# Variational quantum algorithms for permutation-based combinatorial problems: Optimal ansatz generation with applications to quadratic assignment problems and beyond

Dylan Laplace Mermoud[1,2], Andrea Simonetto[1,2], and Sourour Elloumi[1,2,3]

[1]UMA, ENSTA, Institut Polytechnique de Paris, 91120 Palaiseau, France
[2]CEDRIC, Conservatoire National des Arts et Métiers, 75003 Paris, France
[3]ensIIE, École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise, 91025, Évry, France

**Abstract.** We present a quantum variational algorithm based on a novel circuit that generates all permutations that can be spanned by one- and two-qubits permutation gates. The construction of the circuits follows from group-theoretical results, most importantly the Bruhat decomposition of the group generated by the `cx` gates. These circuits require a number of qubits that scale logarithmically with the permutation dimension, and are therefore employable in near-term applications. We further augment the circuits with ancilla qubits to enlarge their span, and with these we build ansatze to tackle permutation-based optimization problems such as quadratic assignment problems, and graph isomorphisms. The resulting quantum algorithm, QuPer, is competitive with respect to classical heuristics and we could simulate its behavior up to a problem with 256 variables, requiring 20 qubits.

Dylan Laplace Mermoud: dylan.laplace@ensta.fr, This research benefited from the support of the FMJH Program PGMO, project number 2023-0009, and from the ANR project HQI-ANR-22-PNCQ-0002.

## Contents

# 1 Introduction

Combinatorial optimization has been advocated as one of the key benchmarks for quantum computing: its classically hard nature combined with its wide societal importance makes it an attractive area of research for quantum algorithms. One of such algorithms is the celebrated quantum approximate optimization algorithm, or QAOA, specifically designed for noisy intermediate-scale quantum computers (or NISQ) [1, 2], which now exists in many variants, see [3] for an account. Other algorithms have been also proposed in other noise regimes, but they will not be our main focus in this paper.

Even though QAOA has been widely studied, its performance on hard combinatorial problems is still under scrutiny. Theoretically, we know that QAOA is not classically simulable, and it may have some, albeit limited, advantage with respect to some classical algorithm [4, 5, 6]; however, more broadly, QAOA is not readily expected to solve relevant combinatorial problems at scale. There are at least two main issues that would challenge QAOA to achieve that.

First, despite their hardness, there exist methods performing well at solving combinatorial problems classically, or at least at finding approximate solutions. As Berstimas and Dunn describe in a recent paper, *"in the last 25 years, algorithmic advances in integer optimization coupled with hardware improvements have resulted in an astonishing 800 billion factor speedup in mixed-integer optimization"* (a class of hard combinatorial problems) [7]. This translates, for example, into solving some sparse instances of quadratic unconstrained binary optimization problems (QUBOs) exactly up to tens of thousands of variables [8]. QAOA would require at least tens of thousands of logical qubits to achieve a similar feat.

Second, in general, QAOA cannot handle constraints in an hard form. We are forced to soft-constrain the problem by lifting the constrains into the cost, thereby creating numerical issues and approximations. Constraints are at the heart of optimization: we model and then simplify optimization problems by adding constraints of all sorts.

In order to overcome the first issue, mainly due to the basis encoding of the search space in QAOA, we have seen some interest in amplitude encodings [9, 10]. Amplitude encodings require a number of qubits that scales logarithmically with the number of the variables of the optimization problem, thereby being more resource-friendly. Amplitude encodings present other issues, however. The fact that they encode directly vector components and matrices yield typically non-diagonal observables. Furthermore, in most cases, they are trivially classically simulable. Take for example the max-cut formulation in [11]. If $n$ indicates the number of decision variables of a max-cut formulation, then the approach requires only $O(\log n)$ qubit to run. However, the computation of the observable, i.e., the cost, requires $O(n^2)$ calls to the quantum circuit and a total number of $O(n^3)$ floating operations (cf. Appendix A). To run the same heuristic classically, without quantum computations, we would only require $O(n^2)$ floating point operations to evaluate the cost.

In contrast, in [12] the authors propose an amplitude encoding of the subgraph isomorphism problem. This is a genuine quantum algorithm, requiring $O(\log(n))$ qubits to represent a problem in $n$ variables, and with a circuit delivering directly the cost function (the observable is the projector on the first basis state), so a simple average of a number of samples is required classically to retrieve the expected value of the quantum cost. In contrast, a solely classical approach would require $O(n^2)$ floating point to compute the cost. So dealing with amplitude encodings requires great care.

In order to tackle the second issue (the constraints), different variations of QAOA have been proposed, in which we encode in one way or another some constraints into the variational circuit to optimize over. A popular approach is to encode the constraints

into the mixer Hamiltonian [13, 14], which works well for some constraints but might be hard in general [15]. Another approach is to map the feasible space and define a continuous quantum walk that transverses this space [16]. Other approaches are presented in [17, 18, 19]. All these approaches are constraint-dependent and either hard to generalize, to implement, or their accuracy is limited. For example, mapping the whole feasible space classically may even be as hard as solving the original combinatorial problem.

With this in mind, in this paper, we depart from standard QAOA with the aim to develop variational quantum algorithms that can tackle one typical combinatorial constraint at scale. We focus here on permutation-based binary optimization problems [20]: problems which are defined over permutations, which are widespread in operations research. For example, the quadratic assignment problem, the graph isomorphism problem, and the travel salesperson problem are permutation-based optimization problems. We argue that these problems provide a very useful baseline to construct more complicated problems.

Besides their usefulness, we focus here on permutations, since they have quite a natural quantum structure. Permutations are in fact unitary matrices and they can be encoded in quantum gates natively. Furthermore, we will see that permutations have a number of useful mathematical properties that makes them the ideal candidate for quantum circuits.

To try to overcome some of the drawbacks of QAOA, instead of interpreting the cost as observable or encoding it in a quantum circuit, we encode directly a permutation ansatz. The circuit, when measured, directly delivers a doubly-stochastic matrix, i.e., a positive superposition of permutation matrices. This has several advantages: first, the cost can be computed classically and can be anything. This renders the approach extremely general, which allows us to test it on various problems. Second, we can use amplitude encodings to reduce the number of used qubits without having the problem of a non-diagonal observable. This win-win setting is also made non classically simulable by adding additional ancilla qubits, following the very clever circuit design of Mariella and coauthors [21].

More specifically, in this paper, we offer the following contributions.

1. We study variational quantum circuits that can span a given permutation set. In particular, we construct a quantum circuit made of rotations and parametrized `cx` gates that can deliver *all* the permutations available by using these local gates alone. For a number of qubits $q = \log_2 n$ if $n$ is the permutation size, the circuit has $O(q^2)$ classical parameters, $O(q)$ depth, and spans $2^{O(q^2)}$ permutations.
2. We propose a quantum circuit that can bridge the gap between the maximum span and the total number of permutations, i.e., $2^{O(q^2)}$ vs. $(2^q)!$, by using ancilla qubits. We study how the ancilla qubits can deliver doubly-stochastic matrices, their maximum span, and how to construct permutations with them.
3. We test the circuit on several optimization problems of wide-reaching interest proposing QUPER, a variational quantum algorithm for permutation-based optimization problems. In particular, we tackle the quadratic assignment problem and the graph isomorphism. We show that we are competitive with classical heuristic on small and sometimes bigger instances with $n = 256$ variables, and we perform better than other variational quantum algorithms, when they exist.

## 1.1 Closely related work

Classically, permutation-based problems have received a lot of attention. We will report in the sections related to each problem that we simulate an account of the literature for that specific problem. However, we can already refer to [22] as a typical convex approach for the

this general class of optimization problems. This approach would convexify the constraints, by letting the permutation being a doubly-stochastic matrix, solve the problem exactly, and then project the solution onto the permutation set. This particular approach can offer good approximate solutions when the cost is convex in the permutation variable. Our quantum approach is somewhat similar, since it generates doubly-stochastic matrices, but it is a non-convex heuristic, instead of being convex, and it can be applied to a large variety of problems. Of [22], we keep the projection-onto-permutations strategy [23], which is standard in classical optimization and which we will describe in details.

The permutation circuit construction that we propose is largely inspired from the circuit of [21]. There, Mariella and coauthors propose a non-classically simulable circuit to generate doubly-stochastic matrices that they then use to solve an optimal transport problem. In this paper, we study with great care which unitary to use within the circuit to span as much as possible the permutation set, and we prove additional results on the span of the obtained double-stochastic matrices. We see that this is not a trivial question and different choices of the unitary lead to very different results.

In order for us to study the span of certain unitaries that we construct based on rotation and parametrized `cx` gates, we make heavy use of group theory concepts, such as semi-direct products and the Bruhat decomposition. Here, we build on the work of [24], who studied quantum circuits of `cx` gates and their span. But then we largely generalize his work to rotation gates, parametrized `cx`s, and their combinations. We will provide specific references as we go along. As a side note, the Bruhat decomposition has been also used in the context of stabilizer circuits, where the decomposition of the symplectic groups allows for designed circuits of reduced depth [25].

## 2 General problem formulation and main results

We are interested in solving the problem

$$\min_{P \in \Pi_n} f(P), \tag{1}$$

where $\Pi_n$ is the set of permutation matrices of dimension $n \times n$, and $f : \Pi_n \to \mathbb{R}$ is a cost function to minimize. We let $n = 2^q$ for any integer $q$, without loss of generality. Problem (1) is binary, combinatorial, and in general NP-Hard. When $f$ is linear in $P$, then the problem reduces to the linear assignment problem, a convex optimization problem that is easily solvable at scale, e.g., with the Auction algorithm [26]. In most of the other cases, Problem (1) is very hard to solve classically even for $n$ in the order of a few hundreds. In fact, the feasible space for (1), that is the number of permutations of a given order $n$, is $n!$, which grows faster than an exponential.

### 2.1 Spanning permutations via a parametric circuit

To tackle Problem (1), we turn to variational quantum algorithms. In particular, we will devise a parametric quantum circuit $\mathscr{P}(\theta)$ that will be able to represent a large subset of the whole permutation set. When measuring the circuit, we will obtain the coefficients of a doubly-stochastic matrix $\hat{P}_\theta$, that can then be used classically to optimize a certain cost $f(\hat{P}_\theta)$. On the classical side, we will be using a classical optimizer (hereunder Adam [27]) to optimize for $\theta$, while asking to the quantum circuit the value of $\hat{P}_\theta$ for each given $\theta$. During the run, we have the possibility to classically project the obtained doubly-stochastic matrix $\hat{P}_\theta$ onto the permutation set, thereby obtaining an approximate solution $\tilde{P}$.

The advantage of the approach proposed in this paper lies in the representation of the quantum circuit $\mathscr{P}(\theta)$, which allows us to span a larger set of the permutation set with a limited number of continuous parameters $\theta$. We will see that this circuit is, in general, non classically simulable, thereby granting our variational quantum algorithm the label of a genuine quantum approach.

We report in Figure 1, the aforementioned approach. We remark that $\mathscr{P}(\theta)$ is designed as such that permutations are encoded in amplitudes, so that the number of required qubits will scale only logarithmically with the permutation set $n$. $\mathscr{P}(\theta)$ will be also augmented by an arbitrary even number $2m$ of ancilla qubits to guarantee both a larger span and a non-trivially classically simulable circuit.

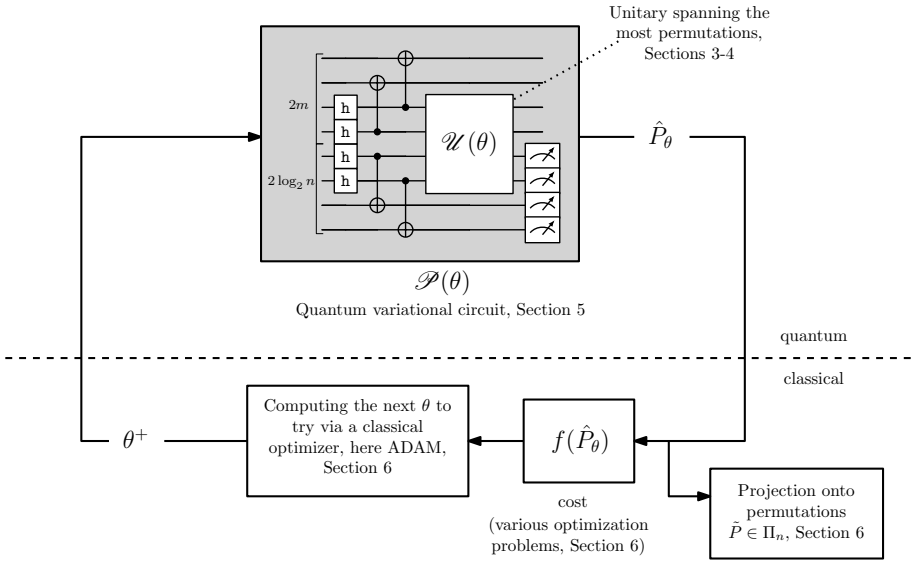


Figure 1: Pictorial description of the proposed variational approach, labeled QuPer, with the reference to all the sections where each element is discussed.

The first important result that we will discuss in great detail pertains how much of the permutation set we can span with only rotation and parametrized `cx` gates and how to construct such a circuit. These gates are the basic gates that we will use in our construction, since readily available in most quantum hardware. The result is given informally below.

**Theorem 1** (Informal)**.** *Consider a $q$-qubit circuit whose unitary is constructed to represent a $n$-dimensional permutation.*

*(i) The permutations spanned only by using rotation and parametrized* `cx` *gates can be computed by considering the composition of two circuits: one containing rotations only and one containing parametrized* `cx` *gates only.*

*(ii) Any circuit containing parametrized* `cx` *gates only can be decomposed into three circuits, by the Bruhat decomposition.*

*(iii) The circuit spanning all the permutations available by considering only rotations and parametrized* `cx` *has $O(q^2)$ parameters, $O(q)$ depth, and spans $2^{O(q^2)}$ permutations.*

Theorem 1 will be made formal in Sections 3 and 4, and specifically by Propositions 3 and 5 as well as Theorems 3, 4, and 6. Theorem 1 informs us that to obtain the maximum span that local gates have to offer, we only need a number of classical parameters that scales quadratically in the number of qubits and a linear depth. We will construct such circuit, see for instance Figure 9.

Naturally, $2^{O(q^2)}$ is lower than the total number of permutations $(2^q)! = n!$. The second important result is how to bridge this gap by using $m$ ancilla qubits. We will make use of the very clever circuit design of Mariella and coauthors [21], to generate doubly-stochastic matrices. We will present such circuit in Figure 8 and we will study in details the choice of ansatze to use. In addition, we will be able to show that with our ansatze for permutations, we can span at most $2^{O((q+m)^2)}$ permutations, and therefore we will need at least $m \gtrsim \sqrt{n \log n}$ ancilla to span the whole permutation set of $n!$. We will make this formal in Proposition 8.

### 2.2 Solving optimization problems

Given the circuit and the ansatze that we have designed, we will then set forth to solve permutation-based optimization problems, such as the quadratic assignment problem and the graph isomorphism. We will propose a novel variational algorithm, labelled QuPer, which we will show competitive with respect to classical approaches, at least in small instances but not only, and with which we will be able to simulate problems up to $n = 256$ variables in 20 qubits.

### 2.3 Organization of the paper

The remainder of the paper is organized as follows. In Sections 3 and 4, we discuss how to construct quantum circuits representing permutations: these are the most mathematically heavy sections. We then propose the variational circuit $\mathscr{P}(\theta)$ in Section 5 and analyze its theoretical and numerical properties. In Section 6, we present numerical results for two optimization problems of widespread interest: the quadratic assignment problem and the graph isomorphism. We then conclude.

## 3 Spanning permutations

### 3.1 Preliminaries

The first step of our variational quantum circuit construct is a mathematical one. Given simple local gates, hereunder $\mathtt{x}$ and $\mathtt{cx}$, how many distinct permutation matrices can we construct? We will see how this is important in our variational circuit in Section 5.

As such, we start by defining a quantum circuit acting on $q$ qubits, and the the gates $\mathtt{x}$ and $\mathtt{cx}$, defined by

$$\mathtt{x}_k = |0\rangle\langle 1|_k + |1\rangle\langle 0|_k \qquad \text{and} \qquad \mathtt{cx}_{kl} = |0\rangle\langle 0|_k \otimes \mathtt{id}_l + |1\rangle\langle 1|_k \otimes \mathtt{x}_l.$$

The first gate $\mathtt{x}_k$ acts on the single qubit $k$, while the second gate $\mathtt{cx}_{kl}$ acts on two qubits, $k$ and $l$, respectively. Hence, $\mathtt{x}$ exchanges the two amplitudes of a one-qubit state, and $\mathtt{cx}$ exchanges the amplitudes of the state of the $l$th qubit if the state of the $k$th qubit is $|1\rangle$. These two gates therefore act as permutation matrices on the space $\left(\mathbb{C}^2\right)^{\otimes q}$ of states of $q$-qubit systems.

We need now to introduce some useful group theory concepts that will help us building and counting distinct permutations. The reader is also referred to [28, 29] for good introductions on the subject.

For any group, one can define the generators of the group, as a subset of elements of the group that can generate the whole group. More formally, a set of *generators* of a given

group $G$ is a subset $S \subseteq G$ such that any element of the group can be written as a word of generators, and their inverses. Together with a set of *relations* $R$ that the elements of the group, written as words of generators, satisfy, the set of generators characterizes the group $G$. The pair $\langle\!\langle S|R \rangle\!\rangle$ is called a *presentation*[1] of the group $G$.

