast, `SabreSwap` and `LookaheadSwap` both achieve significantly shallower circuit depths, generally clustering below 500. How-

ever, some layouts, such as *full_ 5_ 4* and *ring_ 5_ 4* show occasional outliers.
Overall, the `SabreSwap` and `LookaheadSwap` strategies perform comparably well, with the latter slightly outperforming `SabreSwap` in certain configurations. The most efficient layouts, characterized by lower circuit depths, include *full_ 7_ 3* and *grid_ 6_ 4*, where both `SabreSwap` and `LookaheadSwap` consistently maintain minimal circuit depth. This suggests that for minimizing circuit depth, `LookaheadSwap` is generally the most effective strategy, closely followed by `SabreSwap`.

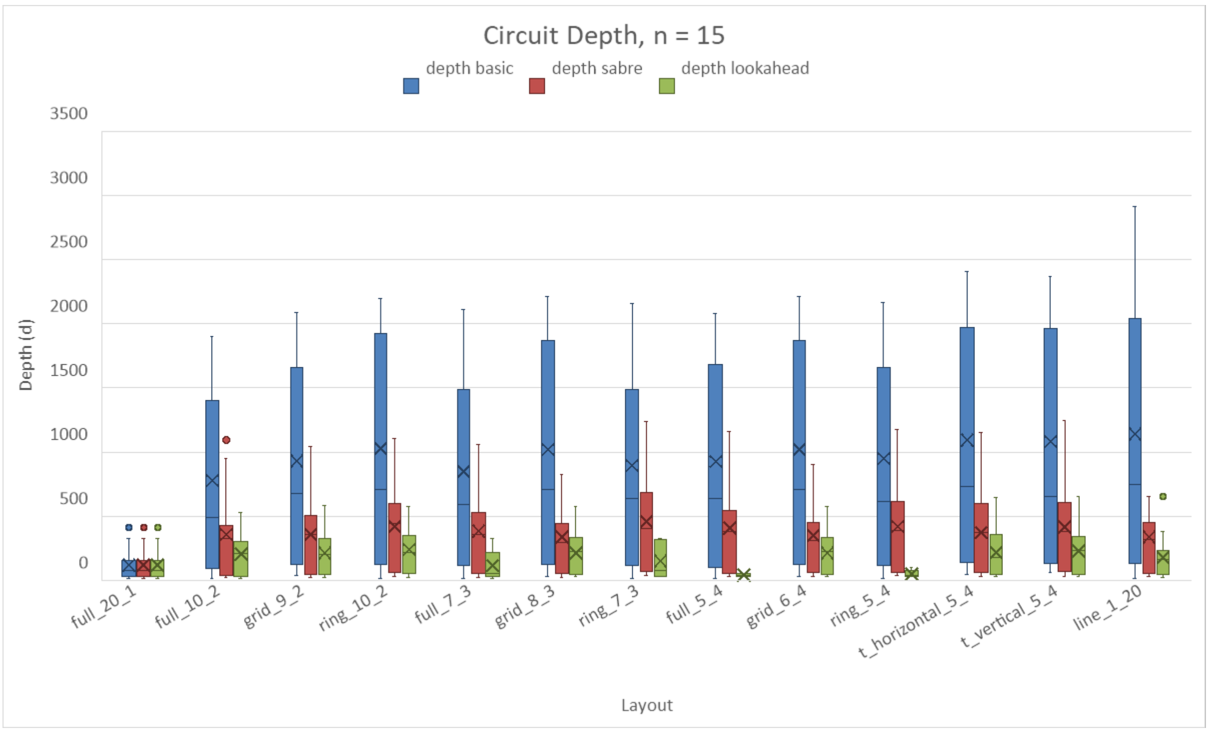
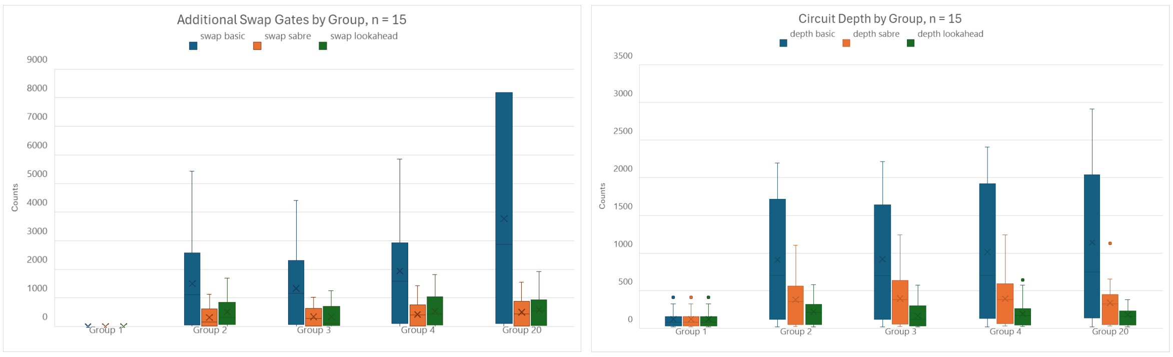


