A doubly stochastic matrices-based approach to optimal qubit routing

Nicola Mariella 1 · Sergiy Zhuk 1

Received: 13 March 2023 / Accepted: 10 June 2023 / Published online: 29 June 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Swap mapping is a quantum compiler optimization that, by introducing SWAP gates, maps a logical quantum circuit to an equivalent physically implementable one. The physical implementability of a circuit is determined by the fulfillment of the hardware connectivity constraints. Therefore, the placement of the SWAP gates can be interpreted as a discrete optimization process. In this work, we employ a structure called doubly stochastic matrix, which is defined as a convex combination of permutation matrices. The intuition is that of making the decision process smooth. Doubly stochastic matrices are contained in the Birkhoff polytope, in which the vertices represent single permutation matrices. In essence, the algorithm uses smooth constrained optimization to slide along the edges of the polytope toward the potential solutions on the vertices. In the experiments, we show that the proposed algorithm, at the cost of additional computation time, can deliver significant depth reduction when compared to the state-of-the-art algorithm SABRE.

Keywords Quantum circuits · Quantum SWAP mapping · Quantum circuit compilers · Quantum circuit optimization

1 Introduction

In the quantum computing field, the qubit routing procedure is a fundamental component of the quantum compiler. Its role is that of mapping logical qubits to physical ones while preserving the resulting unitary (up to a permutation) and fulfilling the connectivity constraints of the target hardware. The constraints consist of the set of pairs of physical qubits upon which a two-qubit gate can be applied. The ability of the
compiler of performing the qubit routing efficiently is particularly important for the current state-of-the-art quantum technology. The latter being commonly denoted as Noisy Intermediate-Scale Quantum (NISQ) [1]. Hardware devices that can implement circuits intractable classically are for example those produced by Google [2] and IBM [3]. The reason for the significance of the qubit routing lies in the fact that the present quantum technology is characterized, in general, by reduced qubits connectivity and coherence time. Consequently, the circuit depth increase that the routing solution may determine, produces additional noise.

The method for solving the routing problem considered in this work is called swap mapping [4, 5]. The routing is obtained by introducing SWAP gates, which determine a permutation of the mapping between logical and physical qubits. The permutations are calculated so that the multi-qubit gates of the compiled circuit do not violate the connectivity restrictions of the hardware. In many architectures, the SWAP gate is constructed using three CNOTs, the latter have usually a higher error rate [6] when compared to single-qubit gates. Moreover, adding gates to the circuit may enlarge the circuit depth, thus increasing decoherence-related issues. Consequently, the qubit routing is an optimization problem that not only is required to fulfill the hardware constraints, but also to minimize some function that depends on the number of added SWAPs and possibly the depth of the resulting circuit.

The computational complexity of the combinatorial problem behind the swap mapping has been proved in several forms. In [4], the mapping problem is formulated as a Hamiltonian cycle problem, resulting in being NP-complete. A reduction from the sub-graph isomorphism problem in [7] shows that the problem is NP-hard. This means that the design of practical algorithms should employ a heuristic component. Among the heuristics, remarkable is the SWAP-based BidiREctional (SABRE) algorithm [8]. At the time of writing, the latter is considered the state of the art and is currently the default swap mapping method for the Qiskit framework [9].

The insertion of SWAP gates is not the only methodology for obtaining a circuit fulfilling the hardware connectivity constraints. In [10], a pattern called bridge template is used to route CNOTs without influencing the subsequent parts of the circuit. In some cases, the method could reduce the number of inserted CNOTs; however in the general case, a tradeoff between SWAP mapping and bridge templates is necessary.

A circuit manipulation related to the qubit routing is the qubit assignment [4, 7, 11, 12], where the initial mapping between hardware and logical qubits is determined. When the qubit assignment and routing are composed, the resulting procedure is denoted as qubit allocation; however, there are inconsistencies in literature regarding the terminology.

Despite the long-standing research on the topic, in [13] it was surveyed that there is still a substantial optimality gap on the solutions produced by the most widespread tools for circuit synthesis.

In this work, we present a swap mapping algorithm based on mathematical optimization and doubly stochastic matrices (DSM) [14]. Doubly stochastic matrices are convex combinations of permutations matrices. The intuition is that since the routing makes use of swaps (permutations) we can model the decision process using a linear combination of swap and identity matrices. Such combination is then tuned by means of continuous parameters controlled by an optimizer. In addition, powerful algebraic
properties of the resulting optimization problem allow the modeling of swap count and depth minimization. On the optimizer side, we propose a solver that scales linearly with the depth of the circuit, and our experiments show that for compiling quantum-volume circuits on 8 qubits our solver outperforms the state-of-the-art algorithm SABRE in terms of depth of the resulting circuit by 20 percentage points.

The paper is organized as follows. In Sect. 2, we introduce the basic definitions of the algebraic structures used throughout the formulation, and Sect. 3.1 contains the construction of the fundamental part of the optimization problem—the hardware constraints function. Sections 3.1.1, 3.1.2 and 3.1.3 expand on the optimization problem, its key features and the numerical method. In Sect. 4, we report the results from our experiments: numerical evaluation of the effect of algorithm’s hyperparameters on compilation outcome, and comparisons against the state-of-the-art algorithms. Finally, the conclusions are elaborated in Sect. 5. All the proofs related to the main results can be found in the appendix.

2 Mathematical preliminaries

**Notation** We introduce the basic notational conventions adopted throughout this work. We denote with $I_m$ the $m \times m$ identity matrix, and also $1_m$ denotes the vector of ones in the real vector space $\mathbb{R}^m$. Canonical basis vectors are identified with $e_i$; also sometimes, we highlight the dimension of the vector space by denoting with $e_i^{(m)}$ the $i$-th (starting from zero) basis vector for $\mathbb{R}^m$. Given the relation with quantum computing, we make use of the Dirac notation. In the context of this work, we interpret $|i\rangle^m$ as the $i$-th canonical basis vector for $\mathbb{R}^m$, that is $|i\rangle^m = e_i^{(m)}$. Consequently, the outer product $|i\rangle^m \langle j|_m$ represents the rank-one matrix $e_i^{(m)} (e_j^{(m)})^T$. For a vector $v \in \mathbb{R}^n$ and some integers $i, j$ such that, $0 \leq i \leq j < n$ we denote with $v[i..j]$ the sub-vector in $\mathbb{R}^{j-i+1}$ obtained through the linear transformation

$$v[i..j] = \sum_{k=0}^{m-1} |k\rangle^m \langle k+i|_n v, \quad (1)$$

with $m = j - i + 1$. Given integers $i, j$ such that $i \leq j$, we denote the set $[i, j] \cap \mathbb{Z}$ with $[i..j]$. The Hadamard product of $n \times m$ matrices $A, B$ is denoted with $A \odot B$ and the resulting $n \times m$ matrix has entries $(A \odot B)_{i,j} = A_{i,j}B_{i,j}$, where $A_{i,j}$ denotes the matrix entry at the $i$-th row and $j$-th column.

**Definition 1** We denote an $n \times n$ matrix $A$ with non-negative entries, as doubly stochastic (DSM), when

$$AJ_n = J_n A = J_n, \quad (2a)$$

where $J_n = 1_n 1_n^T$ is the $n \times n$ matrix of ones. In other words, both rows and columns of $A$ sum to 1 and $A_{i,j} \geq 0$. Moreover, the Birkhoff-von Neumann theorem [14]
asserts that any doubly stochastic matrix can be decomposed as convex combination of permutation matrices, that is

$$A = \sum_{i=0}^{n!-1} \lambda_i P_i$$  \hspace{2cm} (2b)$$

with $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$ and $\{P_i\}$ the set of $n \times n$ permutation matrices.

We mention a well-known result regarding the set of DSMs.

**Lemma 1** The set of $n \times n$ doubly stochastic matrices forms a monoid\(^1\) under matrix multiplication.

**Definition 2** Given any $n \times n$ matrix $A$, we define the row-major vectorization operator $\text{vec}_r : \mathcal{M}_{n,n}(K) \to K^n \otimes K^n$ with rule

$$\text{vec}_r(A) = \sum_{j=0}^{n-1} (A | j)_n \otimes | j)_n = \sum_{i,j=0}^{n-1} A_{i,j} (| i)_n \otimes | j)_n ,$$  \hspace{2cm} (3a)$$

where $\mathcal{M}_{n,n}(K)$ is the set of $n \times n$ matrices over some field $K$. A special case is given by the vectorization of the identity matrix

$$\text{vec}_r(\mathbb{I}_n) = \sum_i | i)_n \otimes | i)_n .$$  \hspace{2cm} (3b)$$

**Lemma 2** Let $A$ be any $n \times n$ matrix, then

$$\text{vec}_r(A) = (A \otimes \mathbb{I}_n) \text{vec}_r(\mathbb{I}_n) = \left( \mathbb{I}_n \otimes A^\top \right) \text{vec}_r(\mathbb{I}_n) .$$  \hspace{2cm} (4)$$

Proof in Appendix B. The next lemma presents some convenient identities that create a link between vectorization, tensor and Hadamard products and trace.

**Lemma 3** Let $A$, $B$ be any $n \times n$ matrices, then

$$\text{vec}_r(\mathbb{I}_n) \top (A \otimes B) \text{vec}_r(\mathbb{I}_n) = \text{Tr} \left( AB^\top \right)$$

$$= \mathbf{1}_n \top (A \odot B) \mathbf{1}_n .$$  \hspace{2cm} (5a)

$$= \mathbf{1}_n \top (A \odot B) \mathbf{1}_n .$$  \hspace{2cm} (5b)$$

Proof in Appendix B.

---

\(^1\) A monoid is defined as a set endowed with a binary associative operation and an identity element. In other words a monoid is a group without the inverse axiom. A counter-example to invertibility of DSMs is the following. Take $X$ to be any permutation matrix that swaps two elements, then its eigenvalues are $\{-1, 1\}$, now $\frac{1}{2}(\mathbb{I}_n + X)$ is a DSM, but it has a zero eigenvalue, consequently it is singular.
3 Main results

3.1 The hardware constraints function

In this section, we expand on the hardware constraints function which, given a configuration of SWAP gates, vanishes when the composition of SWAPs and circuit layers fulfill the connectivity constraints.