The group of all bijections of $\{1, \ldots, n\}$, equipped with the law of composition, is called the *symmetric group*, here written as $S_n$. The elements of $S_n$ are called *permutations*. A well-known presentation for the symmetric group is the one given by *adjacent transpositions*: i.e., any permutation can be generated by permuting (swapping) adjacent elements. Since we will make use of this, we describe it in details in the following example.

**Example 1** (Symmetric group $S_n$)**.** A well-known presentation for the symmetric group is the one given by adjacent transpositions. Indeed, any permutation can be written as a word of adjacent transpositions. For instance, the permutation $(3\,2\,1)$[2] which reverses the order of three elements, can be written as

$$(3\,2\,1) = (2\,1\,3)(1\,3\,2)(2\,1\,3).$$

Usually, the transposition $(\cdots\ k+1\ k\ \cdots)$ is denoted by $\sigma_k$. The set of relations defining the symmetric group from these generators are, for all $1 \leq k, j \leq n-1$ such that $j \neq k \pm 1$,

$$\sigma_k^2 = \mathrm{id}, \qquad \sigma_k \sigma_j = \sigma_j \sigma_k, \qquad \text{and} \qquad \sigma_k \sigma_{k+1} \sigma_k = \sigma_{k+1} \sigma_k \sigma_{k+1}.$$

The last set of relations can be transformed into $(\sigma_k \sigma_{k+1})^3 = \mathrm{id}$ using $\sigma_k^2 = \mathrm{id}$, where id represents the identity. ◊

The language of group theory using presentations can be directly transposed in the language of quantum circuits, at the very least for three reasons. First, a quantum circuit can be interpreted as a word of tensor products of basic gates. Secondly, the concept of generators is tightly connected to the one of basic gates: in order to write efficient circuits for the group under consideration, the generators must be written efficiently, as short words of basic gates. Finally, the relations give rewriting rules to design shorter circuits.

We consider now the group generated by `x` gates alone.

## 3.2 The group generated by `x` gates

Let $X_q$ denote the group generated by $\{\mathtt{x}_k \mid 0 \leq k \leq q-1\}$. The group is abelian (i.e., any pair of elements commutes), and contains $2^q$ elements. Because each generator only applies to one qubit and is an involution, any element of the group can be seen as a tuple of generators, exhibiting the direct sum structure of the group:

$$X_q \simeq \bigoplus_{k=0}^{q-1} \mathbb{Z}_2 =: \mathbb{Z}_2^q.$$

Here, $\mathbb{Z}_2$ denotes the group defined on the set $\{0, 1\}$ equipped with the bitwise 'exclusive-or' additive law $\oplus$ defined by

$$1 \oplus 1 = 0 \oplus 0 = 0, \qquad \text{and} \qquad 1 \oplus 0 = 0 \oplus 1 = 1.$$

[1] We use here the notation $\langle\!\langle \cdot|\cdot \rangle\!\rangle$ instead of the more common $\langle \cdot|\cdot \rangle$ to avoid confusion with the inner product of two quantum states.

[2] We used here the *one-line* notation for permutations. The permutation $(\sigma_1 \cdots \sigma_i \cdots \sigma_n)$ sends $i$ to $\sigma_i$.

A presentation for $X_q$ is therefore given by

$$\langle\!\langle \mathtt{x}_0, \ldots, \mathtt{x}_{q-1} \mid \mathtt{x}_k^2 = \mathtt{id} \text{ and } (\mathtt{x}_k\mathtt{x}_l)^2 = \mathtt{id}, \text{ for all } 0 \leq k, l \leq q-1 \rangle\!\rangle.$$

It is customary in group theory to write the relators as words of generators that correspond to the identity. However, this expression can be rewritten as follows. We can apply the exponentiation to get $\mathtt{x}_k\mathtt{x}_l\mathtt{x}_k\mathtt{x}_l = \mathtt{id}$. By multiplying on both sides by $\mathtt{x}_l\mathtt{x}_k$, and remembering that each $\mathtt{x}$ is an involution, i.e., its own inverse, we obtain $\mathtt{x}_k\mathtt{x}_l = \mathtt{x}_l\mathtt{x}_k$, so $\mathtt{x}_k$ and $\mathtt{x}_l$ commute.

### 3.3 The group generated by cx gates

Let $CX_q$ denote the group generated by $\{\mathtt{cx}_{kl} \mid k \neq l, 0 \leq k, l \leq q-1\}$. This group is no longer abelian, because we have

$$\mathtt{cx}_{kl}\mathtt{cx}_{lm} \neq \mathtt{cx}_{lm}\mathtt{cx}_{kl}, \qquad \text{for } k, l, m \text{ all distinct.}$$

A presentation of $CX_q$ was identified by Bataille [24].

**Proposition 1** ([24])**.** *A presentation for the group $CX_q$ is given by the set of generators $\{\mathtt{cx}_{kl} \mid k \neq l, 0 \leq k, l \leq q-1\}$ and the relations*

- *involution:* $\mathtt{cx}_{kl}^2 = \mathtt{id}$ *where* $k \neq l$,
- *commutation:* $(\mathtt{cx}_{kl}\mathtt{cx}_{mp})^2 = \mathtt{id}$ *where* $k \neq p$ *and* $l \neq m$,
- *non-commutation:* $(\mathtt{cx}_{kl}\mathtt{cx}_{lm})^2 = \mathtt{cx}_{km}$ *where* $k, l, m$ *are distinct.*

Let $\mathrm{GL}_n(\mathbb{F}_2)$ be the group of invertible $(n \times n)$-matrices over the field $\mathbb{F}_2$. Steinberg [30] gave a presentation of it that perfectly coincides with the one of $CX$ given by Bataille [24].

**Proposition 2** ([30, 24])**.** *The group $CX_q$ is isomorphic to $\mathrm{GL}_q(\mathbb{F}_2)$.*

This result will allow us to leverage the properties of $\mathrm{GL}_q(\mathbb{F}_2)$, to implement the full $CX_q$ in quantum circuits. In particular, Bruhat and Tits [31] demonstrated that any element of $\mathrm{GL}_q(\mathbb{F}_2)$ can be written as the product of three elements, belonging to specific subgroups of $\mathrm{GL}_q(\mathbb{F}_2)$. Let us see how.

Let $B$ denote the subgroup of $\mathrm{GL}_q(\mathbb{F}_2)$ only composed of upper-triangular matrices, and let $W$ denote the subgroup only composed of permutation matrices.

**Theorem 2** (The Bruhat decomposition [31, 32])**.** $\mathrm{GL}_q(\mathbb{F}_2)$ *is isomorphic to $BWB$.*

The Bruhat decomposition will be the cornerstone of our construction. It allows us to decompose the circuit synthesis problem into two, much simpler, circuit synthesis problems.

### 3.4 The group generated both by x and cx gates

In this subsection, we present *new results* about the structure of the whole group generated by $\mathtt{x}$ and $\mathtt{cx}$ gates, and explain how it relates to the two groups described earlier. This group includes all the circuits on $q$ qubits composed solely of $\mathtt{x}$ and $\mathtt{cx}$ gates, up to rewriting rules, i.e., relations, that we present in the following.

Let $LX_q$ denote the group generated by

$$\{\mathtt{x}_k \mid 0 \leq k \leq q-1\} \cup \{\mathtt{cx}_{kl} \mid k \neq l, 0 \leq k, l \leq q-1\},$$

the set of all local permutation gates. Necessarily, it contains both $X_q$ and $CX_q$ as a subgroup, and therefore is subject to the same relations as they do. However, new elements generated both by $\mathtt{x}$ and $\mathtt{cx}$ are generated, which are subject to new relations. For instance, we know that any $\mathtt{cx}$ gate commutes with a $\mathtt{x}$ gate that acts on different qubits. The behavior of these gates is however different when they act on the same qubits.

Consider the following. Let $\mathtt{x}_1$ and $\mathtt{cx}_{12}$ act on each of the state $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ in both orders. Then, simple computations lead to

$$\begin{aligned}
\mathtt{cx}_{12}\mathtt{x}_1 |00\rangle &= \mathtt{cx}_{12} |10\rangle = |11\rangle, & \mathtt{x}_1\mathtt{cx}_{12} |00\rangle &= \mathtt{x}_1 |00\rangle = |10\rangle, \\
\mathtt{cx}_{12}\mathtt{x}_1 |01\rangle &= \mathtt{cx}_{12} |11\rangle = |10\rangle, & \mathtt{x}_1\mathtt{cx}_{12} |01\rangle &= \mathtt{x}_1 |01\rangle = |11\rangle, \\
\mathtt{cx}_{12}\mathtt{x}_1 |10\rangle &= \mathtt{cx}_{12} |00\rangle = |00\rangle, & \mathtt{x}_1\mathtt{cx}_{12} |10\rangle &= \mathtt{x}_1 |11\rangle = |01\rangle, \\
\mathtt{cx}_{12}\mathtt{x}_1 |11\rangle &= \mathtt{cx}_{12} |01\rangle = |01\rangle, & \mathtt{x}_1\mathtt{cx}_{12} |11\rangle &= \mathtt{x}_1 |10\rangle = |00\rangle.
\end{aligned}$$

As anticipated, we have that $\mathtt{x}_1$ and $\mathtt{cx}_{12}$ do not commute. However, we can see that the results obtained differ exactly by a bitflip on the rightmost bit. Therefore, we can write

$$\mathtt{cx}_{12}\mathtt{x}_1\mathtt{x}_2 = \mathtt{x}_1\mathtt{cx}_{12}.$$

We obtain similar result when applying $\mathtt{x}_2$ or $\mathtt{x}_1\mathtt{x}_2$ after the $\mathtt{cx}_{12}$:

$$\mathtt{x}_2\mathtt{cx}_{12} = \mathtt{cx}_{12}\mathtt{x}_2, \qquad \mathtt{x}_1\mathtt{x}_2\mathtt{cx}_{12} = \mathtt{cx}_{12}\mathtt{x}_1.$$

The circuit diagrams corresponding to these relations are given in Figure 2.

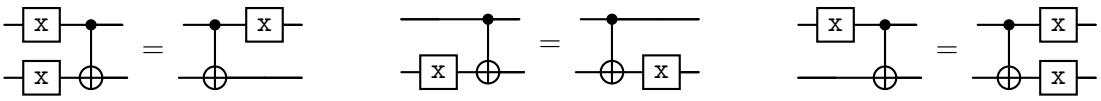


Figure 2: Relations between cx and x introduced in the new group.

Interpreting these relations as rewriting rules, it is possible to move as far as one wishes the $\mathtt{x}$ gates through $\mathtt{cx}$ gates, with possibly a different number of $\mathtt{x}$ gates at the end of the process. This eventually leads to the statement of Theorem 3.

A usual way for a group $H$ to act on another group $G$ is by *conjugation*, i.e., via a group homomorphism $\varphi_h$, defined for all element $h \in H$ by $\varphi_h(g) = hgh^{-1}$, for all $g \in G$. Expressing the relations expressed above and in Figure 2 as conjugations yields

$$\varphi_{jk}(\mathtt{x}_j) = \mathtt{x}_j\mathtt{x}_k, \qquad \varphi_{jk}(\mathtt{x}_k) = \mathtt{x}_k, \qquad \varphi_{jk}(\mathtt{x}_j\mathtt{x}_k) = \mathtt{x}_j,$$

with $\varphi_{jk} = \varphi_{\mathtt{cx}_{jk}}$, remembering that $\mathtt{cx}_{jk}^{-1} = \mathtt{cx}_{jk}$. Notice that $\varphi_{jk}$ is a group morphism, illustrated for instance by $\varphi_{jk}(\mathtt{x}_j\mathtt{x}_k) = \varphi_{jk}(\mathtt{x}_j)\varphi_{jk}(\mathtt{x}_k) = \mathtt{x}_1\mathtt{x}_2\mathtt{x}_2 = \mathtt{x}_1$. To be complete, we can add that $\varphi_{kl}$ acts like the identity on $\mathtt{x}$ gates acting on different qubits than $k$ and $l$. We now can give a presentation of the group $LX_q$ generated by local permutation gates.

**Proposition 3.** *A presentation of the group $LX_q$ is given by the generators*

$$\{\mathtt{x}_j \mid 0 \leq j \leq q-1\} \cup \{\mathtt{cx}_{kl} \mid k \neq l, 0 \leq k, l \leq q-1\}$$

*and the relations given by the group $X_q$, the group $CX_q$ and the relations in Figure 2:*

- *commutation:* $(\mathtt{x}_j\mathtt{cx}_{kl})^2 = \mathtt{I}$ *where* $k \notin \{j, l\}$*, and*
- *non-commutation:* $(\mathtt{x}_j\mathtt{cx}_{jk})^2 = \mathtt{x}_k$ *where* $j \neq k$*.*

*Proof.* We summarize the computation derived above. First, because $LX_q$ is generated both by $\mathtt{x}$ and $\mathtt{cx}$ gates, then it satisfies the relations of $X_q$ and $CX_q$. For the additional ones expressed in Figure 2, commutation relations arise whenever two gates are applied on different qubits, as expected. The last relations are derived exhaustively by trying all possible products of an individual $\mathtt{cx}$ gate and an individual $\mathtt{x}$ gate. There are four cases:

$$\mathtt{x}_j\mathtt{cx}_{jk}, \qquad \mathtt{x}_k\mathtt{cx}_{jk}, \qquad \mathtt{cx}_{jk}\mathtt{x}_j, \qquad \text{and} \qquad \mathtt{cx}_{jk}\mathtt{x}_k.$$

Studying the transformation induced by each of these circuits yields the identities:

$$\mathtt{x}_j\mathtt{cx}_{jk} = \mathtt{cx}_{jk}\mathtt{x}_j\mathtt{x}_k, \qquad \mathtt{x}_k\mathtt{cx}_{jk} = \mathtt{cx}_{jk}\mathtt{x}_k, \qquad \text{and} \qquad \mathtt{x}_j\mathtt{x}_k\mathtt{cx}_{jk} = \mathtt{cx}_{jk}\mathtt{x}_j.$$

The expression in the middle gives another commutation relation, which completes their enumeration. On the other ones, we multiply by $\mathtt{cx}_{jk}$ on the right on all these expressions:

$$\mathtt{x}_j = \mathtt{cx}_{jk}\mathtt{x}_j\mathtt{x}_k\mathtt{cx}_{jk} = \varphi_{jk}(\mathtt{x}_j\mathtt{x}_k), \qquad \text{and} \qquad \mathtt{x}_j\mathtt{x}_k = \mathtt{cx}_{jk}\mathtt{x}_j\mathtt{cx}_{jk} = \varphi_{jk}(\mathtt{x}_j).$$

Because $\varphi_{jk}(\mathtt{x}_k) = \mathtt{x}_k$ and $\varphi_{jk}$ is a group morphism, we can derive $\varphi_{jk}(\mathtt{x}_j\mathtt{x}_k)$ from $\varphi_{jk}(\mathtt{x}_j)$:

$$\varphi_{jk}(\mathtt{x}_j\mathtt{x}_k) = \varphi_{jk}(\mathtt{x}_j)\varphi_{jk}(\mathtt{x}_k) = \mathtt{x}_j\mathtt{x}_k\mathtt{x}_k = \mathtt{x}_j,$$

therefore only $\varphi_{jk}(\mathtt{x}_j) = \mathtt{cx}_{jk}\mathtt{x}_j\mathtt{cx}_{jk} = \mathtt{x}_j\mathtt{x}_k$ is necessary to complete the presentation. To put it under the standard form of a presentation, we rewrite it by multiplying on the left by $\mathtt{x}_j$ on both sides, giving the non-commutation relations. □

We now have a presentation of the group $LX_q$, which is the group of all circuits on $q$ qubits only composed of $\mathtt{x}$ and $\mathtt{cx}$ gates. Each element of $LX_q$ represents such a circuit, up to the rewriting rules expressed as the relations of the presentation.

For the time being, we still do not know how to efficiently expressed elements of $LX_q$ as quantum circuits. So, to complete the study of the structure of the group $LX_q$, we present a decomposition result for its elements $U$ into two elements $U = U_1U_2$, with $U_1$ written only using $\mathtt{x}$ gates, and $U_2$ only by $\mathtt{cx}$ gates. This decomposition result follows from the *semi-direct product* structure of $LX_q$.

**Definition 1** (Semi-direct product [28, p. 177])**.** Let $N$ and $H$ be groups and let $\varphi$ be a group morphism from $H$ into $\mathrm{Aut}(N)$[3]. Then the (outer) *semi-direct product* $N \rtimes_\varphi H$ of $N$ and $H$ with respect to $\varphi$ is defined as follows:

- The underlying set is the Cartesian product $N \times H$,
- The group operation is determined, for all $n_1, n_2$ in $N$ and $h_1, h_2$ in $H$, by

$$\begin{aligned}(N \rtimes_\varphi H) \times (N \rtimes_\varphi H) &\to N \rtimes_\varphi H \\ ((n_1, h_1), (n_2, h_2)) &\mapsto (n_1\varphi_{h_1}(n_2),\, h_1h_2).\end{aligned}$$

When $\varphi$ is clear from the context, we simply write $N \rtimes H$.

The semi-direct product of two groups can also be computed using the presentations of the two factor groups $N$ and $H$ and a choice of group morphism $\varphi : H \to \mathrm{Aut}(N)$.