Figure 3.2: Box plot circuit depth for a circuit size of 15

The box plots present the additional swap gates 3.3a and circuit depth 3.3b categorized by group for a circuit size of 15. Group 1 and Group 20 serve as baselines, with Group 1 representing a fully connected graph and Group 20 a line coupling graph. Among other groups, Group 2 stands out as the best performer. In Group 2, the `LookaheadSwap` strategy performs particularly well, showing very low swap gate counts and moderate circuit depths, reflecting consistent and efficient performance. Group 3, while similar to Group 2, displays slightly more variability in circuit depths and a minor increase in swap gate counts, particularly with the `BasicSwap` strategy, though the `LookaheadSwap` approach still outperforms the others. Group 4, on the other hand, shows the most variability, especially with the `BasicSwap` strategy, which has a much higher median and a widespread, suggesting inconsistent performance. Although the `LookaheadSwap` strategy in Group 4 still achieves the lowest swap gate counts and circuit depths, its performance is less consistent compared to Groups 2 and 3. Therefore, Group 2 emerges as the top performer in minimizing swap gates and lowering circuit depths with the `LookaheadSwap` strategy, followed by Group 3, while Group 4 shows the least favourable results.



(a) Box Plot Additional Swap Gates by Group (b) Box Plot Circuit Depth by Group

Figure 3.3: Swap strategies categorized in group for a circuit size of 15

3.2.2 The Total Number of Algorithms Successfully Run for Each Layout

The previous section mentioned that the algorithms for the *full_ 5_ 4* and *ring_ 5_ 4* layouts show outliers, indicating that they do not execute all the algorithms. Therefore, the chart in Figure 3.4 specifies the total number of successful algorithms run across different layouts, focusing on circuit sizes ranging from 5, 10, and 15 qubits, with a total of 15 algorithms per round. Most layouts perform consistently well, achieving the maximum count of 15 successful runs per benchmark.

One thing to notice is that *full_ 7_ 3* and *full_ 5_ 4* layouts fail to complete the benchmarks for the largest circuit size of 15 qubits, with *full_ 7_ 3* achieving only 9 successful runs and *full_ 5_ 4* barely completing 3 runs. Additionally, *ring_ 7_ 3* and *ring_ 5_ 4* layouts also encounter difficulties, particularly as the circuit size increases beyond 10 qubits. *Ring_ 7_ 3* can only achieve 10 and 8 successful runs for circuit sizes 10 and 15, while *ring_ 5_ 4* struggles even at the 5-qubit benchmark, with its performance worsening as the circuit size increases. The failed algorithms and layouts are listed in Appendix D.

These variations suggest that certain layouts are more robust across different run sizes, while others, particularly *ring_7_3* and *ring_5_4* face challenges with medium run sizes, indicating possible inefficiencies or challenges in these configurations. In general, the layouts in groups 1 and 2 are more consistent across different run sizes, while those in groups 3 and 4 tend to show more variability.