Let $M$ represent the hardware graph, in which the vertices are interpreted as the physical qubits and the edges the connectivity between them. For simplicity, we assume the connectivity to be undirected, and also the graph $M$ is assumed connected and nonempty. In the case of quantum technologies where the only 2-qubit gate is the CNOT gate, the reversal of the control target of such gate (Fig. 1) comes at no cost in terms of additional 2-qubit gates. Thus, the latter justifies the choice for the undirected graphs. We associate to graph $M$ the $m \times m$ adjacency matrix $M$. As a consequence of the assumed structure of $M$, we have that $M$ is symmetric, that is $M = M^\top$. Let $J_m = 1_m 1_m^\top$ be the $m \times m$ matrix of ones, so denote $M_c = J_m - M$ the adjacency matrix of the complement graph to $M$.

As outlined in [5], we decompose the circuit into layers of commuting two-qubits gates. An example of such circuit partitioning is presented in Fig. 2. The same circuit is also presented in Fig. 3 using a diagrammatic form that capture the essential information required for the swap mapping algorithm. In Fig. 4, instead we provide an example of circuit layer. Let $G(t)$ denote the graph corresponding to the layer at time step $t \in [0, T - 1]$, and $G(t)$ its adjacency matrix. The vertices of $G(t)$ correspond to the logical qubits, and the edges are the two qubit gates (Fig. 5). Let $V_{G(t)}$ denote the set

---

2 We should also assume that $M$ is not fully connected otherwise the swap mapping would not be necessary.
Fig. 3 The braid diagram with gate arcs (left) corresponding to the circuit in Fig. 2. In this example, we assume the line connectivity, so on the right we propose a qubit allocation solution. Note on the right-hand side that the initial and final qubit layouts do not match. Here the braids on the right-hand side can be interpreted as the permutation of logical qubits \( q_k \) associated to physical ones (vertical positions) of vertices of the graph \( G(t) \), then we impose \( |V_M| = |V_{G(t)}| = m \) and \( V_{G(t_1)} = V_{G(t_2)} \) for all \( t_1, t_2 \in [0..T-1] \).

Now, let \( P \) be an \( m \times m \) permutation matrix, then we devise the following constraint function for layer \( t \),

\[
\ell(t)(P) = 1_m^\top \left( PG(t) P^\top \right) \odot M_c 1_m \geq 0. \quad (6)
\]

We observe that the global minima (also matching with zeros) of \( \ell(t) \) correspond to the permutations of the logical qubits such that the two-qubits gates overlap hardware arcs. In general the solution \( P \) to the problem of making \( \ell(t)(P) \) vanish is not unique; however, later we will add a further restriction, that is the minimization of the number of SWAPs generating \( P \).

We obtain an equivalent formulation for the function in (6). Using Lemmas 2 and 3, we have

\[
\ell(t)(P) = \text{Tr} \left( \left( PG(t) P^\top \right) M_c^\top \right) \quad (7a)
\]

\[
= \text{Tr} \left( PG(t) (M_c P) ^\top \right) \quad (7b)
\]

\[
\overset{(5a)}{=} \text{vec}_r(\mathbb{I}_m)^\top \left( PG(t) \right) \otimes (M_c P) \text{ vec}_r(\mathbb{I}_m) \quad (7c)
\]

\[
= \text{vec}_r(\mathbb{I}_m)^\top (\mathbb{I}_m \otimes M_c) (P \otimes P) \left( G(t) \otimes \mathbb{I}_m \right) \text{ vec}_r(\mathbb{I}_m) \quad (7d)
\]

\[
\overset{(4)}{=} \text{vec}_r(M_c)^\top (P \otimes P) \text{ vec}_r(G(t)), \quad (7e)
\]

The new form in (7e) is convenient for the theory that follows, the key fact is that the constraint function is now linear with respect to \( P \otimes P \). The intent is that of obtaining a linear combination of constraint values (non-negative) for individual solutions. Specifically, each solution is determined by a permutation matrix \( P_i \); then given a set
of candidate solutions \{P_i\}, we obtain the convex combination
\[
\sum_i \lambda_i \ell^{(t)}(P_i) = \sum_i \lambda_i \text{vec}_r(M_c)^\top (P_i \otimes P_i) \text{vec}_r(G^{(t)})
\] (8a)
\[
= \text{vec}_r(M_c)^\top \left( \sum_i \lambda_i (P_i \otimes P_i) \right) \text{vec}_r(G^{(t)}),
\] (8b)

with \(\lambda \in \Delta_S\) where \(\Delta_S\) is the unit simplex in \(S\) dimensions [15]. In (8b), we have the first appearance of an interesting structure, whose characterization is given by the following lemma.

**Lemma 4** Let \(Q = \sum_i \lambda_i P_i\) be a doubly stochastic matrix, where \(P_i\) are \(m \times m\) permutation matrices. Then
\[
K = \sum_i \lambda_i P_i \otimes P_i
\] (9)
is doubly stochastic.

Proof in Appendix B.

The function \(\ell^{(t)}(P)\) is the key component of the optimization problem that will be defined later.

**Remark 1** We highlight again the linearity of the optimization problem determined by (8a). Take for simplicity the problem of minimizing the hardware constraints function for the layer \(t\). Consider \(m\) qubits and fix the set of permutation matrices \(\{P_i\}\) corresponding to the representations of the elements of the symmetric group \(S_m\). Then the problem takes the form
\[
\min_{\lambda \in \mathbb{R}^{m!}} \sum_{i=0}^{m!-1} \lambda_i \ell^{(t)}(P_i),
\] (10a)
s.t. \(\lambda_i \geq 0\) \(\forall i\),
\[\sum_i \lambda_i = 1.\] (10b)

The latter is clearly a linear problem over a convex set, however the difficulty arises from the cardinality \(m!\) of the set of permutations. We avoid this difficulty by introducing a construction that produces a subgroup of permutations controlled by a polynomial number of parameters (w.r.t. \(m\)).

The machinery for the swapping requires consistency between layers, that is, we have to assure the connectivity of logical qubits as we pass from one layer to the next. Let \(P^{(t)}\) be the permutation applied before the layer at time step \(t\), for all \(t \in\)
On the left-hand side, an example of circuit layer. Note that the CNOTs commute. On the right-hand side, the circuit after the application of the swap (highlighted)

\[ [0..T-1] \]. We define the finite sequence \( \{C^{(t)}\}_{t \in [0..T-1]} \) of permutations composed up to time \( t \) as

\[
\begin{align*}
C^{(0)} &= P^{(0)}, \\
C^{(t)} &= P^{(t)} C^{(t-1)}, & 1 \leq t \leq T - 1,
\end{align*}
\]

expanded, the sequence takes the following form

\[
C^{(0)} = P^{(0)}, \\
C^{(1)} = P^{(1)} P^{(0)}, \\
\vdots \\
C^{(T-1)} = P^{(T-1)} \ldots P^{(1)} P^{(0)}.
\]

Given a sequence of permutations \( \{P^{(t)}\}_{t \in [0..T-1]} \) we obtain the hardware constraints function for the overall circuit, so

\[
\mathcal{L} \left( \left( P^{(t)} \right)_t \right) = \sum_{t=0}^{T-1} \ell^{(t)} \left( C^{(t)} \right),
\]

again it can be shown that the global minima (zeros) correspond to the set of permutations that implements the circuit on allowable hardware arcs (the edges of graph \( M \)). In relation to Fig. 3, the permutations \( P^{(t)} \) correspond to the braids, and the graphs \( G^{(t)} \) match the layers with the gate arcs implementing the edges.

The limited connectivity of the hardware structure implies that the permutations \( P^{(t)} \) must be generated by the swaps corresponding to the directly connected vertices of graph \( M \). However, before expanding the aforementioned restriction we first proceed with the definition of SWAP. Let \( m \) be the number of qubits of the circuit and assume

\[ 3 \] The hardware graph \( M \) is assumed to be not fully connected.
\( m \geq 2 \). We define the swap operator w.r.t. distinct vertices \( i, j \in [0..m-1] \), as

\[
\text{SWAP}_m(i, j) := \mathbb{I}_m - |i\rangle_m \langle i|_m - |j\rangle_m \langle j|_m + |i\rangle_m \langle j|_m + |j\rangle_m \langle i|_m,
\]

equivalently the operator is determined by its action on the vectors \(|k\rangle_m\) with \( k \in [0..m-1] \),

\[
\text{SWAP}_m(i, j) |k\rangle_m = \begin{cases} 
|j\rangle_m, & k = i, \\
|i\rangle_m, & k = j, \\
|k\rangle_m, & \text{otherwise}.
\end{cases}
\]

**Remark 2** The \( \text{SWAP}_m \) defined in (13) should be considered as the classical counterpart of the swap operator \( \text{SWAP}_Q^m \) in the context of quantum circuits. The \( \text{SWAP}_m(i, j) \) operator acts on a space having dimension \( m \); also the result of its action is the swap of the the basis vectors \(|i\rangle_m\) and \(|j\rangle_m\). The \( \text{SWAP}_Q^m(i, j) \) instead, acts on a space of dimension \( 2^m \) and its action swaps \( 2^m-1 \) basis vectors. The latter is represented diagrammatically using the symbol \( \times \times \); however, in the context of this work, the same symbol is used to indicate both operators interchangeably.

We obtain the set of generators for the permutations implementing the swap mapping, so

\[
\mathcal{P}_M = \{ \text{SWAP}_m(i, j)|M_{i, j} = 1, i < j \},
\]

where \( M \) is the adjacency matrix for the hardware couplings. In other words, each permutation \( P^{(t)} \) for layer \( t \) is constructed by composing a subset of elements of \( \mathcal{P}_M \). Later we will introduce an objective that aims at minimizing the number of elements of such construction.