[3] As usual $\mathrm{Aut}(N)$ denotes the group of all possible automorphisms of a given group $N$.

**Proposition 4** ([29, Corollary 1, p.140])**.** *Let $H = \langle\langle X|R\rangle\rangle$ and $N = \langle\langle Y|S\rangle\rangle$ be groups, and $\varphi : H \to \mathrm{Aut}(N)$ a group homomorphism. Then the semidirect product $N \rtimes_\varphi H$ has the following presentation:*

$$N \rtimes H = \langle\langle X \cup Y | R, S, xyx^{-1} = \varphi(x)(y), \forall x \in X, y \in Y \rangle\rangle.$$

We can now derive the relationship among the three groups $X_q$, $CX_q$ and $LX_q$.

**Proposition 5.** *The group of* `x` *and* `cx` *gates is isomorphic to the semi-direct product of the groups of* `x` *and* `cx` *gates taken alone, that is: $LX_q \simeq X_q \rtimes CX_q$.*

*Proof.* We use the same notation as before. We indeed have that $LX_q$ is generated by the union of the generators of $X_q$ and $CX_q$. Moreover, the elements of $LX_q$ satisfy both the relations of $X_q$ and $CX_q$. Now, we set

$$\begin{array}{rcccl} \varphi : CX_q & \to & \mathrm{Aut}(X_q) & & \\ \mathtt{cx}_{kl} & \mapsto & \varphi_{kl} : X_q & \to & X_q \\ & & w & \mapsto & \varphi_{kl}(w) = \mathtt{cx}_{kl} w \mathtt{cx}_{kl} \ . \end{array}$$

The automorphisms $\varphi_{kl}$ already act by conjugacy, and the relations of $LX_q$ contains all the relations that can be generated from it, which are the ones described in Proposition 3. Then, the presentation of $X_q \rtimes CX_q$ coincides with the one of $LX_q$, and because presentations characterize groups up to isomorphism, we have that $LX_q \simeq X_q \rtimes CX_q$. □

It turns out that the group $LX_q$ is isomorphic to a well-known group. Let $V$ be a vector space. We call the group $\mathrm{AGL}_n(V)$ defined by $\mathrm{AGL}_n(V) = V \rtimes \mathrm{GL}_n(V)$ the *affine group* of $V$.

**Corollary 1.** *The group of permutations generated by* `x` *and* `cx` *gates is isomorphic to the affine group of $\mathbb{F}_2^q$, i.e., $LX_q \simeq \mathrm{AGL}_q(\mathbb{F}_2)$.*

*Proof.* This follows from Proposition 2 and the definition of the affine group. □

*Remark* 3.1. Notice that the roles of $X_q$ and $CX_q$ here are *asymmetric*. Indeed, elements of $CX_q$ are mapped to automorphisms of $X_q$ in a natural way, and this is possible because, for all $\mathtt{cx}_{kl} \in CX_q$, we have that $\varphi_{kl}(w) \in X_q$ for all $w \in X_q$. We can extend this to the whole group, by noticing that $w_1 w_2 w_1^{-1} \in X_q$ for all $w_1, w_2 \in X_q$. Groups satisfying this properties are said to be *normal* in their ambient group, and we write $X_q \trianglelefteq LX_q$.

The next result provides the last tool necessary for the derivation of the decomposition.

**Proposition 6** ([28, Theorem 12 and Proposition 8])**.** *Suppose $G$ is a group with subgroups $N$ and $H$ such that $N \trianglelefteq G$, and $N \cap H = \mathrm{id}$. Let $\varphi : H \to \mathrm{Aut}(N)$ be the automorphism defined by mapping $h \in H$ to the automorphism $\varphi_h : n \in N \mapsto hnh^{-1} \in N$. Then $NH \simeq N \rtimes H$. Moreover, each element of $NH$ can be written uniquely as a product $nh$, for some $n \in N$ and $h \in H$.*

The result above summarizes what it means to be a semi-direct product for two groups. Simply, we can take products of elements of each groups, and form an element of a larger group. However, characterizing the group by identifying all the relations satisfied by the elements is not trivial whenever some of the elements do not commute. We are now able to state our decomposition result.

**Theorem 3.** *Every element $U \in LX_q$ can be uniquely written as a product $U_1U_2$, with $U_1 \in X_q$ and $U_2 \in CX_q$.*

*Proof.* By Proposition 5, we know that $LX_q \simeq X_q \rtimes CX_q$, so we aim to apply Proposition 6. By Remark 3.1, we have that $X_q$ is a normal subgroup of $LX_q$. The last hypothesis to check is that the intersection of $X_q$ and $CX_q$ is the identity. Consider any element $w_X \in X_q$ which is not the identity, and apply it to the ground state of a $q$-qubit register, $|0\rangle_q$. Necessarily, we have that $w_X |0\rangle_q \neq |0\rangle_q$. Now, let $w_{CX}$ be an element of $CX_q$ not being the identity. Necessarily, we have that $w_{CX} |0\rangle_q = |0\rangle_q$. Then, $w_X \neq w_{CX}$, and the only common element between $X_q$ and $CX_q$ is the identity. We can now apply Proposition 6, which concludes the proof. □

Thanks to Theorem 3, it suffices to know how to design a circuit $\mathscr{C}_X$ that spans the entirety of $X_q$ and another circuit $\mathscr{C}_{CX}$ that spans the entirety of $CX_q$ to be able to construct a quantum algorithm spanning every possible circuit formed by `x` and `cx` gates, up to the rewriting rules. The design of these circuits is the topic of the next section.

## 4 Quantum circuits

### 4.1 Circuits for $X_q$

In this section, we translate the group-theoretic results from the previous session into concrete tools to build circuits exploring groups. A straightforward example is the circuit $\mathscr{C}_X(\theta)$ exploring the group $X_q \simeq \mathbb{F}_2^q$. It simply amounts to a layer of $\texttt{rx}(\theta_j)$ gates, with the parameter $\theta_j$ playing the role of a switch: when it is evaluated at 0, the gate is the identity, and when it is evaluated at $\pi$, then the acts like `x`, up to an unimportant global phase factor. The strength of the variational quantum circuit $\mathscr{C}_X(\theta)$ is that it allows for more value of $\theta_j$ than just 0 and $\pi$. Actually, any real number is a valid parameter. Hence, the circuit explores the group $X_q$ while travelling through some extra space giving by this relaxation, therefore being in a superposition of permutations.

### 4.2 Circuits for $CX_q$

The picture gets blurrier when we want to explore the group $CX_q$. We need to leverage the results of the previous section to divide this problem into easier subproblems, i.e., smaller groups for which we know how to design variational quantum circuits exploring them.

We recall that it has been shown by Bataille [24] that $CX_q$ is isomorphic to $\mathrm{GL}_q(\mathbb{F}_2)$. In order to span the latter group in a compact way with a parameterized circuit, we use the Bruhat decomposition of $\mathrm{GL}_q(\mathbb{F}_2)$ (see Theorem 2). Consider the following.

Given a group $G$ with subgroups $B$ and $N$, the data defining a $BN$-pair [33] consists formally of a quadruple $(G, B, N, S)$ subject to the following requirements. First, $G$ is generated by its subgroups $B$ and $N$. Second, $B \cap N$ is a normal subgroup of $N$. Third, the group $W \coloneqq N/(B \cap N)$ is generated by a set $S$ of involutions and, finally, for all $s \in S$ and $w \in W$, we have $sBw \subseteq BwB \cup BswB$, as well as $sBs \neq B$.

In the case $G = \mathrm{GL}_q(\mathbb{F}_2)$, the subgroup $B$ is the *Borel subgroup* formed by the upper triangular matrices, and $N$ is the subgroup having a unique non-zero element per row and per column [32]. Over $\mathbb{F}_2$, $N$ is only composed of permutation matrices. Then, the subgroup $B \cap N = \{\texttt{I}\}$ is trivial, because the only permutation matrix being upper

triangular is the identity. Therefore, the *Weyl group* $W = N/(B \cap N)$ of the $BN$-pair is simply the symmetric group $S_q$.

Recall now that the Bruhat decomposition theorem states that any group $G$ satisfying the conditions described above can be written as $G = BWB$. More specifically, each element of $\mathrm{GL}_q(\mathbb{F}_2)$, and therefore of $CX_q$, can be uniquely written as the product of a permutation matrix, multiplied on the left and the right by upper triangular matrices. Fortunately, both $W$ and $B$ can be easily embedded into quantum circuits.

### 4.3 The Borel subgroup $B$

The Borel subgroup $B$ is the subgroup of $\mathrm{GL}_q(\mathbb{F}_2)$ composed of upper triangular matrices. Because they are invertible, and defined over $\mathbb{F}_2$, their diagonal only contains the entry 1. Hence, their coefficients can only differ strictly above the diagonal. The matrices are of size $q \times q$, so there are $(q-1) + (q-2) + \ldots + 1 = \binom{q}{2}$ entries that vary. Because each of them can only take two values, there are $2^{\binom{q}{2}}$ elements in the Borel subgroup $B$. Hence, the minimal number of gates we need to span $B$ is $\binom{q}{2}$.

Because we are on the field $\mathbb{F}_2$, the group $\mathrm{GL}_q(\mathbb{F}_2)$ is isomorphic to $\mathrm{SL}_q(\mathbb{F}_2)$, the group of invertible matrices with determinant 1. Therefore, we can use the result presented in [34, Theorem 4.6] to say that a set of generators is given by the transvections

$$\{T_{(jk)} \mid 1 \leq j, k \leq q\}, \qquad \text{with} \qquad T_{(jk)} = I + E_{(jk)},$$

where $E_{(jk)}$ is the matrix with a 1 at entry $(j, k)$ and 0 everywhere else. Bataille [24] has shown that the transvections are in bijection with the `cx` gates. More precisely, the transvection $T_{(jk)}$ corresponds to the `cx` gate controlled by qubit $k$ acting on qubit $j$. This construction is used to build the isomorphism between $\mathrm{GL}_q(\mathbb{F}_2)$ and $CX_q$.

As we will demonstrate shortly, the Borel subgroup $B$ consisting of upper-triangular invertible matrices is generated by upper-triangular transvections, i.e., by the set

$$\left\{T_{(jk)} \mid 1 \leq j < k \leq q\right\}.$$

Then, the group $B$ is generated by the `cx` gates where the index of the control qubit is smaller than the one of the target qubit. However, to build a circuit able to span all elements of $B$, we need to find a *universal element* of the group, an element $g_\star \in B$ which can be written as a word $w_\star$ of transvections with the following property: for any element $g \in B$ written as a word $w$ of transvections, $w$ is a subword of $w_\star$. This is a direct generalization of the concept of longest element of the Bruhat order in Coxeter groups.

**Definition 2.** We say that a word $w = s_{j_1} \ldots s_{j_p}$ is a subword of another word $v = s_{k_1} \ldots s_{k_m}$ if there exists an order-preserving injection from the sequence $(j_1, \ldots, j_p)$ into the sequence $(k_1, \ldots, k_m)$. In other words, $w$ can be written by taking arbitrary letters from $v$ as long as the order between them does not change. For example, *ade* is a subword of *abcde*, while *acb* or *abf* are not.

Let $(Q, \prec)$ be the ordered set with $Q = \{(j, k) \mid 1 \leq j < k \leq q\}$ being composed of all the ordered pairs of increasing indices equipped with the lexicographic order $\prec$ given by

$$(j_1, k_1) \prec (j_2, k_2) \qquad \text{if and only if} \qquad [j_1 < j_2] \text{ or } [j_1 = j_2 \text{ and } k_1 < k_2].$$

The order $\prec$ is a total order on $Q$. We denote by $w_\star$ the word of transvections defined as the product of all transvections multiplied on the left in the order of $\prec$, i.e.,

$$w_\star \coloneqq T_{(q-1\,q)} T_{(q-2\,q-1)} T_{(q-2\,q)} \ldots T_{(jq)} \ldots T_{(j\,j-1)} T_{(j-1\,q)} \ldots T_{(23)} T_{(1q)} \ldots T_{(13)} T_{(12)}.$$

**Theorem 4.** *Any element of $B$ can be written as a subword of $w_\star$.*

*Proof.* Let $A$ be an upper triangular matrix of $B$. We show that we can obtain $A$ from the identity matrix by multiplying it on the left by a subword of $w_\star$. We construct it entry by entry, in the order given by $\prec$. Let $(Q_A, \prec)$ be the ordered set of indices defined by $Q_A = \{(j,k) \in Q \mid A_{jk} = 1\}$. We denote by $w_A$ the subword of $w_\star$ obtained by only keeping the symbols $\{T_{(jk)} \mid (j,k) \in Q_A\}$. Let $M \in B$ be a matrix and $(p,r)$ be a pair of indices such that, for all $(p,r) \prec (j,k) \in Q$ we have $M_{jk} = 0$, as well as $M_{pr} = 0$. We have

$$T_{(pr)}M = \left(I + E_{(pr)}\right) M = M + E_{(pr)}M.$$

Because $E_{(pr)}$ has a unique nonzero coefficient at entry $(p,r)$, the product $E_{(pr)}M$ only consists of the $r$th row of $M$ sitting on the $p$th row. But because $M_{jk} = 0$ for all $(j,k) \succ (p,r)$, the $r$th row of $M$ is simply $e_r^\top$, the transpose of the $r$th element of the canonical basis of $\mathbb{F}_2^q$. Then, $T_{(pr)}M$ is equal to $M$ to which we add 1 at the entry $(p,r)$. Because we only apply transvections following the lexicographic order $\prec$, the product $w_A$ adds entries 1 one by one to the identity when the corresponding entry in $A$ is itself 1, so it indeed corresponds to the matrix $A$. □

**Example 2.** We illustrate this result with a specific example. Let $A$ be an upper-triangular matrix defined by

$$A = \begin{pmatrix} 1&0&1&0 \\ 0&1&1&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}.$$

Then, $A$ can be written as $A = T_{24}T_{23}T_{13}$. Indeed, we have

$$T_{24}\begin{pmatrix} 1&0&0&0 \\ 0&1&1&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}\begin{pmatrix} 1&0&1&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}\begin{pmatrix} 1&0&1&0 \\ 0&1&1&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&1&0 \\ 0&1&1&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} = A.$$

◊

The next step is to apply Theorem 4 to actually build a quantum circuit $\mathscr{B}(\theta)$. Using the correspondence between `cx` gates and transvections, we are able to design a variational circuit exploring the Borel subgroup $B$ of $CX_q$. In these circuits, we use a custom gate, displayed as a parameterized box with a control wire. This gate acts as the identity whenever the parameter evaluates to 0, and as a `cx` gate when the parameter is $\pi$, although it is defined for all real-valued parameters. The construction is discussed in Appendix B.

**Example 3.** Let us illustrate this first construction on 6 qubits. In this case, the Borel group $B_6$ is the set of upper-triangular matrices of size $6 \times 6$ equipped with the usual matrix product. Applying Theorem 4, we have that any upper-triangular matrix can be written as a subword of

$$w_\star^{(6)} = \; T_{(56)} \;\; T_{(46)}T_{(45)} \;\; T_{(36)}T_{(35)}T_{(34)} \;\; T_{(26)}T_{(25)}T_{(24)}T_{(23)} \;\; T_{(16)}T_{(15)}T_{(14)}T_{(13)}T_{(12)}.$$

Consider now the implementation of an upper-triangular matrix $M \in B_6$ as a quantum circuit. We first look at the entries of our matrix from the bottom left to the bottom right, line per line. We see that we have a transvection matrix for each of the entries. If the entry $M_{jk}$ is 0, we remove the corresponding transvection $T_{jk}$ from $w_\star^{(6)}$, but if it is 1, we keep it. The word that we then obtain is equal to the matrix $M$. To translate it into a quantum circuit, we use the isomorphism of Bataille [24]. We read the word representing $M$ from right to left, and for each transvection $T_{(jk)}$, we add a $\texttt{cx}_{kj}$ gate to our circuit.

Hence, any upper-triangular matrix is implement as a subcircuit of the circuit spanning the Borel subgroup $B_6$ depicted in Figure 3.

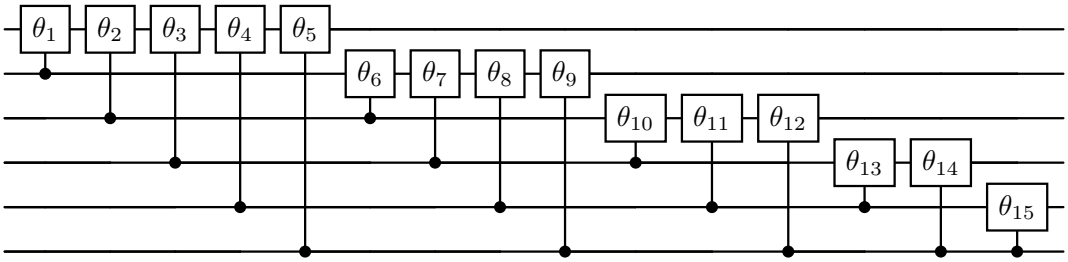


Figure 3: Variational circuit $\mathscr{B}(\theta)$ spanning the Borel subgroup $B_6$.