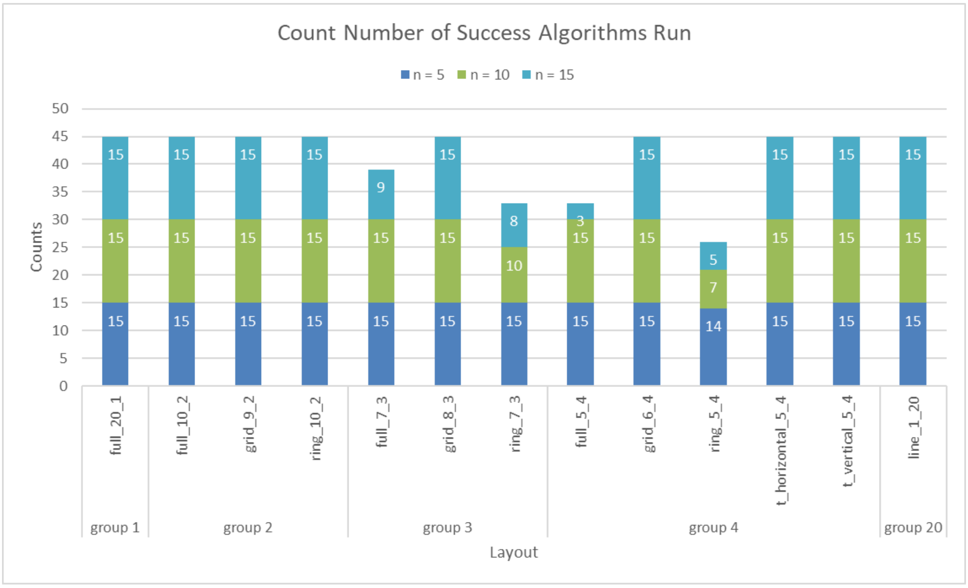
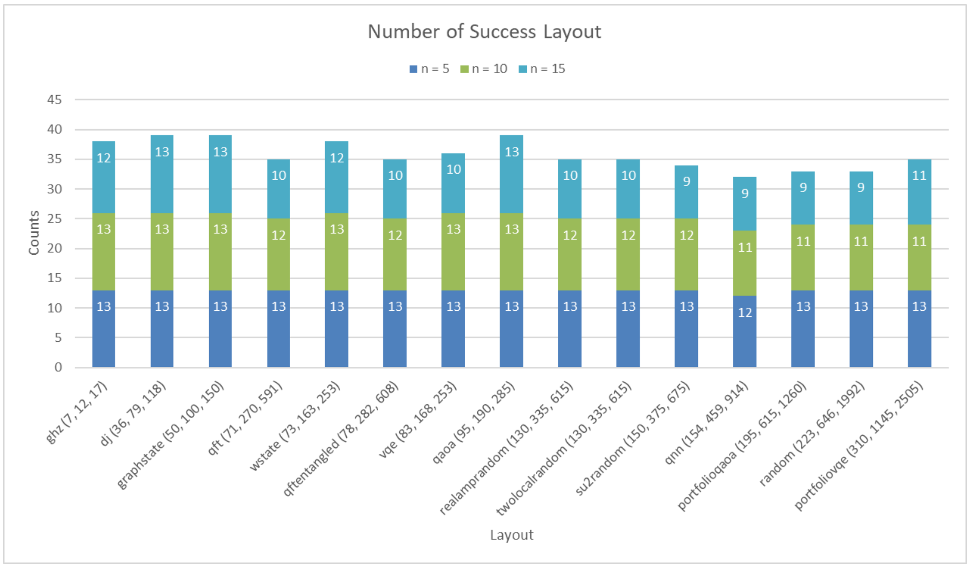


Figure 3.4: Total count of successful algorithm runs across various layouts, $n = 15$

In addition, the chart in Figure 3.5 highlights the total count of successful layout executions across different algorithms. Most algorithms like "dj", "graphstate" and "qaoa" maintain consistent performance across all benchmark sizes with 100% success rate. However, some algorithms, like "qft", "qftentangled", "realamprandom", and "twolocalrandom", show slightly reduced success rates, particularly for $n = 10$, where the count drops to 12. The algorithms "qnn" and "random" display even more pronounced variations, with success rates for $n = 10$ and $n = 15$ decreasing to 11 and 9, respectively.

While most algorithms perform reliably across all benchmarks, a few show inconsistencies, especially with $n = 10$ and $n = 15$, suggesting that certain algorithms may struggle with larger problem sizes.



Layout	n = 5	n = 10	n = 15
ghz (7, 12, 17)	13	13	12
dj (36, 79, 118)	13	13	13
graphstate (50, 100, 150)	13	13	13
qft (71, 270, 591)	13	12	10
wstate (73, 163, 253)	13	13	12
qftentangled (78, 282, 608)	13	12	10
vqe (83, 168, 253)	13	13	10
qaoa (95, 190, 285)	13	13	13
realamprandom (130, 335, 615)	13	12	10
twolocalrandom (130, 335, 615)	13	12	10
su2random (150, 375, 675)	13	12	9
qnn (154, 459, 914)	12	11	9
portfolioqaoa (195, 615, 1260)	13	11	9
random (223, 646, 1992)	13	11	9
portfoliovqe (310, 1145, 2505)	13	11	11