We extend again the approach by introducing a sort of 'smooth swap', that is a linear combination of swap and identity operators. We define the smooth swap operator \( \text{SSWAP}_m(i, j, \theta) \) as one of the following equivalent forms

\[
\text{SSWAP}_m(i, j, \theta) := \cos^2(\theta)\mathbb{I}_m + \sin^2(\theta)\text{SWAP}_m(i, j) \quad (15a)
\]

\[
= \mathbb{I}_m + \sin^2(\theta) (\text{SWAP}_m(i, j) - \mathbb{I}_m), \quad (15b)
\]

which can be shown to be convex combinations of the identity matrix and the swap acting on vertices \( i, j \). It follows from the Birkhoff-von Neumann theorem that \( \text{SSWAP}_m(i, j, \theta) \) is a doubly stochastic matrix [16]. The specific structure of the matrix \( \text{SSWAP}_m(i, j, \theta) \) is also known as an elementary doubly stochastic matrix [14]. By Lemma 1, the set of DSM of the same size is closed under matrix multiplication, so by composing multiple SSWAP operators we obtain another matrix of the same class. Notably, in our case, when the \( \theta \)s are integer multiples of \( \frac{\pi}{2} \) we obtain a vertex of the Birkhoff polytope, that is a permutation matrix. This means that we can substitute the component \( P \otimes P \) in (7e) with a composition (depending on some
hyperparameters) of operators of the form,

\[
PSSWAP_m(i, j, \theta) := \cos^2(\theta)I_m^\otimes 2 + \sin^2(\theta)SWAP_m(i, j)^\otimes 2
\]

\[
=I_m^\otimes 2 + \sin^2(\theta)\left(\text{SWAP}_m(i, j)^\otimes 2 - I_m^\otimes 2\right).
\]

Note that here we extended the concept of SSWAP to that of PSSWAP, where the prefix \(P\) stands for ‘parallelized’ which resembles the effect of the tensor power. In the next section, we continue with the substitution of the PSSWAP in the constraint function (12), resulting in the weighted sum of costs corresponding to the linear combination of solutions determined by the parameters \(\theta_s\).

**Remark 3** In relation to the definition of PSSWAP, we highlight that in general

\[
PSSWAP(i, j, \theta) \neq \text{SSWAP}_m(i, j, \theta)^\otimes 2.
\]

\[\triangle\]

### 3.1.1 The optimization problem: formulation

We start by defining the constructor generating the permutations \(P^{(t)}\) at each time step \(t\). We recall that we denote by \(T\) the number of layers of the input circuit, so the layer index \(t\) belongs to \([0 \ldots T - 1]\). Also each layer determines \(S\) continuous parameters, so overall we have \(S \times T\) parameters, which will be denoted by \(\theta \in \mathbb{R}^{ST}\).

The constructor configuration (hyperparameters) is a pair \((p_1, p_2)\) of functions \(p_1, p_2 : [0 \ldots S - 1] \rightarrow V_M\) mapping a sequence index \(s\) to a physical qubit.\(^4\) Moreover, in the scope of this section we consider the functions \((p_1, p_2)\) as arbitrary; however, Sect. 3.1.2 will develop a specific structure for them. At each time step \(t\) the hyperparameters determine a sequence of \(S\) swaps \(\text{SSWAP}_m(p_1(s), p_2(s), \theta_s^{(t)})\) for

\[^4\text{We also require that } i = p_1(s) \neq p_2(s) = j \text{ for all } s, \text{ so we always have distinct targets } \{i, j\} \text{ for the swap.}\]
Let $s \in [0..S-1]$. So, given a vector of continuous parameters $\theta = (\theta_s^{(t)})_{t,s} \in \mathbb{R}^{ST}$, the hardware constraints function assumes the form

$$
L(\theta) = \sum_{t=0}^{T-1} L_t(\theta),
$$

(18)

with

$$
L_t(\theta) = \beta(t) \text{vec}_t(M_c) \top K^{(t)} \text{vec}_t(G^{(t)}),
$$

(19a)

and

$$
P^{(t)} = \text{PSSWAP}_m(p_1(0), p_2(0), \theta_0^{(t)}) \cdots \text{PSSWAP}_m(p_1(S-1), p_2(S-1), \theta_{S-1}^{(t)}),
$$

(19b)

$$
K^{(0)} = P^{(0)},
$$

(19c)

$$
K^{(t)} = P^{(t)} K^{(t-1)}, \quad 1 \leq t \leq T-1.
$$

(19d)

In (18), each term is scaled by the function $\beta : \mathbb{R} \rightarrow \mathbb{R}$ which is assumed decreasing and positive valued. The function $\beta$ is one of the key components of the heuristic solver—the adaptive feasibility, which is introduced in Sect. 3.1.3. Note that each continuous parameter $\theta_s^{(t)}$ appears as PSSWAP argument, one time for all terms in (18) such that $t \geq t_1$. Also $L(\theta) \geq 0$ for all $\theta$.

We obtain the ideal form for the optimization problem, that is

$$
\min_{\theta \in \mathbb{R}^{ST}} \text{card}(\theta),
$$

s.t. $L(\theta) = 0,
$$

(20)

where $\text{card}(\theta)$ is the cardinality$^5$ of the vector $\theta$. The structure of the problem is justified by the fact that for $\theta_s^{(t)} = 0$, the corresponding PSSWAP is the identity permutation. The next proposition shows that an optimization problem with cardinality as objective and which constraint matches a certain structure can be solved by an equivalent differentiable problem.

**Theorem 5** Given the structure of the hardware constraints function $L$, the following optimization problems are equivalent

$$
\begin{align*}
\min_{\theta \in \mathbb{R}^{ST}} \|\theta\|_2^2, \\
\text{s.t.} \quad L(\theta) = 0,
\end{align*}
\quad \cong \quad
\begin{align*}
\min_{\theta \in \mathbb{R}^{ST}} \text{card}(\theta), \\
\text{s.t.} \quad L(\theta) = 0.
\end{align*}
$$

(21)

Proof in Appendix B.

---

$^5$ The cardinality of a vector $x \in \mathbb{R}^n$ is defined as $\text{card}(x) := |\{i|x_i \neq 0\}|$, where $|\cdot|$ is the set cardinality.
This proposition demonstrates that the problem in (20) can be solved by considering the squared $l_2$-norm of $\theta$ instead of its cardinality.

Let us study the characterization of the stationary points of the hardware constraints function $L$. The next proposition shows that vectors $\theta \in \mathbb{R}^{ST}$ such that the individual elements are integer multiples of $\frac{\pi}{2}$, are a sufficient condition for the stationary points of the hardware constraints function.

**Theorem 6** Consider the hardware constraints function (18) and the element $\theta_s^{(t)}$ with index $(s, t)$ belonging to the vector $\theta \in \mathbb{R}^{ST}$. Let $\Omega = \{ k \frac{\pi}{2} \mid k \in \mathbb{Z} \}$ be the set of integer multiples of $\pi/2$. Then

1. $\theta^T e_s^{(t)} \in \Omega \implies \frac{\partial L(\theta)}{\partial \theta_s^{(t)}} = 0$.

If $\frac{\partial L(\theta)}{\partial \theta_s^{(t)}} = 0$, for some $\theta \in \mathbb{R}^{ST}$, then, either

2. $\theta^T e_s^{(t)} \in \Omega$,

3. or $\theta^T e_s^{(t)} \notin \Omega$ and $\frac{\partial L(\hat{\theta})}{\partial \theta_s^{(t)}} = 0$ for $\hat{\theta} \in \{ \theta + r e_s^{(t)} \mid r \in \mathbb{R} \}$.

Proof in Appendix B.

**Remark 4** Note that Case 3 of Proposition 6 suggests that the constraint is locally “flat” in the direction $e_s$ which means that multiple solutions are possible, but this ambiguity is resolved by 2-norm minimization which simply picks 0 in this case. Let us further illustrate Case 3: given circuits $C_1$ and $C_2$, we define the equivalence relation $C_1 \sim C_2$ whenever the circuits implement the same unitary up to a permutation. Now, assume the line connectivity and consider the swap mapping process that follows.

\[
\begin{align*}
q_0 : & \quad \text{SWAP} \quad q_0 : \\
q_1 : & \quad q_1 : \theta_0 \\
q_2 : & \quad q_2 : \theta_1
\end{align*}
\]

With circuit swap gates interpreted as SSWAPs controlled by a parameter $\theta_k \in \mathbb{R}$, the central circuit reports a solution with $\theta_0 = \pi/2$; however, adding another swap ($q_1 \rightarrow q_2$) as depicted on the right-hand side does not alter the hardware constraints function; consequently, the partial derivative corresponding to the second swap is zero for any $\theta_1 \in \mathbb{R}$.

In other words, there may exist configurations of the parameters vector $\theta$ such that one or more elements of such vector determine a flat cost. However, the objective of the optimization problem (20), being a cardinality, favors the zero value for the free parameters.

### 3.1.2 The optimization problem: constraint specification for different topologies

In the construction of the hardware constraints function $L$, we claimed that the sequence of PSSWAPs in (19b) depends on some hyperparameters. In the present section, we determine the structure of the PSSWAPs for the case of the line connectivity between qubits. Despite the simplicity of the present topology, the results obtained
here are key for the generalization to the arbitrary connectivity. In relation to general topologies, we notice that recently there have been a widespread adoption of quantum processing unit topologies based on hexagonal lattices [17]. An example is depicted in Fig. 6. Moreover, the results obtained in this section are extend to the general case in Appendix A.

The line connectivity model is defined as a chain of $m$ qubits (assuming $m \geq 3$) where the neighborhoods of qubit $k \in [1 \ldots m-2]$ are qubits $k-1$ and $k+1$. The extremes of the chain, that is qubits 0 and $m-1$, have neighborhood, respectively, qubit 1 and qubit $m-2$.