The parameters $\theta$ in the circuits control the application of the corresponding gate in the circuit: putting $\theta = 0$ acts as if we remove the gate, while $\theta = \pi$ implements it fully. The cases when $\theta$ is neither 0 nor $\pi$ is treated in the next section. Notice how the order is flipped: this is due to the fact that functions from formulas are applied from the right to the left, but the opposite in circuit diagrams. ◊

The number of parameters and gates required in this circuit is in $O(q^2)$. However, some of these gates can be applied in parallel. For instance, the gate depending on $\theta_6$ can be applied as early as the one depending on $\theta_3$, and the one depending on $\theta_{10}$ as early as the one depending on $\theta_8$, which is itself applied with the gate depending on $\theta_5$.

We can see the circuit as composed of several blocks, the first one being composed of all the `cx` gates controlled by the first qubit, the second block being composed of the `cx` gates controlled by the second qubit, and so on. We therefore have $q-1$ blocks. The gates from the second block can all be executed in parallel of the one of the first block, except for the last one, the one depending on $\theta_9$. The same phenomenon happens with each block, where all but the last gate can be applied with the ones of the previous blocks.

Hence, even if the size of the circuit is quadratic in the number of qubits, its depth is simply linear, as it is given by $(q-1)+(q-2) \sim 2q \in O(q)$. Indeed, the first block is of depth $q-1$, and each following bloc, $q-2$ of them, only adds a depth of 1.

## 4.4 The Weyl subgroup $W$

As we saw earlier, the Weyl subgroup $W$ of the $BN$-pair is isomorphic to the symmetric group $S_q$, which can be seen as the group of permutations of the $q$ qubits (see also Example 1). A presentation of the group $W$ is therefore given by

$$S_q = \langle\!\langle \sigma_1, \ldots, \sigma_{q-1} \,|\, \sigma_k^2 = 1,\ (\sigma_k \sigma_{k+1})^3 = 1,\ (\sigma_k \sigma_j)^2 = 1 \text{ if } |k-j| > 1 \rangle\!\rangle,$$

with the $\sigma_1, \ldots, \sigma_{q-1}$ being the adjacent transpositions defined by $\sigma_k = (k \ \ k+1)$, written using the cycle notation. Like the upper triangular matrices being written as words of transvections, permutations can be written as words of adjacent transpositions.

To implement these adjacent transpositions of qubits, we use the well-known decomposition of the `swap` gates using three `cx`, the first and the third one pointing in a different direction than the second one (see Figure 4). The first and third gates are not parameterized, but the second one is. By doing so, the whole `swap` gate collapses if the middle `cx`

gate is set to be the identity, with $\phi = 0$. However, when $\psi = \pi$, the parameterized `cx` gate becomes a standard `cx` gate, and the three gates together are indeed a `swap` gate.

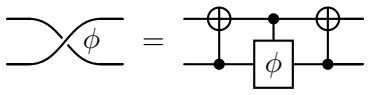


Figure 4: Construction of a parameterized `swap` gate.

We now gather all these adjacent transpositions in a harmonious way in order to generate any required permutation of qubits, which compose the Weyl subgroup $W$.

The partial order on word representations of permutations obtained by subword relations corresponds to the Bruhat order on Coxeter groups, also known as the weak order of permutations in the case of the symmetric group. Each Bruhat order has a unique maximal element. Then, to span the whole group $S_q$, we implement a word representation of this longest element of the Bruhat order as a quantum circuit.

In particular, the following characterization of the Bruhat order on the symmetric group allows us to identify this longest element. For each $\eta \in S_q$ and each $j, k \in [q]$, define

$$\eta[j,k] \coloneqq |\{l \in [j] \mid \eta(l) \geq k\}|.$$

This function counts, for each index $j \in [q]$, the number of smaller indices sent after index $k$ through the permutation $\eta$. We then use the following result.

**Theorem 5** (Björner and Brenti [35])**.** *Let $\eta, \rho \in S_q$ be two permutations. Then, $\eta$ is smaller than $\rho$ in the Bruhat order if and only if $\eta[j,k] \leq \rho[j,k]$ for all $j, k \in [q]$.*

Naturally, the permutation $\eta_\circ \in S_q$ reversing the order of the elements of its input is the unique permutation reaching the maximal value $\eta_\circ[j,k]$ on each pair $j, k \in [q]$. Then, by Theorem 5, it is the maximal element of the Bruhat order. A maximal element for the Bruhat order is necessarily unique, and in the case of $S_q$, its length, as a word of adjacent transpositions, is $\binom{q}{2}$ [35]. A well-known word representation for the permutation $\eta_\circ$ is

$$w_\circ \coloneqq \ \sigma_1 \ \sigma_2\sigma_1 \ \sigma_3\sigma_2\sigma_1 \ \ \ldots \ \ \sigma_{q-1}\sigma_{q-2}\ldots\sigma_2\sigma_1.$$

It is obtain by first multiplying all the adjacent transpositions in the order of the indices, then multiplying on the left the product of all the adjacent transpositions in the order of the indices except the last one $\sigma_{q-1}$, and we repeat this recursively, dropping the last transposition at each step. The $q-1$ first transpositions applied, i.e., the ones on the right of $w_\circ$, serve to put the first element of the input at the last spot. Then, the $q-2$ next ones are used to put the second element of the input at the second-to-last spot, and so on. Then, $w_\circ$ is a word representing the reversing permutation $\eta_\circ$. Moreover, the length of $w_\circ$ is given by $\binom{q}{2}$, and is indeed one of the reduced word representation of $\eta_\circ$.

**Example 4.** Let us continue Example 3, with 6 qubits. In this case, the elements of the Weyl group $W_6$ are the permutations of 6 symbols, here the 6 qubits. Then, the maximal element of the Bruhat order of $W_6$ is given by

$$w_\circ^{(6)} = \ \sigma_1 \ \sigma_2\sigma_1 \ \sigma_3\sigma_2\sigma_1 \ \sigma_4\sigma_3\sigma_2\sigma_1 \ \sigma_5\sigma_4\sigma_3\sigma_2\sigma_1.$$

The quantum circuit to span the Weyl subgroup $W_6$ is depicted in Figure 5.

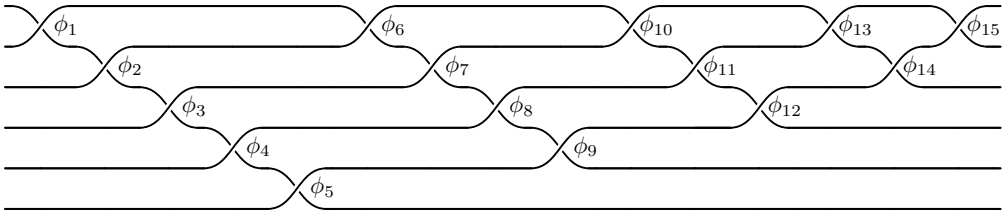


Figure 5: Variational circuit $\mathscr{W}(\phi)$ spanning the Weyl subgroup $W$.

Again, the parameters $\phi$ enable the user to control if the gate is active or not, allowing the access to any subword. ◊

The number of parameters of this circuit is $\binom{q}{2} \in O(q^2)$, and the number of controlled gates is $3\binom{q}{2} \in O(q^2)$. However, as before, the depth is significantly shorter. Indeed, a lot of gates can be applied in parallel, in the same way than for the previous circuit. As before, there are $q-1$ blocks, the first one increases the depth of the circuit by $3(q-1)$, and the following ones increase the depth by 3. Hence, the depth is $3(q-1)+3(q-2) \in O(q)$.

## 4.5 Exploring the whole group generated by `cx` gates and $LX_q$

To span the whole group $CX_q$, thanks to the Bruhat decomposition, we simply need to concatenate the different circuits spanning $B$ and $W$. Therefore, the total number of controlled gates used to span $CX_q$ is $\binom{q}{2} + 3\binom{q}{2} + \binom{q}{2} = 5\binom{q}{2} \in O(q^2)$, while the number of parameters used is $3\binom{q}{2} \in O(q^2)$. The total depth is less than the sum of the depths, because part of the gates of the circuit exploring $W$ can be applied at the same time as some of the first occurrence of the circuit spanning $B$.

**Example 5.** Let us consider 4 qubits. According to Theorem 2, the Bruhat decomposition of $CX_4$ is given by $B_4W_4B_4$. The circuit spanning $CX_4$ is depicted in Figure 6.

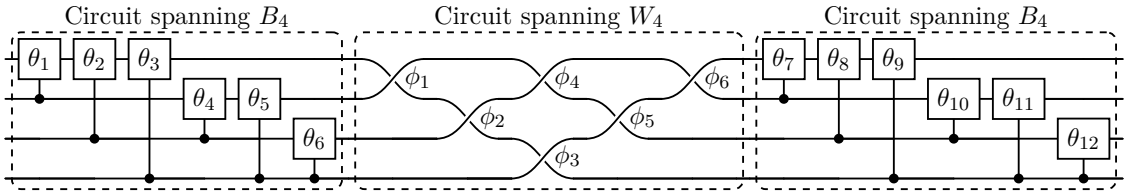


Figure 6: Variational circuit spanning $CX_4$.

Notice here that the parametrization of the two Borel blocks are not parameterized by the same values: the two blocks can encode different upper-triangular matrices. ◊

The final depth of the circuit is given by

$$q + 3(q-1) + 3(q-2) + (q-1) + (q-2) = 9q - 12 \in O(q),$$

hence is linear in the number of qubits.

Finally, to explore the whole group $LX_q$, it suffices to add a layer of `rx` gate upstream of the circuit, which increases the size of the circuit by $q$, and its depth by 1.

## 4.6 Topology constraints on near-term devices

In the circuits presented above, we assume an all-to-all qubit connectivity. However, on some hardware, and especially for near-term processors, this assumption can be too demanding. For the circuit spanning the Weyl group $W_q$, there should be no issue during the physical implementation in terms of connectivity: we are only using 2-qubits gates acting on neighboring qubits. For the circuit spanning the Borel group $B_q$, this is no longer true, because the ($q$-1)-th gate has the longest possible range. To solve this issue, we use the method developed by Bäumer and Wörner [36] to implement long-range `cx` gates. If we have access to ancilla qubits, we can direct use their implementation, which works in constant depth, hence it does not change the overall scaling of the depth of our circuits. For an ancilla-free implementation, we resort to the rewriting rules in Figure 2 and Figure 3 of [36], as follows.

A long-range `cx` gate can be decomposed as two multi-target `cx` gates, which can themselves be decomposed as many `cx` acting on neighboring qubits as follows:

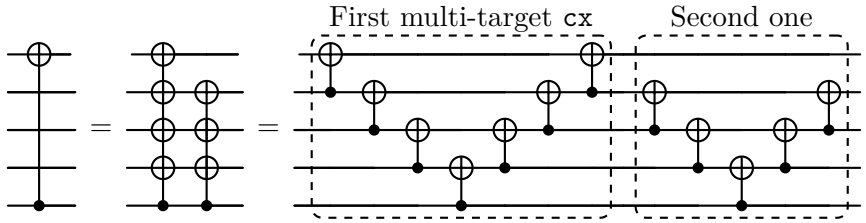


Figure 7: Decomposition of a long-range `cx` gate using `cx` acting on neighboring qubits.

Observe that, if we remove the two topmost `cx` gates, all the remaining ones cancel, and we get the identity. So, we apply the parametrization on these `cx` gates. It might not be exactly the same unitary gate as the long-range `cx` for arbitrary angles, but it is not important, as long as the evaluation at $\theta = 0$ or $\theta = \pi$ are the same, meaning that we span the same permutations.

In terms of complexity, we get the following. When the index of target qubit is $p$ less than the index of the control qubit, we need $2(1 + 2(p-1)) - 2 = 4(p-1)$ neighboring `cx` gates if $p > 1$, and 1 if $p = 1$. So, in a circuit spanning the Borel group $B_q$ on $q > 3$ qubits, the number of long-range `cx` gates targeting qubit $j < q - 2$ is

$$1 + \sum_{p=2}^{q-j} 4(p-1) = 1 + 4 \sum_{p'=1}^{q-j-1} p' = 1 + 4\binom{q-j}{2} \in O(q^2).$$

There are a linear number of these long-range `cx`, so the size and the depth of the circuit spanning the Borel group scale in $O(q^3)$, giving an overall circuit of cubic size and depth.

## 4.7 Counting the number of spanned permutations

In this section, we have translated the group-theoretical results about the structures of the group $LX_q$ of permutations generated by local permutations on $q$ into tools for circuit synthesis. The meta-circuit exploring $LX_q$ is of the form $XBWB$, where $X$ is the group of `x` gates, $B$ is the Borel subgroup of upper-triangular invertible matrices of $\mathrm{GL}_q(\mathbb{F}_2)$, and $W$ is the Weyl subgroup of $\mathrm{GL}_q(\mathbb{F}_2)$, isomorphic to the group of permutations of $q$ qubits.

These circuits, using a quadratic number of parameters, are of linear depth with respect to the number of qubits. Thanks to the extensive characterization of the structure of $LX_q$, we are able to compute the number of permutations spanned by the circuit.

**Theorem 6.** *The maximum number of unique permutations $p$ obtainable by circuits containing* `x` *and* `cx` *gates is*

$$p = 2^{\frac{q(q+1)}{2}} \prod_{k=1}^{q} \left(2^k - 1\right) \in O\left(2^{O(q^2)}\right).$$

*Proof.* Given Theorem 3, we can count permutations by multiplying the ones coming from $X_q$ and the ones from $CX_q$. For the latter, we use the result [24, Corollary 7] about the number of elements of $CX_q \simeq \mathrm{GL}_q(\mathbb{F}_2)$, which we then need to multiply by the number of elements of $X_q$ ($2^q$) to compute the number of elements of $LX_q$. □

Evaluations of this formula for small numbers of qubits are given in Table 1, along with the number of classical parameters $\ell = q + 3\binom{q}{2}$, and the depth $d = 1 + 9q - 12$.

Table 1: Number $p$ of spanned permutations with $q$ qubits, using $\ell$ classical parameters, in $d$ depth. In comparison the size of the permutation set of $\Pi_n$ with $n = 2^q$, which we encode, is $n!$.

| $q$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| $\ell$ | 5 | 12 | 22 | 35 | 51 | 70 |
| $d$ | 7 | 16 | 25 | 34 | 43 | 52 |
| $p$ | 24 | 1,344 | 322,560 | $\sim 3.2 \times 10^8$ | $\sim 1.3 \times 10^{12}$ | $\sim 2.1 \times 10^{16}$ |
| $\lvert\Pi_n\rvert$ | 24 | 40,320 | $\sim 2.1 \times 10^{13}$ | $\sim 2.6 \times 10^{35}$ | $\sim 1.2 \times 10^{89}$ | $\sim 3.8 \times 10^{215}$ |

The number of spanned permutations, i.e., all permutations obtained from one- and two-qubit gates, is rather small compared to the total number of permutations (with the exception of $q = 2$ for which the two numbers match). We see in the next section how to make use of ancilla qubits to both increase the span and to avoid classical simulability. Furthermore, we will see in the numerical simulation section, how, despite the small span, we obtain reasonably good approximate solutions for permutation-based optimization problems.

## 5 Ansatze for optimization problem-solving

### 5.1 Obtaining a doubly-stochastic matrix from a quantum circuit

As mentioned in the introduction, some challenges of QAOA are related to the fact that constraints are not easily implementable and that it is complicated to embed any cost function as observable, especially in an amplitude encoding setting. We usually either have to prepare a specific state, or to derive a cost Hamiltonian and synthesize a circuit representing it. In this paper, we choose to do none of the technics mentioned above. Instead, we leverage a clever quantum circuit exploited recently by Mariella et al. [21].

Consider the variational circuit presented in Figure 8, where $\mathscr{U}(\theta) \in \mathbb{C}^{2^m+n}$ is any arbitrary unitary matrix. The circuit is initialized in the $|0\rangle$ state. We have a total of $2\log_2(n) + 2m$ qubits. The first represents the $n^2$ entries of a square matrix, while the additional $2m$ qubits are ancilla. We measure only the last $2\log_2(n)$ qubits.

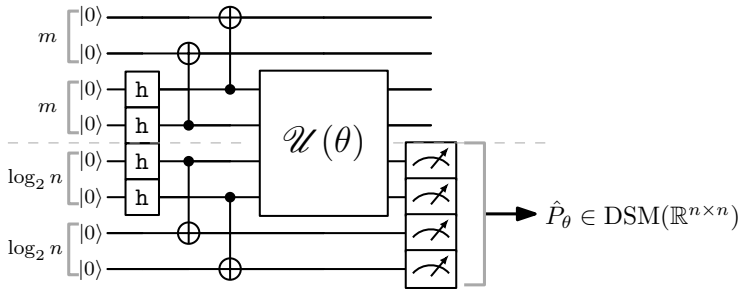


Figure 8: Depiction of the variational circuit of Mariella et al. [21] to obtain a doubly-stochastic matrix (DSM) starting from any unitary. For us this circuit will be denote by $\mathscr{P}(\theta)$.

Mariella and coauthors show in their Lemma 3.1 that, *for any* $\mathscr{U}(\theta)$, measuring the last $2\log_2(n)$ qubits one obtains the corresponding entries of a $n \times n$ doubly-stochastic matrix. We recall that a doubly-stochastic matrix is a convex combination of permutation matrices. Furthermore, they also show that, if $m = 0$, then the measurements deliver the entries of the matrix $\mathscr{U}(\theta) \odot \overline{\mathscr{U}(\theta)}$, where $\odot$ is the Hadamard entry-wise product and $\overline{(\cdot)}$ is the complex conjugate operator. In particular, if $m = 0$ and $\mathscr{U}(\theta)$ is a permutation matrix, then the output of the circuit is the same permutation matrix. We summarize the results obtained in [21] in the following proposition, for later use.