Figure 3.5: Total count of successful layouts across different algorithms

3.2.3 Layouts for Algorithms

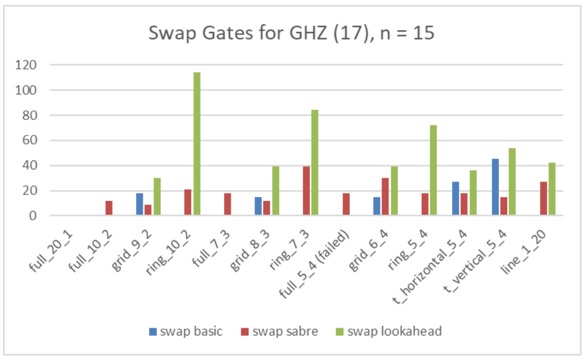
In this section, a selection of representative algorithms - GHZ, Deutsch-Jozsa, Graph State, VQE, and Portfolio QAOA - are analyzed and compared across various layouts. The baseline values for the total circuit gates and circuit depth are provided in parentheses, and the results discussed are based on using 15 qubits. This comparison highlights how different layouts perform when subjected to different swap routing methods.

3.2.4 GHZ

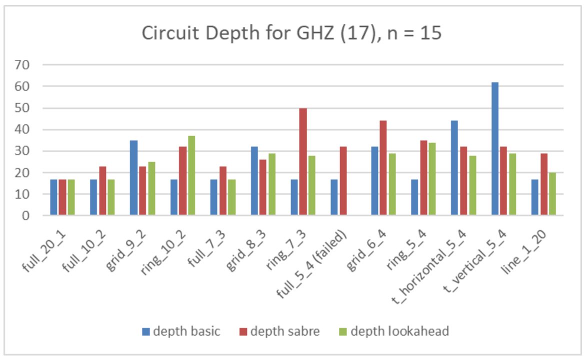
The GHZ charts compare the number of additional swap gates needed for different layouts (Figure 3.6a) and the resulting circuit depth (Figure 3.6b) when running with 17 qubits. It shows that the `BasicSwap` strategy generally performs best, requiring the fewest swaps, particularly in all full, ring, and line layouts, where it results in 0 additional swap gates. In this context, `SabreSwap` tend to perform worse than `BasicSwap`, while the `LookaheadSwap` strategy performs significantly worse than the others, with a notable spike in the *ring_10_2* layout. Regarding circuit depth, the results for `LookaheadSwap` are nearly comparable to those of `SabreSwap`, with both strategies yielding a circuit depth of approximately 30.

3.2.5 Deutsch-Jozsa

The charts for running the Deutsch-Jozsa algorithm with 118 gates (Figure 3.7a) and a circuit depth of 22 (Figure 3.7b) show that the `SabreSwap` and `LookaheadSwap` strategies

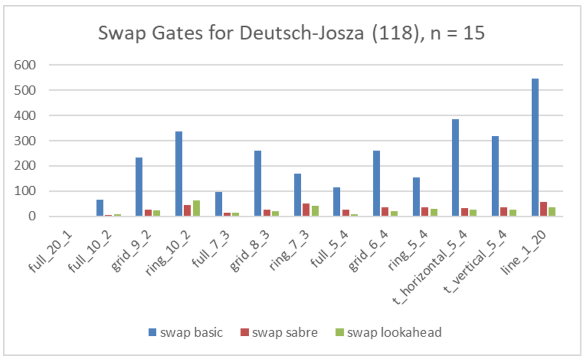


(a) Additional swap gates for GHZ

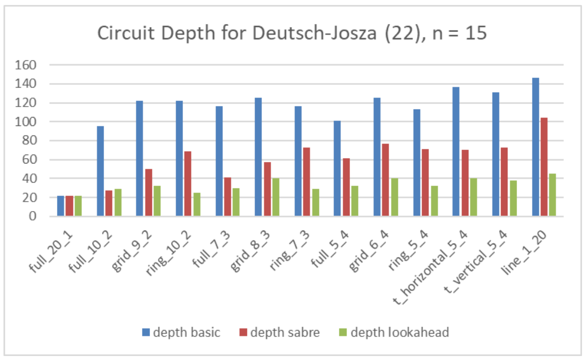


(b) Circuit depth for GHZ

consistently outperform the `BasicSwap` strategy in minimizing swap gates. The *line_ 20_ 1* layout performs poorly, with the `BasicSwap` method leading to an extremely high number of additional swap gates, exceeding 500. Although the additional swap gates for `SabreSwap` and `LookaheadSwap` are similar across all layouts, remaining below 50, the circuit depth with `LookaheadSwap` is significantly lower, staying under 40 for all layouts, compared to `SabreSwap`, which is around 60.



