Free Job-Shop Scheduling With Hardcoded Constraints

Gereon Kölmann, a) Lennart Binkowski, b) and René Schwonnek c)
Institut für Theoretische Physik, Leibniz Universität Hannover

Hardcoding constrained optimization problems into a variational quantum algorithm often turns out to be a challenging task. In this work we provide a solution for the class of free job-shop problems (FJSP), which we achieve by rigorously employing the symmetries of the classical problem.

An established approach for hardcoding the constraints of the closely related traveling salesman problem (TSP) into mixer Hamiltonians was recently given by Hadfield et al.’s Quantum Alternating Operator Ansatz (QAOA). For FJSP, which contains TSP as a subclass, we show that desired properties of similarly constructed mixers can be directly linked to a purely classical object: the group of feasibility-preserving bit value permutations. We also outline a generic way to construct QAOA-like-mixers for these problems. We further propose a new variational quantum algorithm that incorporates the underlying group structure more naturally, and provide a concrete numerical example. Unlike the QAOA, our algorithm allows bounding the amount of parameters necessary to reach every feasible solution from above: If \( J \) jobs are to be distributed, we need at most \( J(J - 1)^2/2 \) parameters.

I. INTRODUCTION

Logistic and scheduling tasks are a major branch within the collection of hard optimization problems with relevant industrial application. A prominent representative of those is the free job-shop scheduling problem FJSP\((M,T,J)\) which we consider in this work: Given \( M \) machines with \( T \) time slots each, one has to distribute \( J \) jobs such that every job gets performed precisely once and no position is filled with more than one job. Not only is FJSP at the mathematical core of many real world problems, it also prominently incorporates the well-known traveling salesman problem (TSP) as a subclass, i.e. we have TSP = FJSP\((T,1,T)\).

Its practical relevance, and the fact that solving an instance of FJSP can easily turn out to be a hard task for classical computers, make FJSP an interesting target for the application of quantum algorithms. Confronted with the restricted capabilities of available quantum computer architectures, the class of variational quantum algorithms\(^1\)–\(^5\) (VQAs) is here receiving a particular amount of attention, since it promises to yield tools for tackling computational challenges within the NISQ\(^6\)–\(^7\) era.

There are, however, certain hurdles to overcome. The basic input for applying a VQA to an optimization problem is an encoding of a problems objective function into a multi-qubit objective Hamiltonian \( H_f \) such that - in case of a minimization problem - optimal solutions correspond to the smallest expectation value attainable by a quantum state\(^8\). In contrast to other popular problems, like MAX-CUT or 3-Sat, we have that FJSP additionally demands us to not only consider the encoding of an objective function rather than also the encoding of further constraints. As a consequence a VQA ideally has to perform its optimization only on a subset of ‘allowed/feasible’ quantum states, i.e. those that respect constraints. The major contribution of this work is a systematic analysis of symmetry structures in FJSP that at the end will enable us to design a class of VQA algorithms that will, by construction, only optimize over feasible states.

In general, there are two strategies for handling constrained optimization problems:

- **softcoded** constraints. Any assignment is considered feasible, but the constraints enter the objective function as additional terms and penalize assignments that were originally infeasible. The modified objective function results in a more complex objective Hamiltonian. In exchange, the ground state search can be conducted in the entire qubit space without the necessity of preserving feasibility throughout the routine.

- **hardcoded** constraints. The objective function (and thus the objective Hamiltonian) is left unmodified but the ground state search is restricted to the subspace corresponding to feasible solutions. Preservation of feasibility is typically ensured by additional gates representing the classical constraints.

Most VQAs such as the variational quantum eigensolver\(^3\) or the quantum approximate optimization algorithm\(^9\) are originally formulated for unconstrained problems and thus are merely applicable to softcoded instances. However, several case studies indicate that softcoding the constraints often either leads to suboptimal optimization landscapes or issues with feasibility\(^10\)–\(^12\). This is why, quantum algorithms with built-in possibilities for hardcoding constraints received increasing interest. Most notably, Hadfield et al.\(^13\) extended the quantum approximate optimization algorithm to the so-called quantum alternating operator ansatz (QAOA) which can also be applied to hardcoded constrained optimization problems. Unlike its predecessor, the QAOA is formu-
lated problem-dependent and is thus sensitive to different feasibility structures.

Albeit there is a massive catalogue\textsuperscript{14} of classical constraint analysis available for these problems, concrete instances were mainly heuristically constructed, not exploiting the whole underlying feasibility structure. One particular interesting construction for scheduling-type problems is the constraint graph model\textsuperscript{15,19}. Here the bit strings are identified with vertices, joint by edges corresponding to the constraints. This allows investigating the feasibility structure with well-known results from graph theory. Most notably, one can identify graph automorphisms with feasibility-preserving bit permutations. With this work, we incorporate the whole structure into a more refined view on the QAOA and also come up with a new VQA design for FJSP-instances.