**Proposition 7.** *[21] Let* $|i\rangle$ *refer to the* $i$*-th basis vector in the computational basis and let* $|ij\rangle = |i\rangle \otimes |j\rangle$*. Define the row-major vectorization operator on a matrix* $M$ *in* $\mathbb{C}^{d\times d}$ *as*

$$\mathrm{vec_r}(M) := \sum_{i=0}^{d-1} M\,|i\rangle \otimes |i\rangle\,.$$

*Consider a bipartite Hilbert space* $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$*, with dimensions* $2^m$ *and* $n$*, respectively. Recall that any (parametrized) unitary matrix* $\mathscr{U}(\theta) \in \mathbb{C}^{2^m+n}$ *can be decomposed into*

$$\mathscr{U}(\theta) = \sum_{i=1}^{d^2} \lambda_i V_i(\theta) \otimes W_i(\theta),$$

*by the operator Schmidt decomposition, with* $\{V_i\}$ *and* $\{W_i\}$ *being sets of unitary operators orthogonal w.r.t. the Frobenius inner product, belonging to* $\mathcal{H}_1$ *and* $\mathcal{H}_2$*, respectively. Furthermore* $\lambda_i \geq 0$*,* $\sum_i \lambda_i^2 = 1$*, and* $d^2 = \min\{(2^m)^2, n^2\}$*.*

*Consider now the circuit in Figure 8. Partition the circuit into two subsystems. The first corresponding to the first auxiliary* $2m$ *qubits and the second corresponding to the last* $2\log_2(n)$ *qubits. The mixed state corresponding the the last* $2\log_2(n)$ *qubits is,*

$$\rho(\theta) = \frac{1}{n}\sum_{i=1}^{d^2} \lambda_i^2\, \mathrm{vec_r}\left(W_i(\theta)\right) \mathrm{vec_r}\left(W_i(\theta)\right)^\dagger. \tag{2}$$

*Furthermore, measuring the last* $2\log_2(n)$ *qubits one obtains the following. Let,*

$$p_{ij}(\theta) := n\mathrm{Tr}(\rho(\theta)\,|ij\rangle\,\langle ij|) = \sum_{k=1}^{d^2} \lambda_k^2\,\langle i|\left(W_k(\theta) \odot \overline{W_k}(\theta)\right)|j\rangle \tag{3}$$

*for* $i, j \in [0\,..\,n-1]$*. Let* $\{\mathbf{e}_i\}$ *be the set of canonical basis vectors (with index* $i$ *starting from 0) for the vector space* $\mathbb{R}^n$*. Then the matrix*

$$\hat{P}_\theta = \sum_{i,j=0}^{n-1} p_{ij}(\theta)\,\mathbf{e}_i\mathbf{e}_j^\top = \sum_{i,j=0}^{n-1} p_{ij}(\theta)\,|i\rangle\,\langle j|\,, \tag{4}$$

*is doubly stochastic.*

Proposition 7 tells us that given the density matrix $\rho(\theta)$ prepared as in Eq. (2), the expectations w.r.t. the observables $|ij\rangle\langle ij|$ provide the corresponding $(i,j)$ entry of $\hat{P}_\theta$, and this $\hat{P}_\theta$ is doubly stochastic (in fact, unistochastic).

For $m=0$, and $\mathscr{U}(\theta) \in \Pi_n$, then $\hat{P}_\theta = \mathscr{U}(\theta) \odot \overline{\mathscr{U}(\theta)} = \mathscr{U}(\theta)$. Which is true, since the entries of a permutation matrix are only 0 or 1.

For $m > 0$, one can see the role of the auxiliary $m$ qubits, that is enlarging the function space as a result of the convex combination of density matrices in Eq. (2). If $m=0$, then the number of terms in Eq. (2) reduces to 1, while for $m>0$, the number of is much larger (in fact, one can prove that all but two of the $\lambda_i$ must be positive [37]).

Having discussed how to generate doubly-stochastic matrices, we are now ready to see how to enlarge the number of explored permutations in our ansatze.

## 5.2 Increasing the number of explored permutations

As we have observed in Theorem 6, the number of spanned permutation scales as $2^{O(q^2)}$ if $q = 2^n$ is the number of qubits used to represent a permutation matrix of dimension $n$. This is the best we can do with local gates, but not quite close to the total number of permutations $n!$, as we can see on Table 1.

The idea is now to enlarge the spanned space by adding extra qubits as in the circuit in Figure 8, with the unitaries that we have constructed in Section 4. With this, we can obtain the following advantages.

First, by Eq. (2), we can see that the number of convex combination augments, giving us the hope that adding addition qubits is similar to spanning a much larger space and then projecting it onto our problem space $n$. This hope is supported by theory (below) and simulation studies.

Second, since to compute the optimization cost $f(P)$ will typically involve $O(n^2)$ arithmetic operations as well as storage, a unitary in $\mathbb{C}^{n\times n}$ is classically simulatable by the same computer. By increasing the number of qubits and choosing $m \gg n$, then our unitaries will cease to be classically simulatable (even though the cost will remain so) and the algorithm derived upon them will be genuinely quantum.

To support the first advantage, we can now prove a useful theoretical result, which characterizes the output $\hat{P}_\theta$, whenever $\mathscr{U}(\theta)$ is a permutation matrix. Specifically, we choose the Bruhat unitary, $\mathscr{U}^{\text{Bruhat}}(\theta)$, and the Borel unitary, $\mathscr{U}^{\text{Borel}}(\theta)$, both obtained in Section 4, and reported in Figure 9, for simplicity.

**Proposition 8.** *Consider the circuit in Figure 8, with unitaries $\mathscr{U}(\theta)$ taken as $\mathscr{U}^{\text{Borel}}(\theta)$, or $\mathscr{U}^{\text{Bruhat}}(\theta)$ . Let $\theta \in \{0,\pi\}^\ell$ for all the $\ell$ parameters, so that the unitary represent a permutation matrix and recall $n = 2^q$. Then,*

$$\hat{P}_\theta = \sum_{i=1}^{w} \lambda_i P_i, \quad P_i \in \text{span}(\text{Bruhat}_n),$$

*with $w > 1$ whenever $m > 0$, i.e., $\hat{P}_\theta$ is a convex combination of permutation matrices belonging to the set of permutations in the Bruhat span of dimension $n$. In addition,*

$$\begin{aligned} \mathscr{U}^{\text{Bruhat}}(\theta) &: w \leq \max\{2^{3mq}, |\text{span}(\text{Bruhat}_n)|\}; \\ \mathscr{U}^{\text{Borel}}(\theta) &: w \leq \max\{2^{mq}, |\text{span}(\text{Bruhat}_n)|\}. \end{aligned}$$

*Finally, the number of distinct $\hat{P}_\theta$ is at most equal to the cardinality of the Bruhat span of dimension $n2^m = 2^{q+m}$.*

*Proof.* The proof is based on the operator Schmidt decomposition of `cx` and `swap` gates, which are the gates that connects the $m$ part to the $n$ part of the circuit. Whenever there is a connecting gate, either the parameter is 0, and the gate is the identity, or the parameter is $\pi$ and we have a `cx` or a `swap` depending on the parametrized connecting gate. Up to a renormalization, we have

$$\texttt{cx} \simeq |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X, \qquad \text{and} \qquad \texttt{swap} \simeq I \otimes I + X \otimes X + Y \otimes Y + Z \otimes Z,$$

see for instance [38]. Since the decomposition involves only matrices which have one non-zero elements per row and column, whose modulus is 1, then, $W_i(\theta) \odot \overline{W_i}(\theta)$ in Eq. (3) is a permutation matrix obtained combining local gates. Specifically, the $Y$'s will act as $X$ and the $Z$'s act as $I$ in the SWAP gate. This implies that the number of distinct $W_i(\theta) \odot \overline{W_i}(\theta)$ matrices is upper bounded by the span of the cardinality of the Bruhat span of dimension $n$. The number of terms in the convex combination is also upper bounded by the number of splits caused by connecting gates. The number of connecting gates are $3mq$ for Bruhat and $mq$ for Borel, giving an upper bound of $2^{3mq}$ and $2^{mq}$, respectively.

Finally, any locally constructed permutation $\mathscr{U}(\theta)$ in $2^{q+m}$ dimension will be among the ones in the Bruhat permutation set of dimension $n\ 2^m$. Each of them will give rise to a different operator Schmidt decomposition and henceforth we can count the maximum number of distinct $\hat{P}_\theta$. □

A few remarks are in order.

*Remark* 5.1. Whenever an ancilla is used, $\hat{P}_\theta$ will be a doubly-stochastic matrix, even if $\mathscr{U}(\theta)$ was a permutation matrix in $2^m + n$ (i.e., all the $\theta$'s were either 0 or $\pi$). This is not a problem, since we know how to project onto the permutation matrices classically (see below and [22, 23]).

*Remark* 5.2. Due to the number of unique $\hat{P}_\theta$, whenever $\mathscr{U}$ is a permutation, one can compute a lower bound on the number of required ancilla qubits (imagining that each doubly-stochastic matrix maps into a unique permutation) to span the set of permutations acting $n$ symbols, which gives us

$$2^{(m+\log_2(n))^2} \approx n! \implies m \gtrsim \Omega(\sqrt{n \log n}).$$

This lower bound is quadratically lower than the number of qubits required in QAOA [39]. In addition, here we have the choice of selecting how much of the set we span by augmenting or reducing the number of used ancillas. This tuning knob gives us the power to trade-off span and implementability on current hardware, which has no parallel in QAOA.

*Remark* 5.3. Due to the presence of the operator Schmidt decomposition characterizing the exact span of $\hat{P}_\theta$ is challenging to do, but the interested reader can see some interesting works in this respect [40, 41, 37].

### 5.3 Considered $\mathscr{U}(\theta)$ and empirical quantum boost

We can now explore by numerical simulations the total number of unique permutations generated by our quantum circuit (Figure 8) with a few different unitaries $\mathscr{U}(\theta)$. For comparison, we use the Bruhat unitary, $\mathscr{U}^{\text{Bruhat}}(\theta)$, and the Borel unitary, $\mathscr{U}^{\text{Borel}}(\theta)$, both obtained in Section 4, and reported here in Figure 9.

The Borel unitary is explored here since, for some applications, it could be beneficial to have a weaker circuit, spanning less permutations, but in return that requires less classical parameters, and have a shallower depth relatively to the number of parameters.

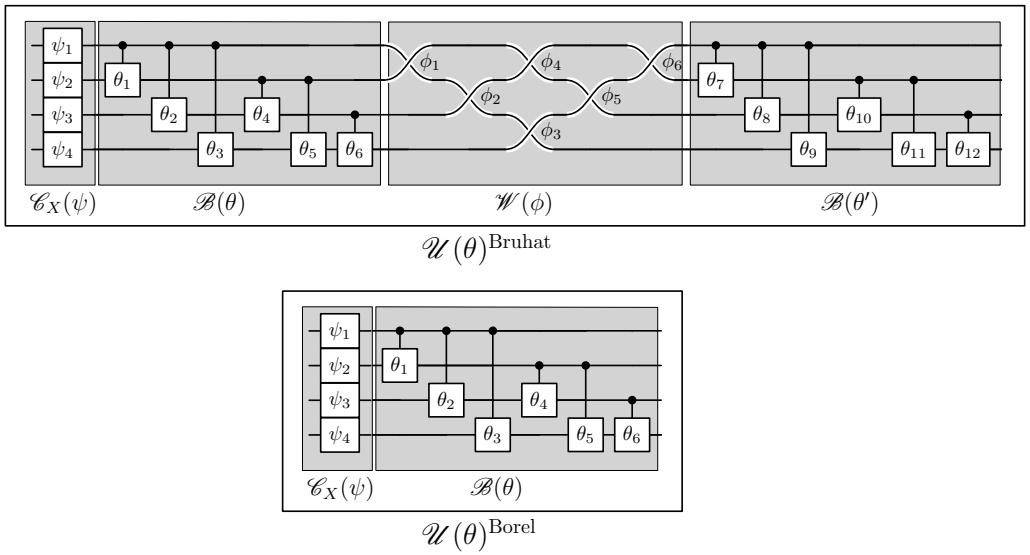


Figure 9: Depiction of the proposed ansatze for four qubits. With a slight abuse of notation, in both $\mathscr{U}(\theta)$, the parameters $\theta$ collect all the parameters in the ansatz. For example for Bruhat, $\theta := \{\psi, \theta, \phi, \theta'\}$. Single qubit gates with one angle indicate `rx` gates.

We also use a widely used unitary that has been proposed in the context of quantum machine learning: the Strong Entangling Layer (SEL) unitary [42] as baseline.

We report in Figures 10 and 11 the obtained spans performing one projection into the permutation set if $m > 0$. Both figures feature the number of spanned permutations depending on the number of parameters that we have in the circuit. For each number of parameters $\ell$, we consider either all the possible combinations of the parameters in $\{0, \pi\}$, i.e., $2^\ell$, or $250,000$ samples among them, whichever is the lowest. As such, we consider discrete parameters: this fits the presented theory and the practical results, as we introduce regularization terms in the optimization problem to force the parameters in $\{0, \pi\}$.

As we can observe in the figures, for $m = 0$, the Bruhat ansatz performs the best, obtaining the best we can achieve. For $m > 0$, all the ansatze perform similarly and offer a significant boost in spanned permutations, arriving close[4] to the total number of permutation $8! = 40,320$. Interestingly, the boost in performance, which we could call as quantum boost, since for $m > 0$ we are in a genuine quantum regime, is nearly identical for $m = 1$ and $m = 2$ and depends quite closely on the number of parameters as $2^\ell$ (i.e., the number of distinct possibilities). As such, for the same number of parameters $\ell$, quantum circuits with greater $m$'s will be shorter than the ones with lower $m$'s.

The similarities of the behavior for different $m$'s is due to the low number of considered qubits for $n = 8$. As discussed in Proposition 8, we know that the number of spanned permutation will be at most,

$$\min\left\{2^\ell, 2^{\frac{q'(q'+1)}{2}} \prod_{k=1}^{q'} (2^k - 1), n!\right\}, \quad q' = \log_2(n) + m,$$

thereby indicating a difference in performance for different $m$'s starting from $n = 16$. To test this, we run the Bruhat's ansatz with $q = 4$ and $m = 0, 1, 2$ over 400 millions different

[4]The larger span for $m = 1$ is obtained by Bruhat at $40,199$ and for $m = 2$ by Borel and SEL at $40,243$ with a Bruhat at $40,234$.

parameterizations[5] of $\theta$, and counted the number of distinct permutations returned. We observe in Figure 12 that increasing $m$ significantly improves the span of the circuit by one or two orders of magnitude, thereby empirically validating our approach.

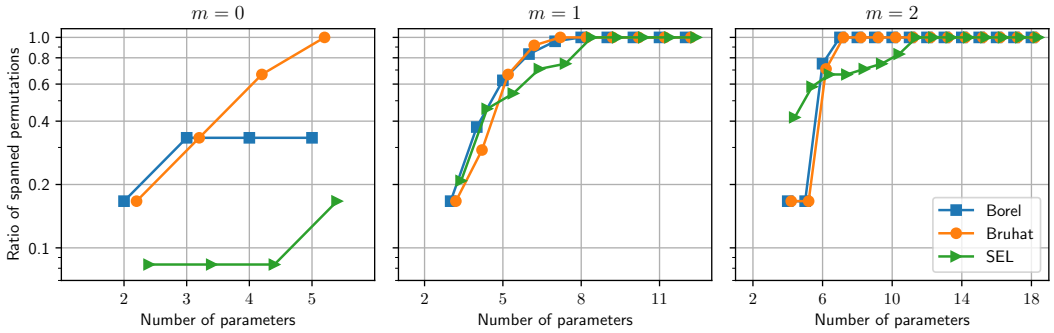


Figure 10: Ratio of the spanned permutations as a function of the number of parameters for $q = 2$, i.e., $n = 4$. The lines are shifted slightly horizontally for better readability. Recall that the Bruhat span of $n$ is 24 and $n! = 24$.

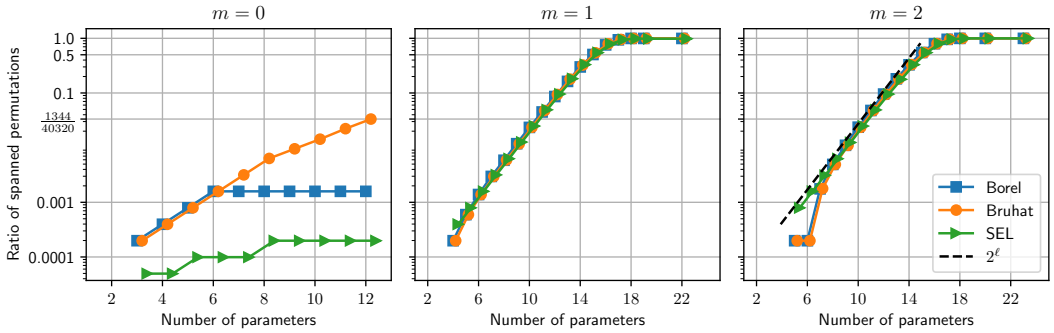


Figure 11: Ratio of the spanned permutations as a function of the number of parameters for $q = 3$, i.e., $n = 8$. The lines are shifted slightly horizontally for better readability. Recall that the Bruhat span of $n$ is $1,344$, $n! = 40,320$, while the Bruhat span of $n \times 2^1$ is $322,560$.