Assume for simplicity that the number of qubits $m$ is an odd integer greater than two. Then the set of generating swaps $\mathcal{P}_M$ defined in (14), contains $m-1$ elements. We partition the set $\mathcal{P}_M$ into the following subsets

$$\Gamma_1 = \{ \text{SWAP}(2k, 2k+1) | k \in [0 \ldots (m-1)/2] \},$$

$$\Gamma_2 = \{ \text{SWAP}(2k+1, 2k+2) | k \in [0 \ldots (m-1)/2] \},$$

(23)

so $\mathcal{P}_M = \Gamma_1 \cup \Gamma_2$. We note that for any $Q_1, Q_2 \in \Gamma_1$, then $Q_1 Q_2 = Q_2 Q_1$, similarly it also holds for $\Gamma_2$.

The next lemma extends the commutativity from the generating permutations to the generated DSM. The result is immediate, so we state the claim without proof.

**Lemma 7** Let $P_1, P_2$ be $m \times m$ permutation matrices such that $P_1 P_2 = P_2 P_1$, that is the permutations commute. Then the following doubly stochastic matrices also commute

$$Q_k = (1 - \alpha_k) I_m + \alpha_k P_k,$$

(24)

with $\alpha_k \in [0, 1]$, for $k = 1, 2$. That is, $Q_1 Q_2 = Q_2 Q_1$.

Consequently, the product of doubly stochastic matrices obtained from either $\Gamma_1$ or $\Gamma_2$ commute. Using (16b) we define the composition of such commuting matrices

---

6 Equivalent to the SSWAP defined in (15a), or the PSSWAP defined in (16a).
Fig. 7 Example of the pattern of swaps for the line connectivity with \( m = 5 \) qubits. The dashed frames represent, respectively, the set \( \Gamma_1 \) and \( \Gamma_2 \).

(PSSWAPs) controlled by the continuous parameters \( \theta \), so

\[
C(\Gamma_k, \theta) = \prod_{P_i \in \Gamma_k} \left( \mathbb{I}_m^{\otimes 2} + \sin^2(\theta_i) \left( P_i^{\otimes 2} - \mathbb{I}_m^{\otimes 2} \right) \right). \tag{25}
\]

To complete the construction, we specialize the definition of \( P(t) \) related to the hard-

ware constraints function in \((19b)\), with a (finite) sequence of alternating structures of the form \( C(\Gamma_k, \theta) \), that is

\[
P(t) = C(\Gamma_1, \theta(t)) \cdot C(\Gamma_2, \theta(t)) \cdot C(\Gamma_3, \theta(t)) \cdot C(\Gamma_4, \theta(t)) \cdots \tag{26}
\]

The next lemma shows one of the motivations that justify this construction, that is efficient matrix multiplication for permutations within the same partition.

**Lemma 8** Let \( a, b \in S_m \) be distinct, involutory and commuting elements of the symmetric group of degree \( m \), that is \( a \neq b, a^2 = b^2 = e \) and \( a \circ b = b \circ a \), where \( e \in S_m \) is the identity element. Let \( P_a, P_b \) be the permutation representations [18] of \( a \) and \( b \), respectively. Then

\[
P_a P_b = P_a + P_b - \mathbb{I}_m. \tag{27}
\]

Proof in Appendix B. As a corollary, it can be readily proven that given a (finite) non-empty set of \( n, m \times m \) permutation matrices \( \{P_t\}_{t=1}^n \), in which each pair of elements \( P_a, P_b \), fulfills the conditions of the Lemma, then

\[
\prod_{t=1}^n P_t = \left( \sum_{t=1}^n P_t \right) - (n - 1)\mathbb{I}_m. \tag{28}
\]

We also note that the maximum number of disjoint and thus commuting permutation matrices is \( n = \lfloor m/2 \rfloor \).

The next step is that of applying the latest result to a product of doubly stochastic matrices whose generating permutations fulfill the conditions of Lemma 8.

**Theorem 9** Let \( \{P_t\}_{t=1}^n \) be a non-empty set of \( m \times m \) matrices such that each element is involutory and each pair commutes. Then the following identity holds for the composition of elementary DSM generated by the \( P_t \),

\[
\prod_{t=1}^n \left( \mathbb{I}_m + \sin^2(\theta_t) (P_t - \mathbb{I}_m) \right) = \mathbb{I}_m + \sum_{t=1}^n \sin^2(\theta_t) (P_t - \mathbb{I}_m). \tag{29}
\]
Proof in Appendix B. Finally, we apply the results just obtained to the definition of layer of PSSWAPs (25), so

\[
C (\Gamma_k, \theta) = \mathbb{I}^\otimes_m + \sum_{P_i \in \Gamma_k} \sin^2(\theta_i) \left( P_i^\otimes^2 - \mathbb{I}^\otimes_m \right).
\]

(30)

Assume we have \(m\) qubits, then it can be shown that the repetition of patterns constructed as depicted in Fig. 7 generates a set of permutations that contains a subgroup of the symmetric group \(S_m\). Also as one may expect, increasing the number of replica creates subgroups that approach in term of order, the group \(S_m\). So, it would be interesting obtaining results regarding the required number of repetitions and their efficiency. For now however, we do not expand the latter point which will be addressed in future research. To conclude, we claim without proving that this construction favors the minimization of the resulting circuit depth. However, evidence for the latter assertion emerges in the experiments section.

3.1.3 The optimization problem: numerical method

In this section, we develop a heuristic for the solution of problem (20). We start from a well-known technique called Rolling Horizon (RH). The latter consists of partitioning a decision problem into a sequence of sub-problems which aggregated solutions constitute a solution for the whole problem. The sub-problems often are identified by time windows of fixed length. Examples of the aforementioned strategy can be found in [19, 20].

We build on the RH strategy to obtain a process we call adaptive feasibility. The terminology finds the following motivation—we call it adaptive because the RH depth adjusts to the best sub-problem where we can reach feasibility. Considering the structure of the hardware constraints function, we note that for all \(s\), the variables \(\theta_s(\tau)\), appear in the terms \(L_t\) of (18) with \(t \geq \tau\). Since the permutation at time \(t\) influences the permutations of the subsequent layers, then we favor the feasibility of lower layers (w.r.t. time \(t\)) using the decreasing function \(\beta(t)\) introduced in (19a). Now, it follows from Proposition 5 that the optimization problem takes the equivalent form

\[
\min_{\theta \in \mathbb{R}^{ST}} \|\theta\|^2_2, \\
\text{s.t. } L(\theta) = 0,
\]

which we solve using the Differential Multiplier Method [21]. The method, by introducing the Lagrange multiplier \(\lambda\), produces a sequence of updates for the variables \(\theta\) and \(\lambda\), so

\[
\theta \leftarrow \theta - \eta_\theta \nabla_\theta \left( \|\theta\|^2_2 + \lambda L(\theta) \right), \quad (31a)
\]

\[
\lambda \leftarrow \lambda + \eta_\lambda \nabla_\lambda \left( \|\theta\|^2_2 + \lambda L(\theta) \right), \quad (31b)
\]
until a stop condition is reached. We denoted with \( \eta_\theta \) and \( \eta_\lambda \) the step sizes for the variables \( \theta \) and \( \lambda \), respectively. Interestingly, since \( \mathcal{L}(\theta) \geq 0 \), the second update corresponds to a monotonic increase of \( \lambda \), that is

\[
\lambda \leftarrow \lambda + \eta_\lambda \mathcal{L}(\theta).
\]  

(32)

The latter can be used to prove that the optimizer gets attracted by the stationary points of \( \mathcal{L} \), but according to Proposition 6, such points have a known and convenient structure.

The entire procedure is split into two algorithms, the DSM-SWAP and the Knitter, with the former being the main algorithm.

### 3.1.4 The Knitter algorithm

The Knitter\(^7\) algorithm can be interpreted as a global solver (w.r.t. the circuit) for the swap mapping problem. However, the input circuit is split into sections using the RH strategy so the Knitter only acts upon each of the sub-circuits separately.

We describe the steps of the algorithm as presented in Algorithm 1. The input circuit is given as a sequence of graphs \( G^{(t)} \), the latter is used alongside the hardware graph \( M \) to construct (function \text{BuildHardwareCost}) the hardware constraints function \( \mathcal{L} \). The construction depends on the machine topology; details are elaborated in sections A and 3.1.2.

In the for-loop at line 6, the vector \( \theta \) is updated in a gradient descent fashion using the gradient of the function \( f(\theta) = \|\theta\|_2^2 + \lambda \mathcal{L}(\theta) \). Since the problem is non-convex, the iteration is executed \( \max_{\text{trials}} \geq 1 \) times. At the end of each iteration, the projection\(^8\) \( \mathcal{P}_{\Omega_{ST}'} \) onto the set \( \Omega_{ST}' \) is applied to the vector \( \theta \), with \( \Omega \) defined as in Proposition 6. Furthermore, the application of the projector is justified by Proposition 6.

Completed the trials at line 15, we use the merit function \( g(\theta) = \|\theta\|_2^2 + \alpha \mathcal{L}(\theta) \) to choose the best solution\(^9\). Here the parameter \( \alpha > 0 \) is a trade-off between swaps minimization and feasibility maximization. Once the best solution is realized, at line 16 we count the number \( l \) of subsequent layers, starting from the first, that fulfill the hardware constraints. Here we denote with \( \delta_{\{0\}} \) the indicator function\(^{10}\) for the set \( \{0\} \subseteq \mathbb{R} \). Remarkably, line 16 constitutes one of the key elements for the adaptive feasibility. Finally, the algorithm returns the sub-vector of \( \theta^* \) corresponding to the first \( l \) layers.

\(^7\) The name Knitter is inspired by the braid diagrams (Fig. 8) used to represent the iteration of the SWAPs.

\(^8\) The definition of projection is given in the proof of Lemma 13.

\(^9\) In our specific implementation, we choose the solution that maximizes the number of feasible layers; consequently, the adaptive horizon step is maximized, leading to a reduction of the overall computation time for the method.

\(^{10}\) The indicator function for a subset \( C \subseteq U \) is defined as the extended real-valued function \( \delta_C : U \to \mathbb{R} \cup \{\infty\} \) with rule \( \delta_C(x) = \begin{cases} 0, & x \in C, \\ \infty, & \text{otherwise}. \end{cases} \)
Algorithm 1 The pseudocode for the Knitter algorithm.