In Section III we apply the constraint graph model to the general FJSP. We first embed the notions of solutions and solution-preserving functions into the graph-theoretical language. Utilizing this additional point of view, we fully characterize the group $F$ of feasibility-preserving bit permutations. Furthermore, we uncover block structures within the FJSP-constraints. This ultimately reveals that $F$ acts transitively on the set of all solutions.

In Section IV we then draw the connection between the classical description of $F$ and specific VQA designs. Firstly, we review how the QAOA works in general and proceed with a discussion of its main ingredients. We give refined definitions for ‘phase separator’ and ‘mixer’ gates and detail a general construction for suitable mixers from elements of $F$. In particular, the transitive action of $F$ is directly translated into substantial mixing properties. Second, we introduce a new VQA suitable for FJSP-instances. It is fundamentally based on decomposing bit value permutations into products of transpositions. In contrast to the QAOA, we can bound the number of parameters necessary to reach every possible solutions: While the number of FJSP-solutions is $O(J!)$, only $O(J^3)$ parameters are necessary.

We test our new VQA on a FJSP($2, 4$)-instance. The results are presented in Section V. We observe a strong sampling of optimal solution states but also residual sampling of ‘nearby’ states, that is, for computational basis states with small Hamming distance from the optimal state. However, we explain this behavior in context of our method.

### II. PRELIMINARIES

For the readers convenience we will review the basic notion of combinatorial optimization problems (COPs), especially the free job-shop scheduling problem, briefly introduce the constraint graph model, and cover the very basics of problem encoding onto quantum computers. Throughout this work, we will use the shorthand $[N] := \{1, \ldots, N\}$ where $N$ is any natural number.

1. Constrained Combinatorial Optimization

In the following, we restrict to minimization problems, as maximization tasks may be considered analogously. Then a generic COP of size $N$ is of the form

$$\min_{z \in Z(N)} f(z) \quad \text{s.t.} \quad c_a(z) = 1, \quad a \in [A],$$

where

- $Z(N) := \{0, 1\}^N$ is the bit strings of length $N$,  
- $f : Z(N) \rightarrow \mathbb{R}$ is the objective function, and  
- $c_a : Z(N) \rightarrow \{0, 1\}$ are the constraints.

Accordingly, a bit string $z$ is said to fulfill a constraint $c_a$ iff $c_a(z) = 1$. We will refer to COPs formally as triples $(N, f, \{c_a\})$. For a given COP $\mathcal{C}$ we define its solution set as the set of all bit strings fulfilling every constraint:

$$S(\mathcal{C}) := \{z \in Z(N) : c_a(z) = 1, \quad a \in [A]\},$$

Furthermore, the optimal solution set is the set of all bit string minimizing $f$, i.e.,

$$S_{\text{min}}(\mathcal{C}) := \left\{z \in S(\mathcal{C}) : f(z) = \min_{z' \in S(\mathcal{C})} f(z') \right\} \subseteq S(\mathcal{C}).$$

We now examine particular types of constraints: For two bit strings $z, z' \in Z(N)$ the bit-wise and operation produces a new bit string $z \wedge z' \in Z(N)$. $|z|$ denotes the Hamming weight of $z$. Furthermore, for a subset $I \subseteq [N]$ let $z_I \in Z(N)$ with $z_n = 1$ iff $n \in I$. The one-hot constraint and the at-most-one constraint associated with the index set $I$ are

$$\zeta_I(z) := \begin{cases} 1, & \text{if } |z \wedge z_I| = 1 \quad \text{and} \quad (4) \\ 0, & \text{otherwise} \end{cases}$$

$$\eta_I(z) := \begin{cases} 1, & \text{if } |z \wedge z_I| \leq 1 \quad \text{and} \quad (5) \\ 0, & \text{otherwise} \end{cases}$$

2. Free Job-Shop Scheduling

We now formally introduce the FJSP, depending on the three parameters

- $M$: number of machines,  
- $T$: number of time slots per machine, and  
- $J$: number of jobs.

The $J$ jobs are to be distributed to the $MT$ available positions so that

$J$: every job gets performed precisely once and  
$P$: no position is filled with more than one job.
Note that FJSP\((T, 1, T)\) has the exact same structure as the \(T\)-city TSP.