## 6 Application to optimization problems

We are now ready to see how to use our quantum circuit to the resolution of permutation-based optimization problems as Problem (1). We will focus on quadratic assignment problems and graph isomorphism.

In all the problems, we will use the ADAM optimizer to optimize classically for the $\theta$. We will also use a projection-onto-the-permutations technique discussed in Appendix C.

[5]Since $\theta \in \{0, \pi\}^\ell$, there are $2^\ell$ distinct parameterizations ($\sim 10^6$ for $m = 0$, $\sim 10^{10}$ for $m = 1$ and $\sim 10^{15}$ for $m = 2$) and $\sim 2 \times 10^{13}$ possible permutations. There are too many of them to explore exhaustively as we did for 2 and 3 qubits.

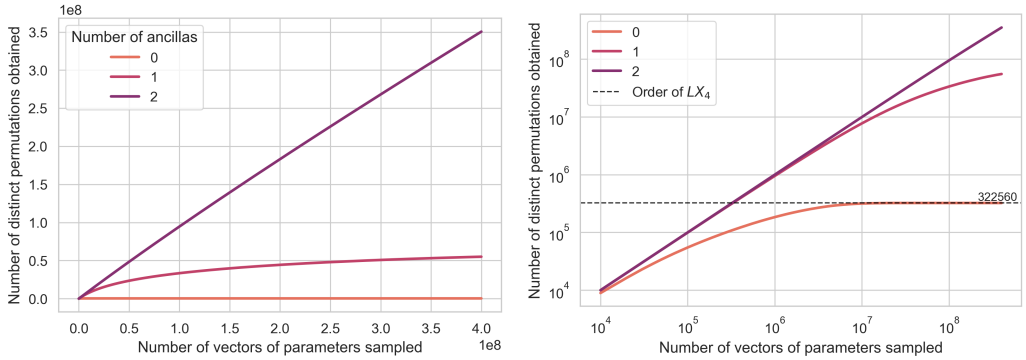


Figure 12: Number of permutations spanned as a function of the number of vectors of parameters sampled for $q = 4$, i.e., $n = 16$, plotted using a standard and a log-log scaling. The dashed horizontal line indicates the order of the group $LX_4$, i.e., the Bruhat span for $q = 4$ and $n = 16$.

## 6.1 Quadratic assignment problem

The first optimization problem we consider is the quadratic assignment problem [43] (or QAP for short), for which we can set in Problem (1), the objective function $f$ given by

$$f(P) := \operatorname{tr}\left(WPD^\top P^\top\right),$$

where $W$ and $D$ are two matrices in $\mathbb{R}^{n\times n}$. While there exist more general formulations of the problem (as well as polynomial generalization) [44, 45], the cost we have selected is general enough to consider most of the interesting instances appeared in the literature.

The quadratic assignment problem is an important problem in combinatorial optimization, with widespread use in practice and whose research stays very active [46]. For example, the travel salesperson problem is an instance of the QAP. Moreover, the QAP is NP-hard, and even finding an $\varepsilon$-approximation is intractable.

**Theorem 7** (Sahni and Gonzalez [47, 48])**.** *The quadratic assignment problem is strongly* NP*-hard. Moreover, for an any $\varepsilon > 0$, the existence of a polynomial time $\varepsilon$-approximation algorithm for the quadratic assignment problem implies* P = NP.

It was later shown by Pardalos, Rendl and Wolkowicz [49] that even finding a locally optimal solution, relatively to some specific neighborhood structure, is among the hardest local search problems. For a comprehensive examination of the quadratic assignment problem, see the survey of Burkard, Çela, Pardalos and Pitsoulis [48].

## 6.2 Quantum QAP solver: QuPer

We implement our code in python and use the package Pennylane [50] for quantum computations. All the circuits are simulated exactly. We have implemented three different ansatze, the Borel, the Bruhat, and the SEL.

As depicted in Figure 1, we can think of the whole quantum circuit as a mapping $\theta \mapsto \hat{P}_\theta$, with $\hat{P}_\theta$ being a doubly stochastic matrix. We therefore aim at minimizing the following objective function

$$\ell(\theta) := f(\hat{P}_\theta) = \operatorname{tr}\left(W\hat{P}_\theta D^\top \hat{P}_\theta^\top\right),$$

which is a non-standard non-convex continuous relaxation of the objective function of quadratic assignment problems. To guide the circuit to return doubly stochastic matrices that are closed to permutations, we regularize this objective function, which becomes

$$\ell_R(\theta) = \ell(\theta) + \frac{1}{10}\mathrm{st}(\hat{P}_\theta) + \frac{15}{100}S_\varepsilon(\hat{P}_\theta) + \frac{1}{100}\mathrm{ort}(\hat{P}_\theta)$$

with

$$\begin{cases} \mathrm{st}(\hat{P}_\theta) & = \sum_{j=1}^n \left(\sum_{i=1}^n \left[\hat{P}_\theta\right]_{ij} - 1\right)^2, \\ S_\varepsilon(\hat{P}_\theta) & = -\sum_{i,j=1}^n \left[\hat{P}_\theta \log\left(\hat{P}_\theta + \varepsilon\right)\right]_{ij}, \\ \mathrm{ort}(\hat{P}_\theta) & = \sum_{i,j=1}^n \left(\hat{P}_\theta^\top \hat{P}_\theta - I\right)^2. \end{cases}$$

Each additional term helps at forcing $\hat{P}_\theta$ to be a permutation, and at smoothing the loss landscape of the objective function.

We search for a minimum of $\ell_R$ using Adam with Nesterov momentum, so that the vector of parameters $\theta^{(k)}$ at step $k$ is updated according to the following rule

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \frac{\overline{\mu}^{(k)}}{\sqrt{\overline{\nu}^{(k)}} + \varepsilon} \quad \text{with} \quad \begin{cases} \mu^{(k)} = \beta_1 \mu^{(k-1)} + (1-\beta_1) g_k, \\ \nu^{(k)} = \beta_2 \nu^{(k-1)} + (1-\beta_2) g_k^2, \end{cases}$$

$$\text{as well as} \quad \overline{\mu}^{(k)} = \frac{\beta_1}{1-\beta_1^{k+1}}\mu^{(k)} + \frac{1-\beta_1}{1-\beta_1^k} g_t \quad \text{and} \quad \overline{\nu}^{(k)} = \frac{\nu^{(k)}}{1-\beta_2^k},$$

with $g_k = \nabla_\theta \ell_R(\theta^{(k-1)})$ and the usual hyperparameters fixed at

$$\eta = 0.005, \quad \beta_1 = 0.9, \quad \beta_2 = 0.999, \quad \text{and} \quad \varepsilon = 10^{-8}.$$

At every iteration we transform the double stochastic matrix $\hat{P}_\theta$ in two different ways:

- by solving a linear assignment problem using the Hungarian algorithm,
- by taking the permutation that alters the order of the entries of a randomly drawn vector the same way as the double stochastic matrix $\hat{P}_\theta$, as explained in Appendix C.

We then evaluate the original loss function $\ell$ at both permutations, keep the minimum of the two, and then compare with the best permutation currently known.

We initialize the parameters $\theta$ by drawing them around the center of the span with some uniform distribution. Since the ansatz with a given ancilla $m > 0$ contains the ansatz with ancilla $m-1$, we initialize the parameters $\theta$ of $m$ with the ones of $m-1$ whenever they are accessible (this last step is not necessary, but it helps having consistent results).

The resulting algorithm is presented in Algorithm 1 and is called QuPer.

**Algorithm 1** QuPer

**Require:** Ansatz, number of ancilla qubit $m^*$, number of iterations $I$

Define $\theta^{(0)}$ uniformly in $\frac{\pi}{2} \pm 0.05$ according to the chosen ansatz
**for** $m$ in $\{0, \dots, m^*\}$ **do**
  **while** convergence not reached **do** ▷ We set $I = 1000$ iterations
    $\theta^{(k+1)} \leftarrow \textsc{Adam}(\theta^{(k)})$ ▷ Adam calls the quantum circuit
    get $\tilde{P}$ by projecting $\hat{P}_{\theta^{(k+1)}}$ following Appendix C
    get $\tilde{v}$ by evaluating the cost classically
    **if** $\tilde{v} < v$ **then**
      $v \leftarrow \tilde{v}$, $P \leftarrow \tilde{P}$
  **return** $v$, $P$ ▷ We returned the results for each $m$
  pad the final $\theta$ with $\frac{\pi}{8}$ for the gates that will be added with the increment of $m$

### 6.2.1 Numerical results: QAPlib

We start by testing QuPer on instances of QAP collected in the QAPlib benchmark repository [51]. The instances range from small scale $n = 16$ to large scale $n = 256$ and beyond, the largest having $n = 729$ vertices. The instances are known to be hard, and for some of them we do not know the optimal permutation value. We consider all the solved instances in the benchmark for the following number of nodes $n \in \{16, 32, 64, 128, 256\}$, which have $\{12, 7, 2, 1, 1\}$ instances, respectively.

We compare QuPer with two baselines. First, we use a state-of-the-art classical heuristic [52], which is implemented in Python via SciPy. The heuristic solves a sequential convexification of the quadratic assignment problem, by relaxing the permutation constraint into a double stochastic matrix constraint and then using the Frank-Wolfe algorithm to solve the resulting continuous convex problem. Then they would project onto the permutation set by a linear assignment problem. In this context, the heuristic of [52] is close in spirit to our quantum heuristic, even though we span the set differently[6]. This algorithm has an empirical complexity of $O(n^3)$, with small leading constants ($\sim 10^{-9}$).

Second, we implement a naive random method, which draws permutations randomly. The number of random trials is set to be equal to the number of random projections we perform in QuPer: $50\lceil \frac{I}{10} \rceil$, which is generally small compared to the permutation set.

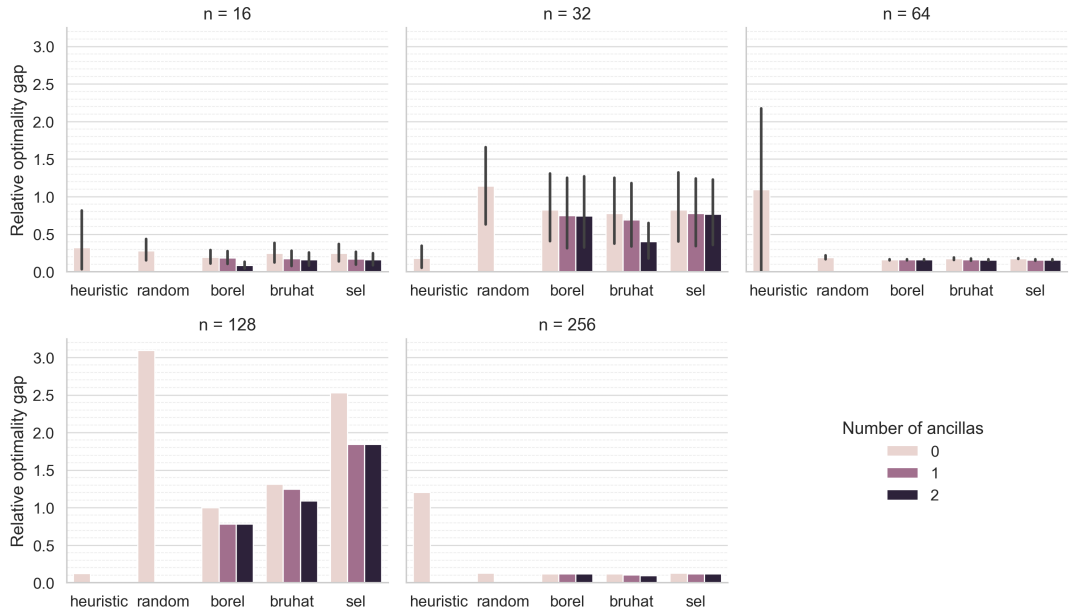


Figure 13: Numerical comparisons on instances taken from QAPlib. 'heuristic' denotes the classical heuristic [52] and 'random' the naive random method. We present here the mean over the instances and the 95% confidence interval. The full results are available in Appendix D.

In Figure 13 and Figure 14, we report the results for our algorithm in different settings as well as the other baselines. The results are reported in terms of the relative optimality gap, defined as,

$$\text{relative optimality gap} = \frac{f(\tilde{P}) - f^*}{f^*},$$

since for these instances we know the optimal solution value $f^*$.

[6] We notice that the classical heuristic in [52] is not necessarily the best on every problem, other exists in the literature, but it is quite robust, it can scale quite well and serves to us as a baseline.

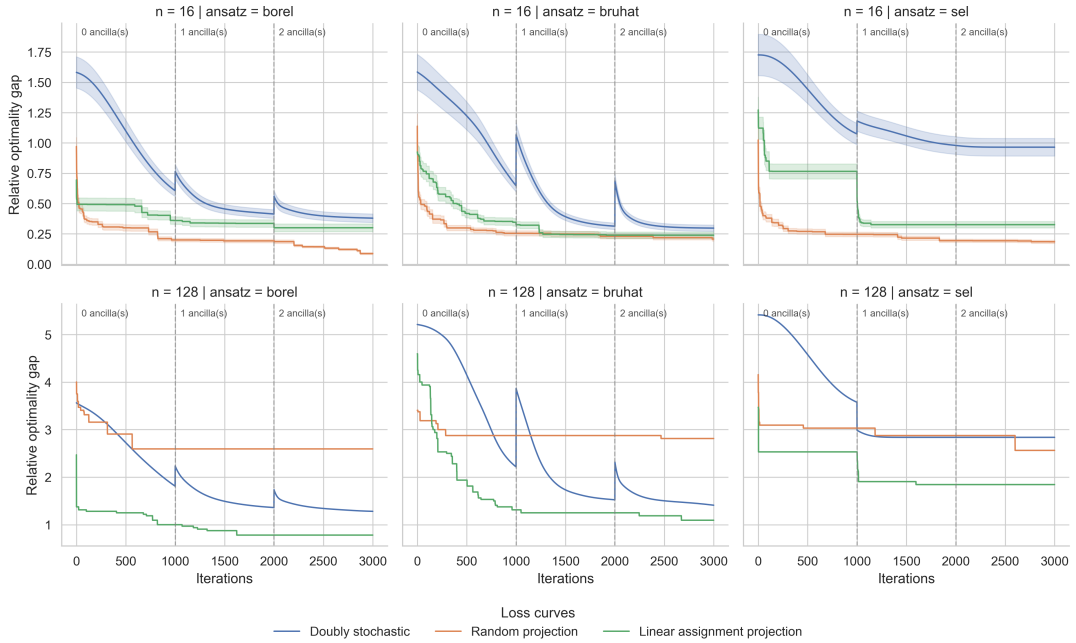


Figure 14: Numerical comparison on instances taken from QAPlib of the value of the loss functions, and the cost associated to the two different permutations obtained from the current doubly stochastic matrix returned by the circuit. Vertical dashed lines indicate the iterations at which an additional ancilla qubit is introduced in the variational ansatz. The full results are available in Appendix D.

As we can see, QuPer achieves good performance compared to the classical heuristic [52] and it is better than the naive random method. For some instances, QuPer outperforms the classical algorithm ($n = \{16, 64, 256\}$), for the others, the classical algorithm appears to be better. We can also see the quantum benefit of using a $m > 0$ (since the ansatz at $m + 1$ includes the one at $m$, a greater $m$ is always not worse, provided the classical optimizer can optimize the parameters exactly). The choice of ansatz seems to have a smaller influence on the result, but in general we think this is mainly due to the difficulty of the classical optimizer to optimize, rather than the expressivity of the ansatz.

Figure 13 and the accompanying Appendix D depict a quantum algorithm (QuPer) which can tackle large instances, it has good potential, and it could be in par with classical heuristics on difficult QAP instances. For reference, the largest circuit that we have used for the case $n = 256$, $m = 2$ has 20 qubits. Interesting future work here would be to bring QuPer onto HPC architectures and simulate large instances with large $m$, where we expect a clearer quantum benefit. Also, for large $m$, we may experience a more favorable optimization landscape, due to over-parametrization.

### 6.2.2 Numerical results: Random instances

We move on to evaluate QuPer on random instances, generated by drawing the elements of matrices $W$ and $D$ uniformly between 0 and 10.

We compare the naive random method and QuPer with the Borel ansatz as a percentage difference with respect to the classical heuristic. We report the results in Figure 15. As we can see, QuPer is competitive on random instances and it is generally better than the classical heuristic for $n = 8$. We believe that the fact that the performance of QuPer degrades for larger instances is mainly due to the classical optimizer, which fails to find a better minimum.

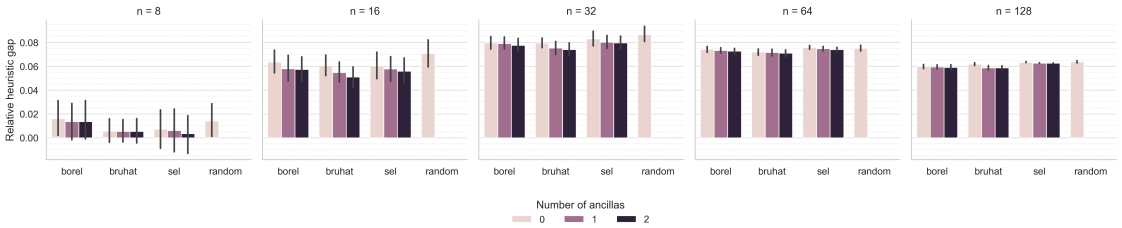


Figure 15: Numerical comparisons on 10 uniformly drawn random instances (5 for n = 128) with respect to the classical heuristic [52]. We present here the mean and the 95% confidence interval.