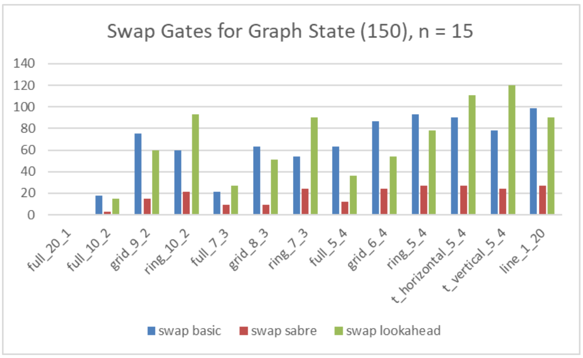
(a) Additional swap gates for Deutsch-Jozsa



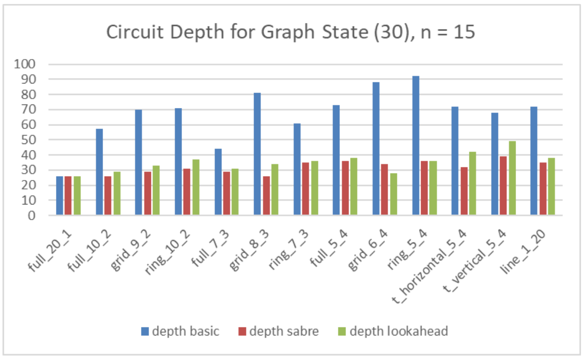
(b) Circuit depth for Deutsch-Jozsa

3.2.6 Graph State

The Graph State charts compare additional swap gates with an initial count of 150 gates (Figure 3.8a) and a circuit depth of 30 (Figure 3.8b) across different layouts. In this algorithm, the performance of `LookaheadSwap` is similar to `BasicSwap`, with the poorest performance observed in the *t_ vertical_ 5_ 4* layout, which requires 120 additional swap gates. The `SabreSwap` strategy, on the other hand, results in significantly fewer swap gates, with counts around 20. Despite the difference in additional swap gates between `SabreSwap` and `LookaheadSwap` being as much as 200%, the resulting circuit depth is almost comparable, ranging from 30 to 40 across all layouts.



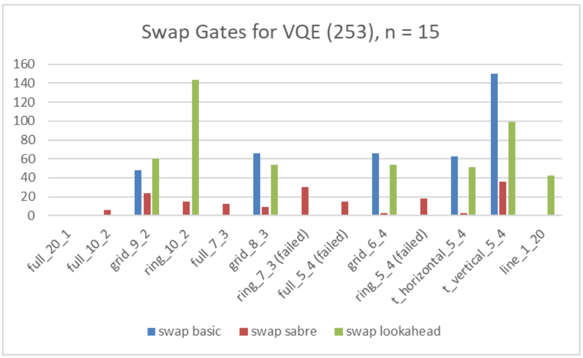
(a) Additional swap gates for Graph State



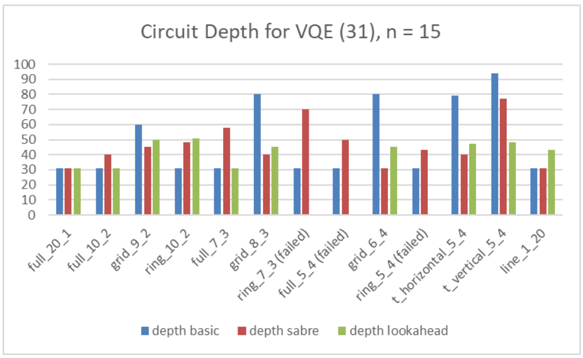
(b) Circuit depth for Graph State

3.2.7 VQE

The chart illustrates that for the VQE algorithm with 253 gates (Figure 3.9a), the `SabreSwap` strategy generally performs best across most layouts, particularly in *grid_8_3*, *grid_6_4* and *t_horizontal_5_4*, where it minimizes the number of additional swap gates. In contrast, the `LookaheadSwap` strategy tends to require significantly more swaps, especially in layouts like *ring_10_2* and *t_vertical_5_4*. The `BasicSwap` strategy shows similar behaviour to `LookaheadSwap`, with the exception of the *t_vertical_5_4* layout, where it exceeds 140 additional swap gates. When examining circuit depth (Figure 3.9b), `BasicSwap` does not show much difference from the baseline, remaining at 31. Meanwhile, the circuit depth performance of `LookaheadSwap` is comparable to that of `SabreSwap`, with layouts like *full_7_3* and *t_vertical_5_4* delivering significantly better results, with a reduction of nearly 40%.