Data: Sequence of $T'$ circuit layers as graphs $G^{(t)}$. Hardware connectivity graph $\mathcal{M}$.
Result: $\theta^*$; // Parameters for the feasible layers
$R \leftarrow \emptyset$; // Set of solutions from trials
$\mathcal{L} \leftarrow \text{BuildHardwareCost}\left(\left(G^{(t)}\right)_{t=1}^{T'}, \mathcal{M}\right)$; // Prepare the hardware constraints

function

for $t \leftarrow 1$ to max_trials do

$\theta \sim \text{U}_{ST'}(0, \epsilon)$; // Sample initial $\theta \in \mathbb{R}^{ST'}$, with $\epsilon > 0$ a small constant.
$\lambda \sim \text{U}(0, \epsilon)$; // Sample initial $\lambda$. $\text{U}$ is the uniform distribution.

for $\tau \leftarrow 1$ to max_optim_steps do

$\theta \leftarrow \theta - \eta\nabla_{\theta}\left(\|\theta\|_2^2 + \lambda\mathcal{L}(\theta)\right)$; // Update $\theta$ with GD step
$\lambda \leftarrow \lambda + \eta\nabla_{\lambda}\mathcal{L}(\theta)$; // Update $\lambda$
if $\|\nabla_{\theta}\mathcal{L}(\theta)\|_2^2 \leq \gamma$ then
break; // Early stopping condition met
end

$R \leftarrow R \cup \left\{P\Omega_{ST'}(\theta)\right\}$; // Store projection of current $\theta$
end

$\theta^* \leftarrow \text{argmin}_{\theta \in \mathbb{R}^{ST'}} \|\theta\|_2^2 + \alpha\mathcal{L}(\theta)$; // Best solution selection policy

$l \leftarrow \text{argmin}_{k \in [0..T']}(\sum_{i=0}^{k-1}\mathcal{L}_i(\theta^*)) - k$; // Count feasible layers in $l$
if $l$ is 0 then

$\theta^* \leftarrow \text{nil}$ return
end

$\theta^* \leftarrow \theta^*[1..l*S]$; // Select parameters (sub-vector) for the $l$ feasible layers

3.1.5 The DSM–SWAP algorithm

The DSM–SWAP algorithm in essence partitions the input circuit into sub-circuits upon which the Knitter algorithm is executed. The pseudocode is presented in Algorithm 2, also we recall that we denote with $T$ the number of layers of the circuit and with $S$ the number of parameters (equivalently SWAPs) per layer. The depth of the sub-circuits is given by the hyperparameter horizon; however, the starting point for the horizon advances adaptively (line 31) depending on the feasibility reached by the previous iteration.

Function ThetasToSwaps invoked at lines 25 and 34 takes a vector of angles $\theta$ to a sequence of permutations matrices. We note that the elements of the vector $\theta$ are expected to belong to the set $\Omega$ (Proposition 6), that is the angles represent a vertex of the Birkhoff polytope. But this is consistent with the value returned by function Knitter.

We remark one additional point at line 25. The permutations applied to layer $t$, influence all the subsequent layers from $t + 1$ to $T - 1$. Consequently, we make the algorithm consistent with the mechanism by pre-permuting the qubits of each block.
with the permutations from the previous layers. In the specific expression at line 25, we denote with \((\cdot) \bullet (\cdot)\) the action of the permutation \(P\) on the circuit \(C\).

Finally, the result of the method consists of a sequence of \(T \times S\) involutory permutation matrices (either identity or SWAP). In Fig. 6, the reader can appreciate the resulting structure visually.

Algorithm 2 The pseudocode for the DSM-SW AP algorithm.

```
Data: Sequence of \(T\) circuit layers as graphs \(G^{(t)}\). Hardware connectivity graph \(M\).
Result: \(R\); // Sequences of swaps for each layer

\(\theta^* \leftarrow ()\); // Init empty vector (dim(\(\theta^*\)) = 0) for the parameters
\(t \leftarrow 0\) while \(t < T\) do
    // Iterate over circuit layers
    \(h \leftarrow \min(\text{horizon}, T - t)\); // Compute effective horizon
    \(C \leftarrow (G^{(t)})_{t=1}^{t+h}\); // Construct sub-circuit with depth up to horizon
    \(P \leftarrow \text{ThetasToSwaps}(\theta^*)\) \(C \leftarrow P \bullet C\); // Apply the resulting permutation up to layer \(t\)
    \(\theta \leftarrow \text{Knitter}(C, M)\); // Run Knitter on permuted sub-circuit \(C\)
    if \(\theta\) is nil then
        // If there are no feasible layers
        \(R \leftarrow \text{nil}\); // fail or run alternative strategy
        return // Nil result
    end
    \(t \leftarrow t + (\text{dim}(\theta))/S\); // Adaptively update the next horizon starting point
    \(\theta^* \leftarrow \theta^* \oplus \theta\); // Extend the vector of parameters
    // Note \(\text{dim}(\theta^*) = t \times S \leq T \times S\).
end // The final \(\theta^*\) belongs to \(\mathbb{R}^{ST}\)
\(R \leftarrow \text{ThetasToSwaps}(\theta^*)\); // Obtain swaps for all layers
```

4 Experiments

The purpose of the experiments is twofold. On one hand, we aim at obtaining a clearer view of the effects of the hyperparameters. On the other one, we compare the results of the proposed method with other well-known algorithms in literature.

The algorithm DSM-SWAP has been implemented \(^{11}\) on top of the frameworks Qiskit \([9]\) and PyTorch \([22]\). From the Qiskit library, we also made use of the algorithm SABRE and the circuit generators for the multi-controlled X gate and Quantum Volume.

The circuits involved in the experiments may present features with different distributions, depending on patterns and number of qubits. Also the measurements we are considering are taken both before and after the application of the swap mapping process. To evaluate the generalization of the strategy we consider several circuit patterns,

\(^{11}\) Source code available at https://github.com/qiskit-community/dsm-swap. Also data regarding the experiments is available on request.

\(\text{Springer}\)
specifically the quantum volume (QV\textsuperscript{12}) circuits [23] and the multi-cX gates compiled with an alphabet consisting of CNOT and single qubit gates. Given a circuit pattern\( C \) and a feature\( X \), we denote with \( X_0(C) \) and \( X_s(C) \) the random variables for the feature \( X \) measured on \( C \), respectively, before and after the swap mapping. Then given the samples \( x_0 \sim X_0(C) \) and \( x_s \sim X_s(C) \), assuming \( x_0 \neq 0 \), we define the relative measure as \( d = \frac{x_s - x_0}{x_0} \). The features we consider are the number of CNOTs and the circuit depth.\textsuperscript{13} The features are computed on the circuit resulting from the invocation of the Qiskit.\textsuperscript{14} transpiler with optimization level three\textsuperscript{15} We note that the transpiler embodies algorithms based on the stochastic approach, thus modeling using random variables is justified. Take for example the number of CNOTs and let \( c_0, c_s \) be their count before and after the swap mapping. Then the relative measure \( dcnots = \frac{c_s - c_0}{c_0} \) represents the fractional increase in the number of CNOTs as a result of the qubit allocation. So for example, if the circuit prior to the swap mapping contains 100 CNOTs and the process produces a fractional increase of 0.75 units (\( dcnots = 0.75 \)), then the final circuits contains 175 CNOTs. Similarly, we denote the fractional increase in depth with \( ddepth \).

4.1 Study of the hyperparameters

We start by describing the structure of the sampling. We generate 250 QV circuit instances for each qubit count from 5 to 8 (both inclusive). Each circuit instance is then processed by the DSM-SWAP algorithm configured with combinations of increasing horizon \{1, 2, 4\} and maximum optimizer steps \{10, 30, 100\}. In Fig. 9, we highlight the general positive effect of a longer horizon on both \( dcnots \) and \( ddepth \). Moreover, Fig. 8 provides the same evidence using braid diagrams. In Fig. 10, we observe vague evidence that a higher number of steps increases the quality of the results. This means that the optimizer converges quite fast so we conclude that there is not much difference between 30 and 100 steps, and thus the former value is set as the default.

4.2 DSM-SWAP vs SABRE

We compare the new method with the algorithm SABRE configured with the look-ahead strategy [8]. The circuits considered are the quantum volume and the multi-cX gates. The data for DSM-SWAP are the same one obtained for the hyperparameters investigations\textsuperscript{16} we denote the corresponding features with \( dcnots_{dsm} \) and

\textsuperscript{12} In the interest of space, we denote with QV\{\( n \}\) a quantum volume circuit with \( n \) qubits. Moreover, the latter is generated via the function qiskit.circuit.quantumcircuit.QuantumCircuit\((n, seed=seed)\), where seed determines the circuit instance.

\textsuperscript{13} We consider the Qiskit [9] definition of circuit depth which can be be obtained through the method qiskit.circuit.QuantumCircuit.depth.

\textsuperscript{14} Qiskit 0.36.1 and Qiskit Terra 0.20.1.

\textsuperscript{15} Specifically, the structure of the invocation is qiskit.compiler.transpile(\ldots, optimization_level=3).

\textsuperscript{16} We fix the hyperparameter max_optim_steps = 30.
\(d\text{depth}_{dsm}\). In addition, we execute SABRE on the same circuits and measure the features \(dc\text{not}_{sabre}\) and \(d\text{depth}_{sabre}\). In Figs. 11 and 12, we plot the cCDFs for the features gaps

\[
\begin{align*}
d\text{nots} &= dc\text{not}_{sabre} - dc\text{not}_{dsm} \\
d\text{depth} &= d\text{depth}_{sabre} - d\text{depth}_{dsm}.
\end{align*}
\]  

Since the selected features, the smaller they are the better the performance, then points on the positive abscissa correspond to DSM-SW AP performing better than SABRE. In Fig. 11, we see that for \(\text{horizon} \geq 2\), the new method has at least 80% chances to produce a shallower circuit. In Fig. 12, we distinguish the results for QV8 circuits assuming line and ring couplings. Surprisingly, in the ring connectivity case, the gap is remarkable even for \(\text{horizon} = 1\). We think that the latter could be an important clue for the development of new swap mapping methods (Fig. 13).