## 6.3 Graph isomorphism problem

The second optimization problem we study is the graph isomorphism problem [53], for which we can set in Problem (1), the objective function $f$ given by

$$f(P) := \|A - PBP^\top\|_{\mathrm{F}}^2,$$

where $A, B \in \{0,1\}^{n\times n}$ are the adjacency matrix of two undirected graphs $\mathcal{A}$ and $\mathcal{B}$, respectively, both with $n$ vertices and $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. An example of a graph isomorphism problem instance is given in Figure 16. We remark here that we could already tackle the harder sub-graph isomorphism problem with a similar formulation, but we focus on the easier graph isomorphism for simplicity.

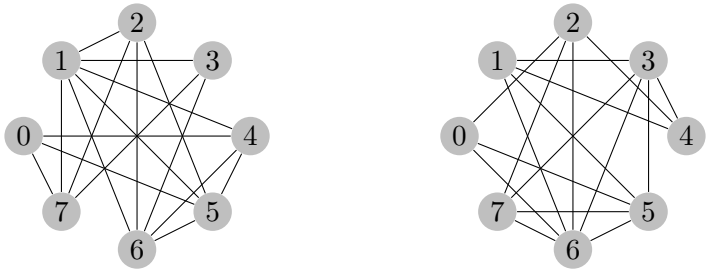


Figure 16: Two isomorphic graphs with $P = (4\ 6\ 7\ 0\ 1\ 3\ 5\ 2)$.

The graph isomorphism problem has numerous applications when data can be represented as networks [54, 55, 56]. Mathematically, the graph isomorphism problem is not known to be NP-hard in general [57], even though is still hard to solve in practice. However, many classical algorithms exist tailored to different real-word graphs and instances that can deal with graphs with thousands to millions of vertices and edges [58, 56, 59, 60].

We choose to focus here on the graph isomorphism to provide a quantum comparison of our approach to the amplitude encoding methodology presented in [12], which we expect to be somewhat of similar performance. It is also worth noticing that graph problems are reasonable to encode on photonic quantum computers [61], thereby making a good benchmark for future comparison of different technologies. Finally, we remark that a QAOA approach to this problem, tempted in [62], requires $O(n^2)$ qubits, which immediately requires $\approx 10^{12}$ qubit machines to hope to match current classical results.

### 6.3.1 Numerical results

We report numerical results to compare our QUPER algorithm to other baselines. We use the classical heuristic [52] as before as a classical baseline, and the variational algorithm

of [12] as a quantum baseline. In the latter, we substitute their ansatz for $P$ with our ansatze for a fairer comparison.

The graph isomorphism problem can be seen as a variant of a QAP [52]. In fact,

$$\begin{aligned}\|A - PBP^\top\|_{\mathrm{F}}^2 &= \mathrm{tr}((A - PBP^\top)^\top (A - PBP^\top)) \\ &= \mathrm{tr}((A - PBP^\top)(A - PBP^\top))\end{aligned}$$

by symmetry of the adjacency matrices $A$ and $B$, next

$$\begin{aligned}\|A - PBP^\top\|_{\mathrm{F}}^2 &= \mathrm{tr}(A^\top A - A^\top PBP^\top - PBP^\top A + PB^\top P^\top PBP^\top) \\ &= \mathrm{tr}(A^\top A) - \mathrm{tr}(A^\top PBP^\top) - \mathrm{tr}(PBP^\top A) + \mathrm{tr}(PB^\top BP^\top).\end{aligned}$$

We use the symmetry and the linearity of the trace operator to conclude with

$$\|A - PBP^\top\|_{\mathrm{F}}^2 = \mathrm{tr}(A^\top A) + 2\mathrm{tr}(AP(-B)P^\top) + \mathrm{tr}(B^\top B) \simeq \mathrm{tr}(AP(-B)P^\top).$$

Where $\simeq$ here stands for equivalence when optimized over. The latter is a QAP with $W = A$ and $D = -B$.

We generate graphs $\mathcal{A}$ randomly by sampling a Bernoulli distribution with a certain probability of 1 set to 0.5. The corresponding graph $\mathcal{B}$ is generated by drawing a random permutation and applying it to $\mathcal{A}$. The settings of the different algorithms is the same as in the QAP results. The step size of Adam is adjusted to 0.4.

In Figure 17, we report the numerical results for $n \in \{4, 6, 8, 12, 16\}$ comparing the variational quantum approach of [12] where we have used our ansatze: Borel$^{ms}$ and Bruhat$^{ms}$, for their $P$, and our QuPer with Borel and Bruhat with different ancillas $m$. The comparison is with a normalized version of the heuristic gap, i.e.,

$$\text{normalized heuristic gap} = \frac{\text{quantum result} - \text{heuristic classical result of [52]}}{n^2/2}.$$

The normalization is done with respect to the worst case error (and not with respect to the heuristic classical result, since it is often 0).

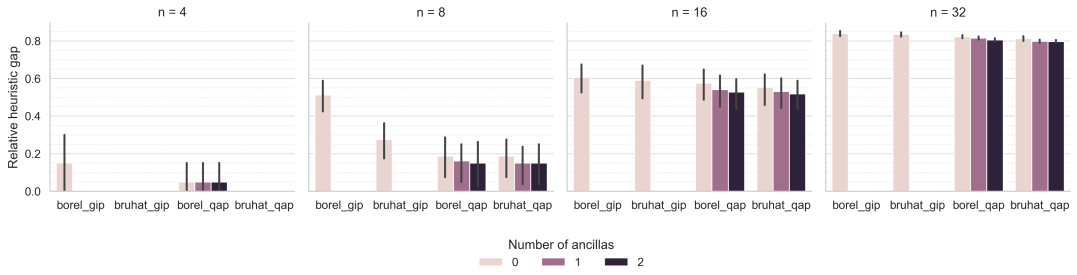


Figure 17: Numerical comparisons on 10 random instances of the graph isomorphism problem, with respect to the classical heuristic [52]. We present here the mean over the instances and the 95% confidence interval. The labels 'borel_gip' and 'bruhat_gip' represent the approach of [12] with these ansatze, while 'borel_qap' and 'bruhat_qap' represent the new circuits introduced in this paper.

By Figure 17, one can appreciate how our QuPer is competitive with respect the classical approach, especially for small instances, and it is better than the approach of [12]. In larger instances, QuPer is less competitive, but as before, we think this is mainly due to the difficulties of the classical optimizer Adam to solve non-convex problems.

We then conclude our simulations testing the case in which the graph $\mathcal{B}$ is generated by a permutation which is in the span of Bruhat. The results are presented in Figure 18, which shows better results than a general permutation, further advocating the need for an optimal spanning ansatz.

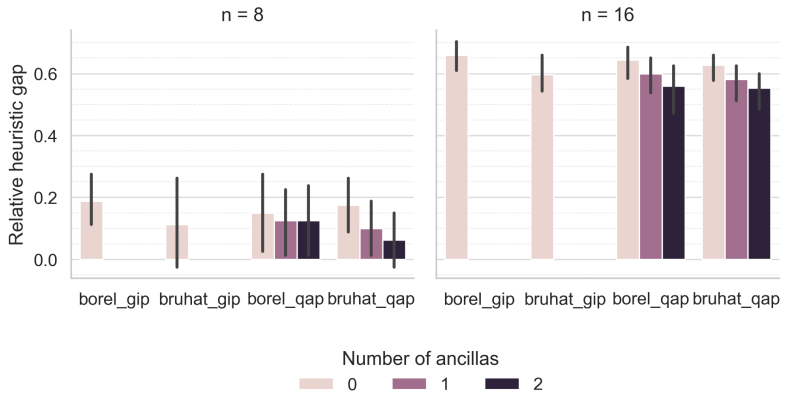


Figure 18: Numerical comparisons on 10 random instances of the graph isomorphism problem when the permutation lies in the span of Bruhat.

## 7 Conclusions

We have presented a variational quantum method to solve permutation-based optimization problems. The method is based on an optimal ansatz, with ancilla qubits, that can generate doubly-stochastic matrices. The resulting circuit is tunable and requires at least a number of qubits that scales only logarithmically with the number of variables. We have discussed and developed the theory behind the circuit generation and shown numerically its competitiveness on several optimization problems. We demonstrate that we can be in par with classical heuristics (typically for small instances but not only), and perform better than other quantum algorithms (on simpler problems, where those algorithms exist). The circuit that we generate is very general and adapted to any permutation-based optimization problems, which we will set forth to investigate in future work.

## References

[1] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm" (2014). arXiv:1411.4028.

[2] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices". Phys. Rev. X **10**, 021067 (2020).

[3] Amira Abbas, Andris Ambainis, Brandon Augustino, Andreas Bärtschi, Harry Buhrman, Carleton Coffrin, Giorgio Cortiana, Vedran Dunjko, Daniel J. Egger, Bruce G. Elmegreen, Nicola Franco, Filippo Fratini, Bryce Fuller, Julien Gacon, Constantin Gonciulea, Sander Gribling, Swati Gupta, Stuart Hadfield, Raoul Heese, Gerhard Kircher, Thomas Kleinert, Thorsten Koch, Georgios Korpas, Steve Lenk, Jakub Marecek, Vanio Markov, Guglielmo Mazzola, Stefano Mensa, Naeimeh Mohseni, Giacomo Nannicini, Corey O'Meara, Elena Peña Tapia, Sebastian Pokutta, Manuel Proissl, Patrick Rebentrost, Emre Sahin, Benjamin C. B. Symons, Sabine Tornow, Víctor Valls, Stefan Woerner, Mira L. Wolf-Bauwens, Jon Yard, Sheir Yarkoni, Dirk Zechiel, Sergiy Zhuk, and Christa Zoufal. "Challenges and opportunities in quantum optimization". Nature Reviews Physics **6**, 718–735 (2024).

[4] Sami Boulebnane and Ashley Montanaro. "Solving boolean satisfiability problems with the quantum approximate optimization algorithm". PRX Quantum **5**, 030348 (2024).

[5] Ruslan Shaydulin, Changhao Li, Shouvanik Chakrabarti, Matthew DeCross, Dylan Herman, Niraj Kumar, Jeffrey Larson, Danylo Lykov, Pierre Minssen, Yue Sun, Yuri Alexeev, Joan M. Dreiling, John P. Gaebler, Thomas M. Gatterman, Justin A. Gerber, Kevin Gilmore, Dan Gresh, Nathan Hewitt, Chandler V. Horst, Shaohan Hu, Jacob Johansen, Mitchell Matheny, Tanner Mengle, Michael Mills, Steven A. Moses, Brian Neyenhuis, Peter Siegfried, Romina Yalovetzky, and Marco Pistoia. "Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem". Science Advances **10**, eadm6761 (2024).

[6] Scott Aaronson. "Quantum advantage for NP approximation? for REAL this time?". `https://scottaaronson.blog/?p=8375` (2024). Blog post.

[7] Dimitris Bertsimas and Jack Dunn. "Optimal classification trees". Machine Learning **106**, 1039–1082 (2017).

[8] Daniel Rehfeldt, Thorsten Koch, and Yuji Shinano. "Faster exact solution of sparse MaxCut and QUBO problems". Mathematical Programming Computation **15**, 445–470 (2023).

[9] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. "Encoding patterns for quantum algorithms". IET Quantum Communication **2**, 141–152 (2021).

[10] Javier Gonzalez-Conde, Thomas W. Watts, Pablo Rodriguez-Grasa, and Mikel Sanz. "Efficient quantum amplitude encoding of polynomial functions". Quantum **8**, 1297 (2024).

[11] Marko J. Rančić. "Noisy intermediate-scale quantum computing algorithm for solving an n-vertex maxcut problem with log(n) qubits". Physical Review Research **5**, l012021 (2023).

[12] Nicola Mariella and Andrea Simonetto. "A quantum algorithm for the sub-graph isomorphism problem". ACM Transactions on Quantum Computing **4**, 1–34 (2023).

[13] Zhihui Wang, Nicholas C. Rubin, Jason M. Dominy, and Eleanor G. Rieffel. "Xy mixers: Analytical and numerical results for the quantum alternating operator ansatz". Physical Review A **101**, 012320 (2020).

[14] Franz Georg Fuchs, Kjetil Olsen Lye, Halvor Møll Nilsen, Alexander Johannes Stasik, and Giorgio Sartor. "Constraint preserving mixers for the quantum approximate optimization algorithm". Algorithms **15**, 202 (2022).

[15] Hannes Leipold, Federico M. Spedalieri, Stuart Hadfield, and Eleanor Rieffel. "Imposing constraints on driver hamiltonians and mixing operators: From theory to practical implementation" (2024). arXiv:2407.01975.

[16] Samuel Marsh and Jingbo B. Wang. "Combinatorial optimization via highly efficient quantum walks". Physical Review Research **2**, 023302 (2020).

[17] Claudio Gambella and Andrea Simonetto. "Multiblock admm heuristics for mixed-binary optimization on classical and quantum computers". IEEE Transactions on Quantum Engineering **1**, 1–22 (2020).

[18] Maike Drieb-Schön, Kilian Ender, Younes Javanmard, and Wolfgang Lechner. "Parity quantum optimization: Encoding constraints". Quantum **7**, 951 (2023).

[19] Hyakka Nakada, Kotaro Tanahashi, and Shu Tanaka. "Inductive construction of variational quantum circuit for constrained combinatorial optimization" (2025). arXiv:2501.03521.

[20] Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, and Jose A. Lozano. "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems". Progress in Artificial Intelligence **1**, 103–117 (2012).

[21] Nicola Mariella, Albert Akhriev, Francesco Tacchino, Christa Zoufal, Juan Carlos Gonzalez-Espitia, Benedek Harsanyi, Eugene Koskin, Ivano Tavernelli, Stefan Woerner, Marianna Rapsomaniki, Sergiy Zhuk, and Jannis Born. "Quantum theory and application of contextual optimal transport". In Proceedings of the 41st International Conference on Machine Learning. (2024). arXiv:2402.14991v3.

[22] Fajwel Fogel, Rodolphe Jenatton, Francis Bach, and Alexandre d'Aspremont. "Convex relaxations for permutation problems". SIAM Journal on Matrix Analysis and Applications **36**, 1465–1488 (2015).

[23] Alexander Barvinok. "Approximating orthogonal matrices by permutation matrices". Pure and Applied Mathematics Quarterly **2**, 943–961 (2006).

[24] Marc Bataille. "Quantum circuits of `cnot` gates: optimization and entanglement". Quantum Information Processing **21**, 269 (2022).

[25] Dmitri Maslov and Martin Roetteler. "Shorter stabilizer circuits via bruhat decomposition and quantum circuit transformations". IEEE Transactions on Information Theory **64**, 4729–4738 (2018).

[26] Dimitri P. Bertsekas. "Auction algorithms for network flow problems: A tutorial introduction". Computational Optimization and Applications **1**, 7–66 (1992).

[27] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization" (2014). arXiv:1412.6980.

[28] David Steven Dummit and Richard M. Foote. "Abstract algebra". Wiley Hoboken. (2004). Third edition.

[29] David Lawrence Johnson. "Presentations of groups". Number 15 in London Mathematical Society Student Texts. Cambridge university press. (1997).

[30] Robert Steinberg. "Lectures on Chevalley groups". Volume 66. American Mathematical Society. (2016).

[31] François Bruhat and Jacques Tits. "Groupes réductifs sur un corps local: I. données radicielles valuées". Publications Mathématiques de l'Institut des Hautes Études Scientifiques **41**, 5–251 (1972).

[32] Nicolas Bourbaki. "Groupes et algèbres de Lie: Chapitres 4, 5 et 6". Springer Berlin, Heidelberg. (1968).

[33] Jacques Tits. "Buildings of Spherical Type and Finite BN-Pairs". Springer Berlin Heidelberg. (1974).

[34] Emil Artin. "Geometric algebra". Wiley Classics Series. (1988).

[35] Anders Björner and Francesco Brenti. "Combinatorics of Coxeter groups". Volume 231. Springer. (2005).

[36] Elisa Bäumer and Stefan Woerner. "Measurement-based long-range entangling gates in constant depth". Physical Review Research **7**, 023120 (2025).

[37] Alexander Müller-Hermes and Ion Nechita. "Operator Schmidt ranks of bipartite unitary matrices". Linear Algebra and its Applications **557**, 174–187 (2018).

[38] Mark W. Coffey and Ron Deiotte. "Relation of operator Schmidt decomposition and CNOT complexity". Quantum Information Processing **7**, 117–124 (2008).

[39] Adam Glos, Aleksandra Krawiec, and Zoltán Zimborás. "Space-efficient binary optimization for variational quantum computing". npj Quantum Information **8**, 39 (2022).

[40] Lin Chen and Li Yu. "Entanglement cost and entangling power of bipartite unitary and permutation operators". Physical Review A **93**, 042331 (2016).

[41] Tristan Benoist and Ion Nechita. "On bipartite unitary matrices generating subalgebra-preserving quantum operations". Linear Algebra and its Applications **521**, 70–103 (2017).