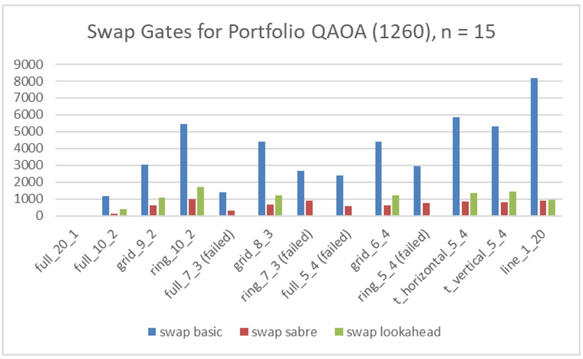
(a) Additional swap gates for VQE



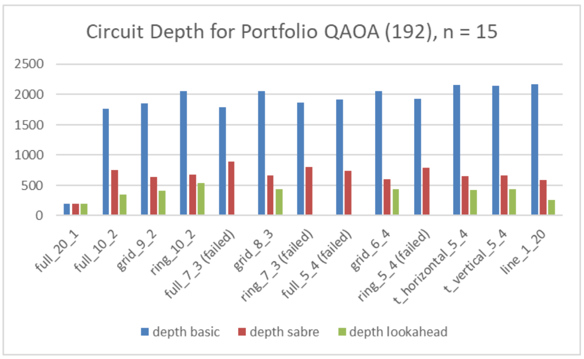
(b) Circuit depth for VQE

3.2.8 Portfolio QAOA

These final charts serve as representatives for various other algorithms listed in Table 3.2, ranging from "qft" to "portfoliovqe". The charts compare the performance of swap strategies for the Portfolio QAOA algorithm with 1260 gates (Figure 3.10a) and a circuit depth of 192 (Figure 3.10b). The `BasicSwap` strategy performs poorly, especially in layouts like *ring_10_2*, *t_horizontal_5_4*, and *line_1_20* with swap counts exceeding 5000 in some cases. On the other hand, both `SabreSwap` and `LookaheadSwap` significantly reduce the number of swap gates across most layouts to just below 2000, with `SabreSwap` generally requiring fewer swaps than `LookaheadSwap`. The second chart showing circuit depth follows a similar pattern: `BasicSwap` results in much deeper circuits, often exceeding 2000 in depth, particularly in the same problematic layouts. `SabreSwap` and `LookaheadSwap` again perform better, with `SabreSwap` consistently achieving lower circuit depths around 700, closely followed by `LookaheadSwap` just below 500.



(a) Additional swap gates for Portfolio QAOA



(b) Circuit depth for Portfolio QAOA

References

[1] N. Quetschlich, L. Burgholzer, and R. Wille, “MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing,” *Quantum*, vol. 7, p. 1062, Jul. 2023, arXiv:2204.13719 [quant-ph], ISSN: 2521-327X. DOI: 10.22331/q-2023-07-20-1062.

[2] IBMQuantum, *BasicSwap*, en. [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.transpiler.passes.BasicSwap (visited on 07/31/2024).

[3] G. Li, Y. Ding, and Y. Xie, “Tackling the qubit mapping problem for NISQ-era quantum devices,” in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, Providence RI USA: ACM, Apr. 4, 2019, pp. 1001–1014, ISBN: 978-1-4503-6240-5. DOI: 10.1145/3297858.3304023.

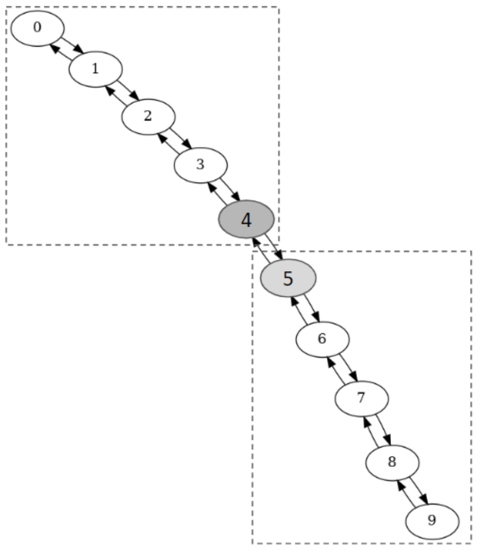
[4] IBMQuantum, *SabreSwap*, en. [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.transpiler.passes.SabreSwap (visited on 07/31/2024).