In Fig. 14, we obtain the average values for the merit measure defined as

\[
\text{merit} = dc\text{nots} + d\text{depth},
\]

which can be interpreted as the overall increase in both CNOT count and depth, as a consequence of the swap mapping. The latter figure shows how our method compares to SABRE as the increases; also we consider MCX and QV8 circuits.

**Remark 5** We expand on the reason for the choice of QV and MCX circuits. The quantum volume circuits have the property that each layer is represented by a graph \(G^{(l)}\) that has the maximum number of edges such that the corresponding two-qubit gates commute. This feature can be appreciated in Fig. 8 by observing the columns (layers) of red edges.

On the other hand, MCX circuits, compiled with an alphabet of CNOT and single qubit gates, present layers that tend to have a single edge, consequently they represent the opposing case to QV.

Following the previous remark, we assert that the case of the QV circuits is the most advantageous for DSM-SW AP since the construction of the SWAPs (Sects. 3.1.2 and A) can maximize the parallelism of the permutations. The MCX circuits instead should be the opposing case, indeed we observe in Fig. 14 that we need a longer horizon to obtain a neat advantage over SABRE.

### 5 Conclusions

The swap mapping process is fundamental for the quantum compiler and increasing its efficiency is essential for improving the performance of the hardware. In this work we obtained a procedure that combines both mathematical optimization and heuristic strategies. In literature however, methods based on mathematical optimization have not recorded particular successes when applied to this problem. Conscious of the computation complexity hardness of the problem we understand that, to be practical, the algorithm must include an heuristic component.
Fig. 8 Braid diagrams for the swap mapping applied to a quantum volume circuit with 8 qubits. The results, starting from the top, correspond to, respectively, horizon 1 and 4. It can be noticed that in the bottom case the density of the braids (permutations) is visibly lower.

The decision process for the insertion of the SWAPs has been modeled as a smooth optimization based on doubly stochastic matrices. Also, we obtained a procedure to build the pattern of SWAPs depending on the topology of the target hardware. One of the bottlenecks in the evaluation of the constraint function is determined by the number of matrix multiplications; however, using the commutativity properties of the pattern, we devised an efficient scheme for reducing the computational cost. The solver heuristic is inspired by the rolling horizon policy and it offers linear scaling of the computational complexity with respect to the depth of the circuit. The algorithm has been implemented on top of the frameworks Qiskit and PyTorch; also, we made the code open source. The experiments revealed clues regarding the algorithm hyperparameters and its relation with the state-of-the-art algorithm SABRE. The optimality of the results has been measured using the relative increment in CNOT gate count and depth. For the hyperparameters, we found that an increased horizon for the adaptive feasibility influences positively the performance of the results. The comparison with the algorithm SABRE shows that at the cost of increased computational time and while preserving the number of CNOTs, the new method delivers significant reduction in the depth of the resulting circuit. We see the potential application of our method to the compiling of quantum libraries where the processing time can be penalized in favor of depth and CNOTs optimality.

Despite the positive results, further research is required to extend the applicability of the method to the upcoming quantum hardware where we expect the number of qubits to climb to hundreds if not thousands.
Fig. 9 The eCDFs for features $\text{dcnots}$ (left column) and $\text{ddepth}$ (right column) measured on the algorithm DSM-SWAP. The top row configuration consists of increasing horizon ($\text{horizon} = 1, 2, 4$) over $\text{QV}\{5, 6, 7, 8\}$ circuits and maximum optimizer steps fixed to 30. The bottom row corresponds to the same configuration; however, it relates exclusively to the QV8 circuits.

**Acknowledgements** This study has received funding from the Disruptive Technologies Innovation Fund (DTIF), by Enterprise Ireland, under project number DTIF2019-090 (project QCoIR). The literature review is based on some unpublished work by Claudio Gambella, Andrea Simonetto, Anton Dekusar and Giacomo Nannicini. The open source implementation was realized thanks to the collaboration of Anton Dekusar, Albert Akhriev and Kevin Krsulich. The authors are highly thankful to Ali Javadi-Abhari, Dr. Martin Mevissen and Tara Murphy for their support and suggestions.

**Declarations**

**Financial or non-financial interests** The authors have no relevant financial or non-financial interests to disclose. The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.
Appendix A: Arbitrary topology

We extend the results obtained in the previous section from line connectivity to arbitrary topology. The disjoint partitioning $\mathcal{P}_M = \Gamma_1 \cup \Gamma_2$ of the set of generating swaps, obtained in (23), can be interpreted as a special case of the graph edge coloring problem. In Fig. 15, we provide an example of line connectivity related to the partitioning and its relation to edge coloring.

We recall that the $m$ swap targets $i, j$ for each swap $\text{SWAP}_m(i, j) \in \mathcal{P}_M$, and the swaps in $\mathcal{P}_M$, correspond, respectively, to the vertices and the edges of the hardware connectivity graph $M$. Then the minimum number of partitions of $M$, such that no two incident edges are in the same subset, is called the edge chromatic index. For a graph $G$, we denote its chromatic index with $\chi_e(G)$. Therefore, the optimal partitions $\Gamma_k$ correspond to the colors in the edge coloring problem. An example for this general case is provided in Fig. 16.
Fig. 11  The eCDFs for features gap dcnots (left) and ddepth (right) for the comparative case DSM-SWAP vs SABRE. The DSM-SWAP algorithm is configured with increasing horizon \(\text{horizon} = 1, 2, 4\) and \(\text{max\_optim\_steps} = 30\). The vertical gray line at abscissa zero determines the threshold where the two algorithms produce the same result (w.r.t. the current feature), whereas on the right-hand side, the DSM-SWAP yields more favorable results.

Fig. 12  The same experiment as that in Figure 11 except that here we distinguish the case with QV8 circuits with line (top row) and ring (bottom row) coupling maps.
A doubly stochastic matrices-based...

Fig. 13 Average time for the DSM-SWAP obtained at increasing horizon with QV circuits

Fig. 14 Comparison DSM-SWAP vs SABRE w.r.t. the merit function defined in (34), on MCX (left) and QV8 (right) circuits

\[
\begin{align*}
\Gamma_1 &= \{\text{SWAP}_5(0, 1), \text{SWAP}_5(2, 3)\}, \\
\Gamma_2 &= \{\text{SWAP}_5(1, 2), \text{SWAP}_5(3, 4)\}.
\end{align*}
\]

Fig. 15 An example of edge coloring for the line connectivity. On the LHS, we have the graph $\mathcal{M}$ with the line pattern representing the color of the edge. On the RHS, we have the partitions obtained in (23)

\[
\begin{align*}
\Gamma_1 &= \{\text{SWAP}_5(0, 1), \text{SWAP}_5(2, 3)\}, \\
\Gamma_2 &= \{\text{SWAP}_5(1, 2), \text{SWAP}_5(3, 4)\}, \\
\Gamma_3 &= \{\text{SWAP}_5(0, 2), \text{SWAP}_5(1, 3)\}.
\end{align*}
\]

Fig. 16 An example of edge coloring and the corresponding partitioning of the generators set. On the LHS, we have the graph $\mathcal{M}$ with the line pattern representing the color of the edge
assume that the hardware graph \( \mathcal{M} \) corresponds to an arbitrary topology. Then, by applying the edge coloring procedure, we determine the optimal partitioning

\[
P_{\mathcal{M}} = \bigcup_{k=1}^{\chi_e(\mathcal{M})} \Gamma_k,
\]

such that fixed any \( k \in [1 .. \chi_e(\mathcal{M})] \), we have that \( P_a, P_b \in \Gamma_k \) implies \( P_a P_b = P_b P_a \).

Given the max degree of a graph \( \mathcal{G} \), denoted by \( \Delta(\mathcal{G}) \), a well-known result by Vizing [24] establishes the boundaries for its chromatic index. We report the aforementioned theorem.

**Theorem 10 (Vizing)** Given any finite and simple graph \( \mathcal{G} \), then

\[
\Delta(\mathcal{G}) \leq \chi_e(\mathcal{G}) \leq \Delta(\mathcal{G}) + 1.
\]

In other words, we need at most \( \Delta(\mathcal{M}) + 1 \) subsets for the partitioning of \( P_{\mathcal{M}} \).

In relation to the heavy-hex lattice topology depicted in Fig. 6, we notice that graphs of such form are planar and have maximum degree 3. Consequently, by Theorem 10, a hexagonal lattice connectivity requires not more than 4 partitions.

### Appendix B: Proofs

**Proof of Lemma 2** The first equality follows immediately from the definitions of \( \text{vec}_r(\mathbb{I}_n) \) and \( \text{vec}_r(A) \). For the second one, consider the decomposition \( A = \sum_{i,j} A_{i,j} |i\rangle_n \langle j|_n \), then

\[
\left( \mathbb{I}_n \otimes A^\top \right) \text{vec}_r(\mathbb{I}_n) = \left( \mathbb{I}_n \otimes \sum_{i,j} A_{i,j} |i\rangle_n \langle j|_n \right) \sum_k |k\rangle_n \otimes |k\rangle_n
\]

\[
= \sum_{i,j} \left( |i\rangle_n \otimes (A_{i,j} |j\rangle_n) \right) = \text{vec}_r(A).
\]

\[\Box\]