[42] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. "Circuit-centric quantum classifiers". Physical Review A **101**, 032308 (2020).

[43] Tjalling C. Koopmans and Martin Beckmann. "Assignment problems and the location of economic activities". Econometrica: Journal of the Econometric Society **25**, 53–76 (1957).

[44] Eugene L. Lawler. "The quadratic assignment problem". Management Science **9**, 586–599 (1963).

[45] Panos M. Pardalos and Leonidas S. Pitsoulis. "Nonlinear Assignment Problems". Combinatorial Optimization. Springer Netherlands. (1999).

[46] Zhentao Tan and Yadong Mu. "Learning solution-aware transformers for efficiently solving quadratic assignment problem". In Proceedings of the 41st International Conference on Machine Learning. (2024). arXiv:10.48550/arXiv.2406.09899.

[47] Sartaj Sahni and Teofilo Gonzalez. "P-complete approximation problems". Journal of the ACM **23**, 555–565 (1976).

[48] Rainer Ernst Burkard, Eranda Çela, Panos M. Pardalos, and Leonidas S. Pitsoulis. "The quadratic assignment problem". In Ding-Zhu Du and Panos M. Pardalos, editors, Handbook of Combinatorial Optimization. Kluwer Academic Publishers (1998).

[49] Panos M. Pardalos, Franz Rendl, and Henry Wolkowicz. "The quadratic assignment problem: A survey and recent developments". Technical report. Faculty of Mathematics, University of Waterloo (1994). url: `https://www.math.uwaterloo.ca/~hwolkowi/henry/reports/qapsurvey94.pdf`.

[50] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. "Pennylane: Automatic differentiation of hybrid quantum-classical computations" (2018). arXiv:10.48550/arXiv.1811.04968.

[51] Rainer E. Burkard, Stefan E. Karisch, and Franz Rendl. "QAPLIB – A Quadratic Assignment Problem Library". Journal of Global Optimization **10**, 391–403 (1997).

[52] Joshua T. Vogelstein, John M. Conroy, Vince Lyzinski, Louis J. Podrazik, Steven G. Kratzer, Eric T. Harley, Donniell E. Fishkind, R. Jacob Vogelstein, and Carey E. Priebe. "Fast approximate quadratic programming for graph matching". PLOS ONE **10**, e0121002 (2015).

[53] Béla Bollobás. "Modern Graph Theory". Springer New York. (1998).

[54] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. "Thirty years of graph matching in pattern recognition". International Journal of Pattern Recognition and Artificial Intelligence **18**, 265–298 (2004).

[55] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. "A subgraph isomorphism algorithm and its application to biochemical data". BMC Bioinformatics **14**, S13 (2013).

[56] Jinsoo Lee, Wook-Shin Han, Romans Kasperovics, and Jeong-Hoon Lee. "An in-depth comparison of subgraph isomorphism algorithms in graph databases". Proceedings of the VLDB Endowment **6**, 133–144 (2012).

[57] Vikraman Arvind and Johannes Köbler. "Graph isomorphism is low for ZPP(NP) and other lowness results". In STACS 2000. Pages 431–442. Springer Berlin Heidelberg (2000).

[58] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. "A (sub)graph isomorphism algorithm for matching large graphs". IEEE Transactions on Pattern Analysis and Machine Intelligence **26**, 1367–1372 (2004).

[59] Yonathan Aflalo, Alexander Bronstein, and Ron Kimmel. "On convex relaxation of graph isomorphism". Proceedings of the National Academy of Sciences **112**, 2942–2947 (2015).

[60] Chems Eddine Nabti. "Subgraph Isomorphism Search in Massive Graph Data". PhD thesis. Université de Lyon. (2018). url: `https://theses.hal.science/tel-01781831v1`.

[61] Rawad Mezher, Ana Filipa Carvalho, and Shane Mansfield. "Solving graph problems with single photons and linear optics". Physical Review A **108**, 032405 (2023).

[62] Cristian S. Calude, Michael J. Dinneen, and Richard Hua. "QUBO formulations for the graph isomorphism problem and related problems". Theoretical Computer Science **701**, 54–69 (2017).

# Appendices

## A Computation complexity considerations

We discuss briefly the computational complexity of [11]. The variational algorithm is an amplitude encoding, requiring $q = \log_2(n)$ qubit to encode a max-cut of $n$ variables. However to compute the observable, one needs to evaluate $4^q = n^2$ circuits, weighted them with some pre-compute coefficient (involving the trace of a matrix-matrix multiplication) and then sum them. The total cost is $O(n^3)$. In comparison, a classical algorithm that evaluates $x^\top Qx$ requires only $O(n^2)$ floating point operations, thereby making the algorithm classically simulable.

## B Construction of parametrized cx gate

The reason why we cannot use the controlled rotation `x`, or `crx`, gate in our circuit is that it introduces some relative phase shift between the control qubit and the target qubit. To overcome this, we beforehand apply a phase gate on the control qubit, to anticipate the introduction of the relative phase with the `crx` gate. These two gates can be written as

$$\mathtt{p}(\theta) = |0\rangle\langle 0| + e^{i\theta}\,|1\rangle\langle 1| \qquad \text{and} \qquad \mathtt{crx}(\theta) = |0\rangle\langle 0| \otimes \mathtt{id} + |1\rangle\langle 1| \otimes \mathtt{rx}(\theta).$$

When evaluated at $\theta = 0$, the `crx` gate does not introduce any relative phase as it acts like the identity, but when it is evaluated at $\theta = \pi$ it becomes

$$\mathtt{crx}(\pi) = |0\rangle\langle 0| \otimes \mathtt{id} + |1\rangle\langle 1| \otimes \mathtt{rx}(\pi) = |0\rangle\langle 0| \otimes \mathtt{id} - i\,|1\rangle\langle 1| \otimes \mathtt{x}.$$

We fix this issue by applying the phase gate on the control qubit with half of $\theta$. We have

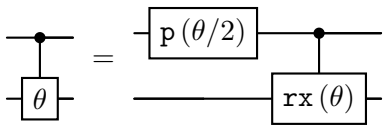


Figure 19: Construction of the `crx` gate introducing no relative phase, using the notation of Figure 3.

claimed that this gate acts like the identity when $\theta = 0$, which is straightforward, and that it acts like a `cx` gate when $\theta = \pi$. We prove the latter by direct computation. We denote by $k$ the control qubit and by $l$ the target qubit.

$$\begin{aligned}
\mathtt{crx}_{kl}(\pi) \cdot \mathtt{p}_k(\pi/2) &= \Big( |0\rangle\langle 0|_k \otimes \mathtt{id}_l + |1\rangle\langle 1|_k \otimes \mathtt{rx}_l(\pi) \Big) \Big( \Big( |0\rangle\langle 0|_k + e^{i\frac{\pi}{2}}\,|1\rangle\langle 1|_k \Big) \otimes \mathtt{id}_l \Big) \\
&= \Big( |0\rangle\langle 0|_k \otimes \mathtt{id}_l - i\,|1\rangle\langle 1|_k \otimes \mathtt{x}_l \Big) \Big( \left( |0\rangle\langle 0|_k + i\,|1\rangle\langle 1|_k \right) \otimes \mathtt{id}_l \Big) \\
&= |0\rangle\langle 0|_k \otimes \mathtt{id}_l - i \cdot i\,|1\rangle\langle 1|_k \otimes \mathtt{x}_l \\
&= \mathtt{cx}_{kl}.
\end{aligned}$$

## C Projection onto permutations

Let $\hat{P}$ be a doubly-stochastic matrix of dimension $n$, i.e., a matrix in $[0,1]^{n\times n}$, whose columns and rows sum to 1. Projecting such a matrix onto the permutation set, we can

either use a linear assignment problem (a convex problem) as,

$$\min_{P\in\Pi_n} \frac{1}{2}\|\hat{P}-P\|_{\mathrm{F}}^2 = -\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{P}^\top P)_{ij},$$

that we solve using the *Hungarian algorithm*, or a randomized algorithm described in [23, 22]. We explain here the latter as it has in general a better performance on small instances and is less known.

The randomized algorithm idea is apply $\hat{P}$ to a sorted random vector and see how the ordering changes. Then one can find a permutation that achieve the same permutation of ordering. Let $v$ the initial random vector, and $\psi(v)$ its ordering, the idea is to find the permutation $P$ for which,

$$\psi\left(\hat{P}v\right) = P\psi(v).$$

In the simulations, we choose to start with a random vector $v$ designed as $v_i := 2^i$. Each coefficient of $v$ is at least twice or half another coefficient, which prevents values of $\hat{P}$ which are quite far from 1 to unexpectedly change the order of the returned vector.

Next, we randomly permute $v$, apply $\hat{P}$ to it, and construct $P$ by looking at the order of the returned vector. We do this 50 times, so we obtain less than 50 unique permutations, closely resembling the one we are looking for. From these, we simply take the one minimizing the objective function.

This approach is efficient, and allows for the exploration of a larger neighborhood of the current relaxed solution, compared to the projection that we can have by solving a linear assignment problem, which does not take into consideration the cost function.

## D Full results for QAP

The following figures show the average relative optimality gaps for the different algorithms. One plot corresponds to a given size of instances. For the size 128 and 256, there is a unique instance, so there is no vertical bar indicating the interval of confidence.

The following tables includes all the results we obtained on the QAPlib instances of size a power of two. Their names are written in the first column. So each line represents a single instance, and the columns represent the different algorithms used to solve the aforementioned instances.

| | Classical | | | Borel | | | Bruhat | | | SEL | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instances | Exact | Heur. | Rand. | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| esc16a | 68 | 70 | 84 | 84 | 80 | 80 | 80 | 78 | 78 | 84 | 78 | 78 |
| esc16b | 292 | 320 | 292 | 294 | 294 | 294 | 296 | 292 | 292 | 294 | 294 | 294 |
| esc16c | 160 | 168 | 192 | 178 | 178 | 168 | 182 | 182 | 182 | 188 | 184 | 180 |
| esc16d | 16 | 62 | 26 | 24 | 24 | 16 | 28 | 16 | 16 | 28 | 24 | 24 |
| esc16e | 28 | 30 | 30 | 32 | 32 | 32 | 32 | 32 | 32 | 34 | 34 | 32 |
| esc16f | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| esc16g | 26 | 30 | 36 | 30 | 30 | 28 | 34 | 34 | 34 | 36 | 36 | 36 |
| esc16h | 996 | 1518 | 1066 | 1026 | 1026 | 1026 | 1026 | 1026 | 1026 | 1036 | 1030 | 1030 |
| esc16i | 14 | 14 | 26 | 18 | 18 | 14 | 22 | 22 | 20 | 18 | 14 | 14 |
| esc16j | 8 | 8 | 12 | 12 | 12 | 10 | 12 | 12 | 12 | 12 | 10 | 10 |
| nug16a | 1610 | 1660 | 1916 | 1860 | 1860 | 1854 | 1878 | 1778 | 1778 | 1866 | 1866 | 1844 |
| nug16b | 1240 | 1282 | 1516 | 1486 | 1446 | 1446 | 1434 | 1434 | 1434 | 1488 | 1484 | 1484 |

Figure 20: The value of the cost function on different instances of size 16. The integers below ansatze names indicate the number $m$ of ancilla qubits used.

| | Classical | | | Borel | | | Bruhat | | | SEL | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instances | Exact | Heur. | Rand. | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| kra32 | 88900 | 92930 | 126020 | 120710 | 118710 | 115710 | 117600 | 114680 | 114680 | 122390 | 122390 | 122390 |
| esc32a | 130 | 160 | 338 | 318 | 304 | 304 | 282 | 258 | 258 | 306 | 306 | 306 |
| esc32b | 168 | 196 | 376 | 264 | 256 | 256 | 256 | 256 | 256 | 352 | 304 | 304 |
| esc32c | 642 | 650 | 810 | 702 | 694 | 694 | 706 | 692 | 692 | 776 | 742 | 742 |
| esc32d | 200 | 226 | 294 | 262 | 262 | 262 | 262 | 258 | 254 | 280 | 280 | 272 |
| esc32e | 2 | 2 | 6 | 6 | 6 | 6 | 6 | 6 | 2 | 6 | 6 | 6 |
| esc32g | 6 | 10 | 18 | 12 | 10 | 10 | 12 | 10 | 10 | 8 | 8 | 8 |

Figure 21: The value of the cost function on different instances of size 32. The integers below ansatze names indicate the number $m$ of ancilla qubits used.

| | Classical | | | Borel | | | Bruhat | | | SEL | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inst. | Exact | Heur. | Rand. | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| sko64 | 48498 | 49102 | 56450 | 56564 | 56482 | 56482 | 56120 | 55592 | 55592 | 56880 | 56612 | 56582 |
| tai64c | 1855928 | 5893540 | 2258166 | 2141126 | 2141126 | 2141126 | 2211470 | 2181600 | 2161392 | 2187554 | 2133174 | 2133174 |
| esc128 | 64 | 72 | 262 | 128 | 114 | 114 | 148 | 144 | 134 | 226 | 182 | 182 |
| tai256c | 44759294 | 98685678 | 50647386 | 50129154 | 50129154 | 50129154 | 50093960 | 49437762 | 49035944 | 50642900 | 50140242 | 50140242 |

Figure 22: The value of the cost function on different instances of size 64, 128 and 256. The integers below ansatze names indicate the number $m$ of ancilla qubits used.

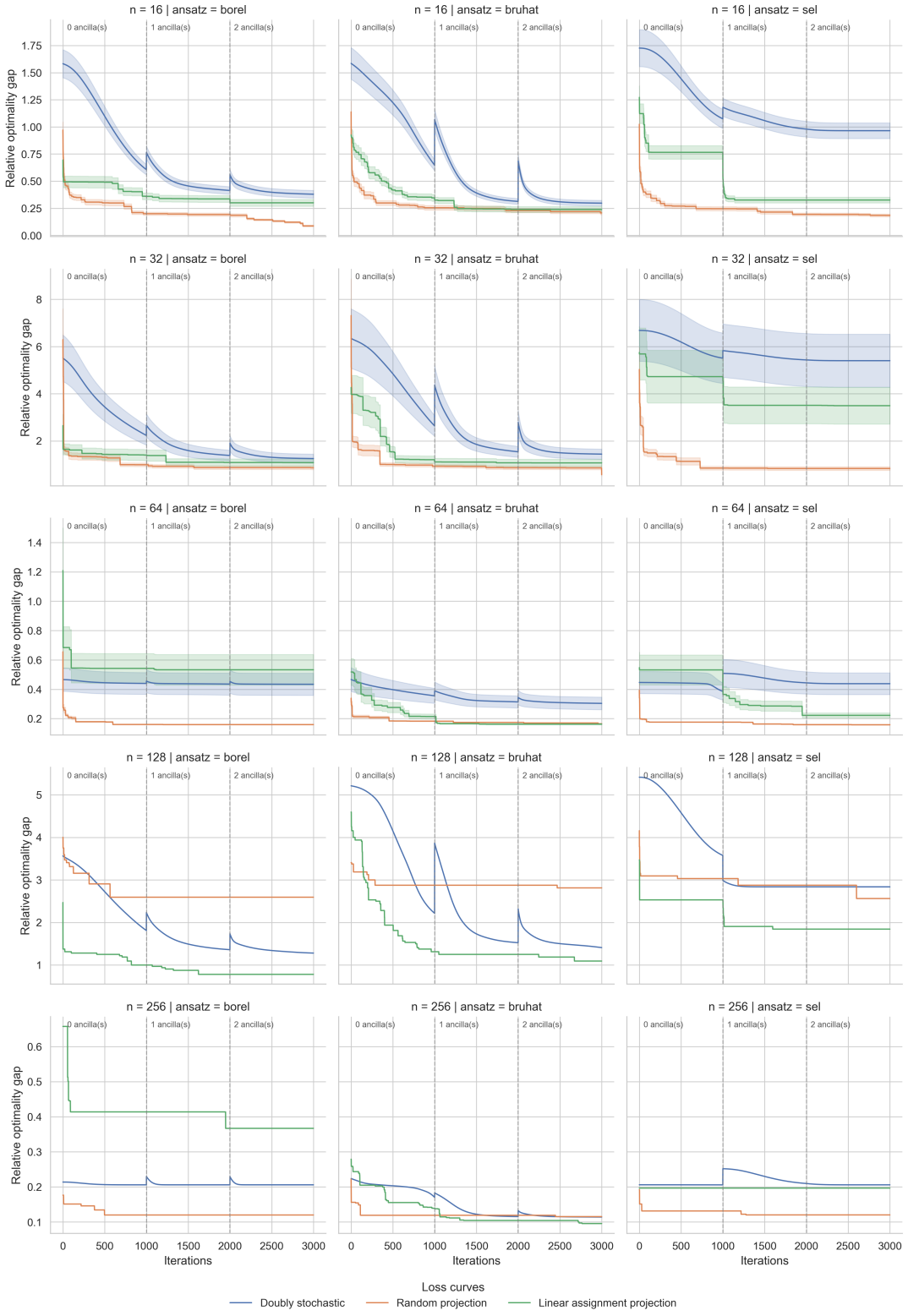


Figure 23: Values of the cost functions during the optimization process. Each row correspond to a specific size of instances. For 128- and 256-vertex graphs, there is a unique instance. Vertical dashed lines indicate the iterations at which an additional ancilla qubit is introduced in the variational ansatz.