A | Source Code

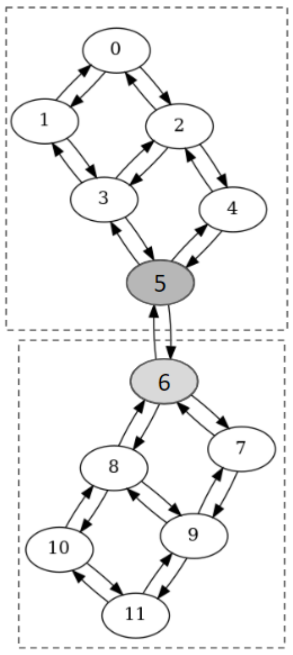
Source code for all of the methods implemented in Chap. 2 for the project can be found in the GitHub repository:

https://github.com/natashaval/qubit-mapping-distributed-qc.

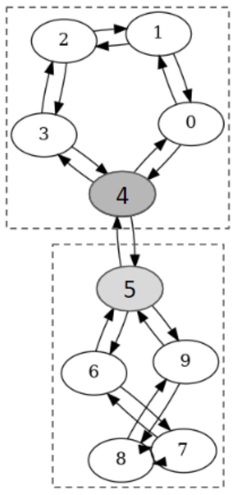
B | Coupling Graph by Group



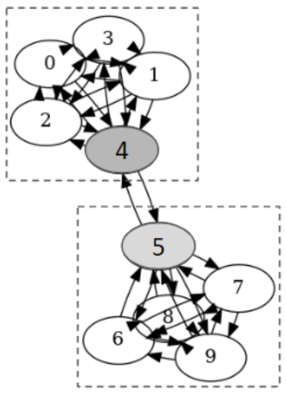
(a) Coupling graph line_10_2



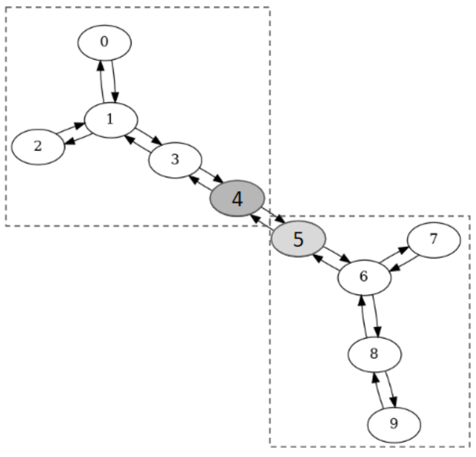
(b) Coupling graph grid_6_2



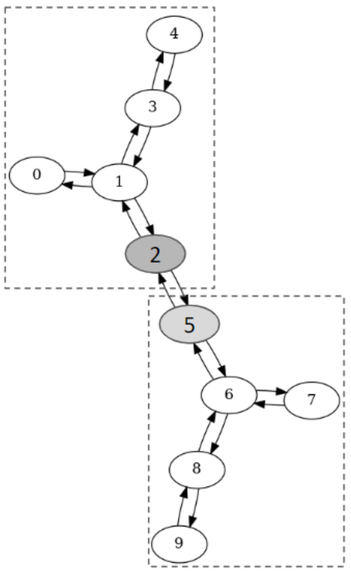
(c) Coupling graph ring_5_2



(d) Coupling graph full_5_2



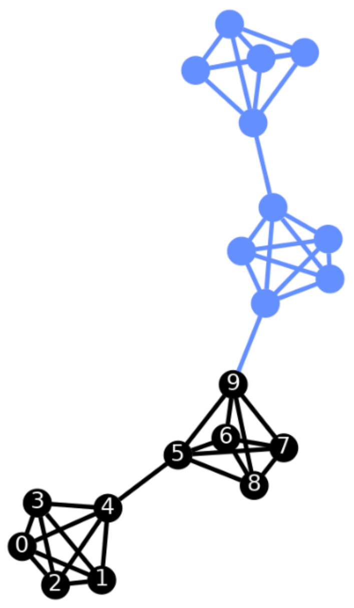
(e) Coupling graph t_horizontal_5_2



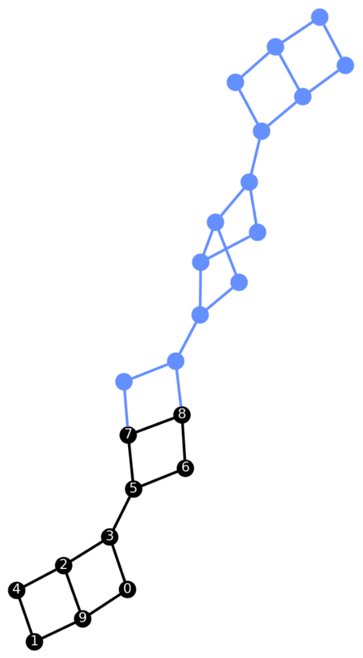
(f) Coupling graph t_vertical_5_2

Figure B.1: Generate group coupling graph

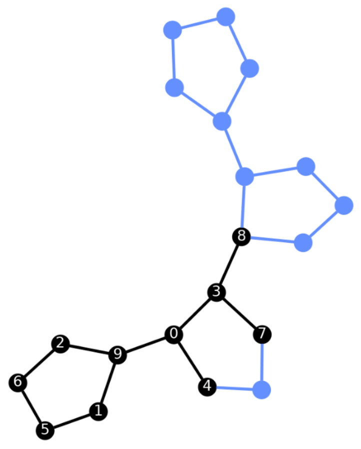
C | Qubit Mapping Illustration



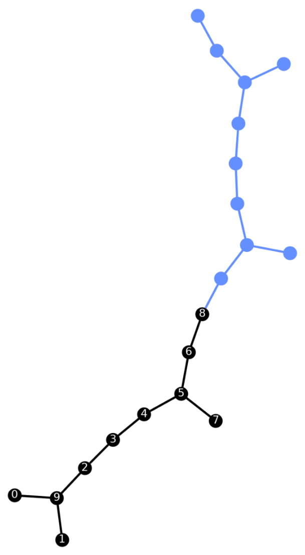
(a) full_5_4 layout



(b) grid_6_4 layout



(c) ring_5_4 layout



(d) t_horizontal_5_4 layout

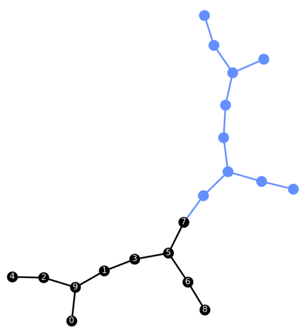

(e) t_horizontal_5_4 layout

Figure C.1: Deutsch-Jozsa n = 10 in coupling map. Black nodes with numbers illustrate the location of logical qubits, and blue nodes illustrate the available physical qubits.

D | Failed Algorithms by Layout and Group

Table D.1 lists failed algorithms of 10 and 15 qubits, showing the layouts where timeout occured.

Table D.1: Failed algorithms grouped by circuit size

Algorithms	**n = 10**		**n = 15**	