**Proof of Lemma 3** We prove the equality between the left-hand side (LHS) of (5a) and (5b), so

\[
\text{vec}_r(\mathbb{I}_n)^\top (A \otimes B) \text{vec}_r(\mathbb{I}_n) = \left( \sum_i \langle i| \otimes \langle i| \right) (A \otimes B) \left( \sum_j |j\rangle \otimes |j\rangle \right)
\]

\[
= \sum_{i,j} \langle i| A |j\rangle \otimes \langle i| B |j\rangle
\]

\[
= \sum_{i,j} A_{i,j} B_{i,j} = \mathbb{I}_n^\top (A \otimes B) \mathbb{I}_n.
\]

\[\Box\]
Using Lemma 2, we prove the trace identity. First note that, for some $i, j \in [0 \ldots n - 1]$, $|j \otimes (i | I_n | j)\rangle = |i\rangle$ when $i = j$ and 0 otherwise. Then

$$
\text{Tr} \left( AB^\top \right) = \sum_{i=0}^{n-1} \langle i | AB^\top | i \rangle 
$$
(B4a)

$$
= \sum_{i, j=0}^{n-1} \langle i | AB^\top | j \rangle \otimes \langle i | I_n | j \rangle 
$$
(B4b)

$$
= \text{vec}_r (I_n)^\top (AB^\top \otimes I_n) \text{vec}_r (I_n) 
$$
(B4c)

$$
= \text{vec}_r (I_n)^\top (A \otimes I_n)(B^\top \otimes I_n) \text{vec}_r (I_n) 
$$
(B4d)

$$
\equiv \text{vec}_r (I_n)^\top (A \otimes I_n)(I_n \otimes B) \text{vec}_r (I_n) 
$$
(B4e)

$$
= \text{vec}_r (I_n)^\top (A \otimes B) \text{vec}_r (I_n) . 
$$
(B4f)

\(\blacksquare\)

**Proof of Lemma 4** Since $Q$ is doubly stochastic, then $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$. We verify the Definition 1, so

$$
K (J_m \otimes J_m) = \left( \sum_i \lambda_i P_i \otimes P_i \right) (J_m \otimes J_m) 
$$
(B5a)

$$
= \sum_i \lambda_i (P_i J_m) \otimes (P_i J_m) 
$$
(B5b)

$$
= \left( \sum_i \lambda_i \right) J_m \otimes J_m = J_m \otimes J_m . 
$$
(B5c)

Similarly, it can be proved for the left-hand side multiplication by $J_m \otimes J_m$. Thus

$$
K (J_m \otimes J_m) = (J_m \otimes J_m) K = J_m \otimes J_m , 
$$

hence the implication follows. \(\blacksquare\)

**Lemma 11** The partial derivatives of the hardware constraints function (18) w.r.t. $\theta_s ^{(t)}$, follow the parameter shift rule [25], that is

$$
\frac{\partial \mathcal{L}(\theta)}{\partial \theta_s ^{(t)}} = \mathcal{L} \left( \theta + \frac{\pi}{4} e_s ^{(t)} \right) - \mathcal{L} \left( \theta - \frac{\pi}{4} e_s ^{(t)} \right) , 
$$
(B6)

where $e_s ^{(t)}$ is the standard basis vector whose index is $(t, s)$.

**Proof** First note that

$$
\frac{\partial \sin^2(\theta)}{\partial \theta} = 2 \sin \theta \cos \theta = \sin(2\theta) 
$$
(B7a)

$$
= \left( \frac{1}{2} + \frac{1}{2} \sin(2\theta) \right) - \left( \frac{1}{2} - \frac{1}{2} \sin(2\theta) \right) 
$$
(B7b)
\[
\begin{align*}
&\frac{1}{2} - \frac{1}{2} \cos(2\theta + \frac{\pi}{2}) - \left(\frac{1}{2} - \frac{1}{2} \sin(2\theta - \frac{\pi}{2})\right) \\
&\quad = \sin^2\left(\theta + \frac{\pi}{4}\right) - \sin^2\left(\theta - \frac{\pi}{4}\right). 
\end{align*}
\]

Consider the matrix-valued function \(SSWAP_m((\cdot); i, j) : \mathbb{R} \rightarrow \mathcal{M}_m\), then fixed some \(v, w \in \mathbb{R}^m\), we have

\[
\begin{align*}
\frac{\partial}{\partial \theta} \left(v^\top SSWAP_m(i, j, \theta)w\right) &= \frac{\partial}{\partial \theta} \left(v^\top SSWAP_m(i, j)w - v^\top w\right) \\
&= v^\top SSWAP_m(i, j, \theta + \frac{\pi}{4})w - v^\top SSWAP_m(i, j, \theta - \frac{\pi}{4})w, 
\end{align*}
\]

and similarly for \(PSSWAP\). But the LHS of (B8a) corresponds to the form of \(\frac{\partial \mathcal{L}_i(\theta)}{\partial \theta_s(i)}\).

Also, since the variables \(\theta_s(i)\) appear not more than once in each term of the constraint function in (18), then by linearity the result follows. 

\(\square\)

Lemma 12 Let \(Q\) be an \(m \times m\) DSM, then the \(\ell_1\)-norm of vectors from the non-negative orthant \(\mathbb{R}_+^m\) is preserved under the action of \(Q\). That is

\[
\forall v \in \mathbb{R}_+^m \implies \|Qv\|_1 = \|v\|_1, \tag{B9}
\]

for all \(m \times m\) DSMs \(Q\).

Proof Since \(v \in \mathbb{R}_+^m\), so each component \(v_i\) is non-negative, then the \(\ell_1\)-norm can be written as \(\|v\|_1 = 1_m^\top v\). Note that for any \(m \times m\) permutation matrix \(P\), we have \(P1_m = 1_m\). Let \(Q = \sum_j \lambda_j P_j\) be any \(m \times m\) DSM, where \(\lambda_j\) and \(P_j\) follow Definition 1. Hence

\[
\|Qv\|_1 = \left\| \sum_j \lambda_j P_jv \right\|_1 = \sum_j \lambda_j1_m^\top P_jv \tag{B10a} \\
= 1_m^\top v \sum_j \lambda_j = \|v\|_1, \tag{B10b}
\]

consequently the claim is proved. \(\square\)

Proof of Proposition 6 From Lemma 11 and the structure of the function \(\mathcal{L}\), it follows that the partial derivative w.r.t. \(\theta_s(i)\) takes the form

\[
\frac{\partial \mathcal{L}(\theta)}{\partial \theta_s(i)} = \sum_{\tau = l}^{T-1} v_\tau^\top \left(1_m^\otimes 2 \sin^2\left(\theta_s(i) + \frac{\pi}{4}\right) F\right) w_\tau 
\]
where \( \mathbf{v}_\tau, \mathbf{w}_\tau \) are some fixed vectors and \( S_s^{(i)} \) is the swap corresponding to the parameter \( \theta_s^{(i)} \). We note that the vectors \( \mathbf{v}_\tau, \mathbf{w}_\tau \) may depend on the elements of \( \theta \) excluding the selected \( \theta_s^{(i)} \). Equating (B11a) with zero we obtain

\[
\sin^2 \left( \theta_s^{(i)} + \frac{\pi}{4} \right) \sum_{\tau} \left( \mathbf{v}_\tau^T \mathbf{w}_\tau \right) = \sin^2 \left( \theta_s^{(i)} - \frac{\pi}{4} \right) \sum_{\tau} \left( \mathbf{v}_\tau^T \mathbf{w}_\tau \right),
\]

then by (B7d), we immediately see that the latter holds when \( \sin(2\theta_s^{(i)}) = 0 \), that is \( \theta_s^{(i)} \in \Omega \). Consequently, points 1. and 2. follow.

For point 3., it is sufficient to show that \( \mathbf{v}_\tau^T \mathbf{w}_\tau \) can be zero, then the value of \( \theta_s^{(i)} \) is uninfluential. Therefore, the partial derivative w.r.t. the same variable is zero, independently from the value of \( \theta_s^{(i)} \).

Vectors \( \mathbf{v}_\tau, \mathbf{w}_\tau \) are the result of some DSM applied to the non-zero vectors, with non-negative entries, \( \text{vec}_c \left( G^{(i)} \right) \) and \( \text{vec}_c \left( M_c \right) \). By Lemma 12 the DSM action on those vectors preserves the \( \ell_1 \)-norm, then \( \mathbf{v}_\tau \neq \mathbf{0}, \mathbf{w}_\tau \neq \mathbf{0} \). Thus \( \mathbf{v}_\tau^T \mathbf{w}_\tau = 0 \) only as a result of the action of \( F \). We show that there exist basis vectors \( |a\rangle_m, |b\rangle_m \) such that the latter equality holds. Assume \( S_s^{(i)} = \text{SWAP}_m(i, j) \) for some \( i \neq j \) such that \( i, j \) are distinct from \( a, b \), then

\[
\langle a | \otimes^2 F | b \rangle \otimes^2 = \langle a | \otimes^2 \text{SWAP}_m(i, j) \otimes^2 | b \rangle \otimes^2 = \langle a | \otimes^2 | b \rangle \otimes^2 = 0,
\]

since \( \text{SWAP}_m(i, j) \) fixes \( |a\rangle_m, |b\rangle_m \). Hence there exist \( \mathbf{v}_\tau, \mathbf{w}_\tau \) such that \( \mathbf{v}_\tau^T \mathbf{w}_\tau \) vanishes, so point 3. is proved. \( \square \)

**Lemma 13** Consider the \( C^1 \) function \( g : \mathbb{R}^n \rightarrow \mathbb{R}_+ \), such that \( g(x^*) = 0 \implies x^* \in \mathbb{Z}^n \) and \( g(x + 2\mathbf{e}_i) = g(x) \) for all \( i = 0 \ldots n - 1 \), with \( \{\mathbf{e}_i\} \) the canonical basis for \( \mathbb{R}^n \). In other words the zeros of \( g \) occur at points having integer entries, additionally the function is entry-wise periodic with fundamental period 2. Then the following optimization problems are equivalent

\[
\begin{cases}
\min_{\mathbf{w} \in \mathbb{R}^n} \|\mathbf{w}\|_2, \\
\text{s.t.} \quad g(\mathbf{w}) = 0,
\end{cases}
\]  
\[
\cong \begin{cases}
\min_{\mathbf{w} \in \mathbb{R}^n} \text{card}(\mathbf{w}), \\
\text{s.t.} \quad g(\mathbf{w}) = 0,
\end{cases}
\]

where \( \text{card}(\cdot) \) denotes the cardinality of the argument.

**Proof** A consequence of the periodicity is that for any \( \mathbf{x} \in \mathbb{R}^n \) and \( \mathbf{y} = (2 \sum_i k_i \mathbf{e}_i) \in 2\mathbb{Z}^n \), with \( k_i \in \mathbb{Z} \), we have \( g(\mathbf{x} + \mathbf{y}) = g(\mathbf{x}) \). \( \square \)
Let $x^* \in \mathbb{Z}^n$ fixed, and consider the projection\(^{17}\) $\mathcal{P}_{2\mathbb{Z}}$ of $x^*$ onto the set $2\mathbb{Z}$, so let

$$y^* \in \mathcal{P}_{2\mathbb{Z}}(x^*) = \arg\min_{y \in 2\mathbb{Z}^n} \left\{ \|x^* - y\|_2^2 \right\}, \quad (B15)$$

then $x^* - y^* \in \{-1, 0, 1\}^n$, so

$$\|x^* - y^*\|_2^2 = \sum_i (x^*_i - y^*_i)^2 = \sum_i |x^*_i - y^*_i| \quad (B16a)$$

$$= |x^* - y^*|_1 \quad (B16b)$$

$$= \text{card}(x^* - y^*) \quad (B16c)$$

Now, consider the first optimization problem in (B14) and perform the substitution $w = x - y$, to obtain

$$\min_{x \in \mathbb{R}^n, y \in 2\mathbb{Z}^n} \|x - y\|_2^2, \quad \text{s.t. } g(x - y) = g(x) = 0. \quad (B17)$$

Let $G = \{x \in \mathbb{R}^n | g(x) = 0\}$ be the set of feasible points, then by assumption, $x \in G \implies x \in \mathbb{Z}^n$, that is $G \subset \mathbb{Z}^n$. Since the constraint is independent from the variable $y$, we rewrite the optimization problem considering the feasible set, so

$$\min_{x \in G, y \in 2\mathbb{Z}^n} \|x - y\|_2^2, \quad \cong \min_{y \in 2\mathbb{Z}^n} \min_{x \in G} \|x - y\|_2^2, \quad (B18)$$

then by (B15) and (B16a), the optimal $y^*$ is the one such that $x^* - y^* \in \{-1, 0, 1\}^n$ and the cardinality of $x^* - y^*$ is minimized. Hence the equivalence in (B14) is established.

\[\square\]

**Proof of Proposition 5** We sketch a proof based on Lemma 13. The aforementioned lemma can be adapted to the periodicity of the hardware constraints function $L$. It follows from (16b) that the period of $L$ is $\pi$. Define the function

$$g(x) = L \left( \frac{\pi x}{2} \right), \quad (B19)$$

which has period 2. Also the cardinality function is invariant to a non-zero vector scaling, that is $\text{card}(x) = \text{card}(\alpha x)$ for all $\alpha \in \mathbb{R} \setminus \{0\}$. Then by applying Lemma 13, the claim follows.

\[\square\]

\(^{17}\) We highlight a technicality regarding the projection $\mathcal{P}_{2\mathbb{Z}}$. Since the set $2\mathbb{Z}^n$ is non-empty closed but non-convex, then we cannot assure that the projection is a singleton [15, first projection theorem], consequently we assume $\mathcal{P}_{2\mathbb{Z}}$ to be a set-valued operator, with the non-emptiness resulting from the set $2\mathbb{Z}^n$ being non-empty and close.
Proof of Lemma 8 Under the assumptions that $P_a^2 = P_b^2 = I_m$ and $P_aP_b = P_bP_a$ we obtain that $(P_aP_b)^2 = P_aP_bP_aP_b = P_aP_bP_bP_a = I_m$, then we need that

$$(P_a + P_b - I_m)^2 = I_m + 2[P_aP_b - (P_a + P_b - I_m)] = I_m,$$

(B20)

which is true if and only if $P_aP_b = (P_a + P_b - I_m)$. The latter being the claim proves the lemma. \(\square\)

Proof of Proposition 9 First we note that, for $m\times m$ permutation matrices $P_a, P_b$, fulfilling the conditions of Lemma 8, then

$$(P_a - I_m)(P_b - I_m) = P_aP_b - P_a - P_b + I_m \overset{(27)}{=} 0.$$  

(B21)

In other words, given any pair of distinct, commuting and involutory permutations (acting on the same vector space), the product of their shift by the identity vanishes.

We proceed by induction. Equation (29) is clearly true for $n = 1$, furthermore, assume it is true for an arbitrary $n \geq 1$, then consider the case $n+1$ by multiplying both sides of equation (29) by $I_m + \sin^2(\theta_{n+1}) (P_{n+1} - I_m)$ to get

$$
\prod_{t=1}^{n+1} \left( I_m + \sin^2(\theta_t) (P_t - I_m) \right) = I_m + \sum_{t=1}^{n+1} \sin^2(\theta_t) (P_t - I_m) + \sum_{t=1}^{n} \sin^2(\theta_t) \sin^2(\theta_{n+1}) (P_t - I_m) (P_{n+1} - I_m) \overset{0 \text{ by } (B21)}{=} 0
$$

(B22a)

$$
I_m + \sum_{t=1}^{n+1} \sin^2(\theta_t) (P_t - I_m) , \quad \text{(B22c)}
$$

hence the claim is proved. \(\square\)

References

1. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum 2, 79 (2018). https://doi.org/10.22331/q-2018-08-06-79
2. Martinis, J., Boixo, S.: Quantum supremacy using a programmable superconducting processor. https://ai.googleblog.com/2019/10/quantum-supremacy-using-programmable.html. Accessed 12 May 2023
3. Chow, J., Gambetta, J.: Quantum takes flight: moving from laboratory demonstrations to building systems. https://www.ibm.com/blogs/research/2020/01/quantum-volume-32/. Accessed 12 May 2023
4. Maslov, D., Falconer, S.M., Mosca, M.: Quantum circuit placement. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 27(4), 752–763 (2008)
5. Nannicini, G., Bishop, L.S., Günlük, O., Jurcevic, P.: Optimal qubit assignment and routing via integer programming. ACM Trans. Quantum Comput. 4(1) (2022). https://doi.org/10.1145/3544563
6. Magesan, E., Gambetta, J.M., Emerson, J.: Characterizing quantum gates via randomized benchmarking. Phys. Rev. A 85(4) (2012). https://doi.org/10.1103/physreva.85.042311
7. Siraichi, M.Y., Santos, V.F.D., Collange, C., Quintão Pereira, F.M.: Qubit allocation. In: CGO 2018—
International Symposium on Code Generation and Optimization, Vienna, Austria, pp. 1–12 (2018).
https://doi.org/10.1145/3168822. https://hal.archives-ouvertes.fr/hal-01655951
8. Li, G., Ding, Y., Xie, Y.: 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, pp. 1001–1014 (2019)
9. Qiskit: An Open-source Framework for Quantum Computing (2021). https://doi.org/10.5281/zenodo.
2573505
10. Meijer-van de Griend, A., Li, S.M.: Dynamic qubit allocation and routing for constrained topologies
by cnot circuit re-synthesis. Quantum Phys. Logic (2022)
11. Zhu, P., Cheng, X., Guan, Z.: An exact qubit allocation approach for nisq architectures. Quantum Inf.
Process. 19(11) (2020). https://doi.org/10.1007/s11128-020-02901-4
12. van Houte, R., Mulderij, J., Attema, T., Chiscop, I., Phillipson, F.: Mathematical formulation of quantum
circuit design problems in networks of quantum computers. Quantum Inf. Process. 19(5), 141 (2020).
https://doi.org/10.1007/s11128-020-02630-8
13. Tan, B., Cong, J.: Optimality study of existing quantum computing layout synthesis tools. IEEE Trans.
Comput. 70(9), 1363–1373 (2021). https://doi.org/10.1109/tc.2020.3009140
14. Brualdi, R.A.: Combinatorial matrix classes. Encyclopedia of Mathematics and its Applications. Cam-
bridge University Press, Cambridge (2006)
15. Beck, A.: First-Order Methods in Optimization. SIAM-Society for Industrial and Applied Mathematics,
Philadelphia (2017)
16. Horn, R.A., Johnson, C.R.: Matrix Analysis, 2nd edn. Cambridge University Press, USA (2012)
17. Quantum, I.: The IBM Quantum heavy hex lattice. Available at https://research.ibm.com/blog/heavy-
hex-lattice. Accessed 7 July 2021
18. Sagan, B.E.: The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Func-
tions. Springer, New York (2001)
19. Palma-Behnke, R., Benavides, C., Lanas, F., Severino, B., Reyes, L., Llanos, J., Sáez, D.: A microgrid
energy management system based on the rolling horizon strategy. IEEE Trans. Smart Grid 4(2), 996–
1006 (2013). https://doi.org/10.1109/TSG.2012.2231440
20. Gambella, C., Malaguti, E., Masini, F., Vigo, D.: Optimizing relocation operations in electric car-
sharing. Omega 81, 234–245 (2018). https://doi.org/10.1016/j.omega.2017.11.007
21. Platt, J., Barr, A.: Constrained differential optimization. In: Anderson, D. (ed.) Neural Information
Processing Systems, vol. 0. American Institute of Physics (1987). https://proceedings.neurips.cc/paper/
1987/file/a87ff679a2f3e71d9181a67b7542122c-Paper.pdf
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein,
N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chil-
amkuryth, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: an imperative style, high-performance
deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d’ Alché-Buc, F., Fox, E.,
Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Cur-
ran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-
performance-deep-learning-library.pdf
23. Cross, A.W., Bishop, L.S., Sheldon, S., Nation, P.D., Gambetta, J.M.: Validating quantum computers
using randomized model circuits. Phys. Rev. A 100(3) (2019). https://doi.org/10.1103/physreva.100.
032328
24. Vizing, V.G.: Critical graphs with given chromatic class. Metody Diskret. Analiz. 5, 9–17 (1965)
25. Crooks, G.E.: Gradients of parameterized quantum gates using the parameter-shift rule and gate decom-
position (2019). https://doi.org/10.48550/ARXIV.1905.13311

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps
and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under
a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted
manuscript version of this article is solely governed by the terms of such publishing agreement and applicable
law.