	mapping timeout	swap timeout	mapping timeout	swap timeout
ghz			full_5_4	
qft		ring_7_3 ring_5_4	full_5_4	
wstate			full_5_4	
qftentangled		ring_7_3	full_5_4	ring_7_3 ring_5_4
vqe		ring_7_3 ring_5_4	full_5_4	
twolocalrandom		ring_5_4	full_5_4	full_7_3 ring_5_4
su2random		ring_5_4	full_5_4	full_7_3 ring_5_4
qnn		ring_7_3 ring_5_4	full_5_4	full_7_3 ring_7_3 ring_5_4
portfolioqaoa		ring_7_3 ring_5_4	full_5_4	full_7_3 ring_7_3 ring_5_4
random		ring_7_3 ring_5_4	full_5_4	full_7_3 ring_7_3 ring_5_4
portfoliovqe		ring_7_3 ring_5_4	full_5_4	full_7_3 ring_7_3 ring_5_4

E | Benchmark Result Table by Group

These tables present the benchmark results for various layouts running different algorithms on circuit sizes of 5, 10, and 15 qubits.
layout: layout configuration, where the first number indicates the number of qubits and the second number represents the number of groups.
benchmark: benchmark algorithms used for testing
g: total number of gates
d: circuit depth
s_B: total additional swap gates for "swap basic"
s_S: total additional swap gates for "swap sabre"
s_L: total additional swap gates for "swap lookahead"
Δs_B: gate difference between "swap basic-lookahead" (%)
Δs_S: gate difference between "swap sabre-lookahead" (%)
d_B: circuit depth for "swap basic"
d_S: circuit depth for "swap sabre"
d_L: circuit depth for "swap lookahead"
Δd_B: depth difference between "swap basic-lookahead" (%)
Δd_S: depth difference between "swap sabre-lookahead" (%)