In order to bring FJSP \(\equiv\) FJSP\((J, M, T)\) into the form (1), we introduce \(N = MTJ\) bits, identify \([N] \cong [M] \times [T] \times [J]\), and set

\[
z_{mtj} := \begin{cases} 1, & \text{if job } j \text{ runs on machine } m \text{ at time } t, \\ 0, & \text{otherwise.} \end{cases}
\]

The job assignment constraints \(J\) and the position assignment constraints \(P\) then read

\[
J : \sum_{m=1}^{M} \sum_{t=1}^{T} z_{mtj} = 1, \quad j \in [J] \quad \text{and} \quad (7)
\]

\[
P : \sum_{j=1}^{J} z_{mtj} \leq 1, \quad (m, t) \in [M] \times [T]. \quad (8)
\]

Thus \(J\) and \(P\) are one-hot and at-most-one constraints, respectively. Namely, we have

\[
J : c_{j} := \zeta_{\Delta(j)}, \quad j \in [J] \quad \text{and} \quad (9)
\]

\[
P : c_{m,t} := \eta_{\Delta(m,t)}, \quad (m, t) \in [M] \times [T] \quad \text{and} \quad (10)
\]

with job blocks \(\Delta(j) := [M] \times [T] \times \{j\}\) and position blocks \(\Delta(m,t) := \{m\} \times \{t\} \times [J]\). The constraints are equivalently captured in the coordinate relation

\[
(m, t, j) \sim (m', t', j') :\iff (m = m' \land t = t') \lor j = j'. \quad (11)
\]

Thus, if \(z\) is a solution to FJSP then \((m, t, j) \sim (m', t', j')\) implies that at least one of the bits \(z_{mtj}\) and \(z_{mt'j'}\) is 0. Since we are mainly interested in the constraints, we leave the objective function \(f\) unspecified.

Lastly, the FJSP solution set (expressed as a coordinate set) is explicitly given by

\[
\bigcup \{(m_{j}, t_{j}) : j \in [J]\} : (m_{1}, t_{1}) \neq \cdots \neq (m_{J}, t_{J}) \in [M] \times [T] \bigg\}. \quad (12)
\]

3. Constraint Graph Model

The solution set of a COP \(C = (N, f, \{c_{a}\})\) can be further studied by introducing the so-called constraint graph\(15,16\). First we identify \([N]\) with a vertex set \(V = \{v_{1}, \ldots, v_{N}\}\). Moreover, the bit string \(z\) associated to the subset \(I \subseteq [N]\) is identified with the subset of vertices \(V_{I} := \{v_{n} : n \in I\} \subseteq V\).

The constraints enter as edges of the graph: Two vertices \(v_{m}\) and \(v_{n}\) are joint by an edge iff there is a constraint that prohibits the two bits \(z_{m}\) and \(z_{n}\) from taking the value 1 at the same time. Since this construction is symmetric, the constraint graph \(G = (V, E)\) is undirected. Thus, the edges are unordered pairs of vertices and we may address an edge \(e \in E\) with its end points. Assuming that we have defined a coordinate relation \(~\) similar to (11), then we can express the set of edges simply as

\[
E = \{(v_{m}, v_{n}) : m \sim n\}. \quad (13)
\]

For one-hot and at-most-one constraints we can be more concrete: If \(\zeta_{j} \in \{c_{a}\}\) or \(\eta_{j} \in \{c_{a}\}\) all vertices \(v_{n}\) with \(n \in I\) are mutually connected, i.e., they form a clique in the constraint graph. Figure 1 shows the constraint graph of an FJSP\((3, 2, 2)\) instance.

A solution to \(C\) does not violate any constraint. Thus it corresponds to an independent set (or coclique) in the associated constraint graph \(G(C)\). If \(C\) incorporates \(J\) one-hot constraints and some additional at-most-one constraints (such as the FJSP), every solution has Hamming weight \(J\). Accordingly, we call any vertex subset \(W \subseteq V\) a solution to \(C\) iff

(i) \(|W| = J\) and

(ii) \(W\) is an independent set.

Moreover, we say that a permutation \(\rho : V \rightarrow V\) preserves feasibility iff \(\rho(W)\) is again a solution whenever \(W \subseteq V\) is a solution. We denote with \(F\) the set of all feasibility-preserving permutations. As the composition of two feasibility-preserving permutations preserves feasibility again and the identity also preserves feasibility, one readily deduces that \(F\) is a subgroup of \(\text{Sym}(V)\). Via the identification of bit strings with vertex subsets, \(F\) can also be interpreted as acting on the solution set \(S(C)\), i.e., on a set of bit strings. However, the notions of solutions and of \(F\) can also be considered independently of any underlying COP.

We remark that we could alternatively have declared the complement graph of \(G(C)\) as the actual constraint graph. Then, solutions would correspond to cliques instead of cocliques, but the underlying structure would not change since a graph and its complement have the same automorphism group (see Section IV). This equivalent point of view is noteworthy since many results in graph theory are formulated in terms of cliques. However, we proceed with our original definition.

4. Encoding on Quantum Computers

Consider a COP \(C = (N, f, \{c_{a}\})\). The standard encoding procedure identifies each bit string \(z\) with a computational basis state \(|z\rangle\) of the \(N\)-qubit space \(\mathcal{H} := \mathbb{C}^{2^{N}}\).

This induces a representation of functions over \(Z(N)\) as linear operators on \(\mathcal{H}\). Namely, the classical objective function \(f\) is mapped to an objective Hamiltonian, diagonal in the computational basis

\[
f \mapsto C := \sum_{z \in Z(N)} f(z) |z\rangle \langle z|. \quad (14)
\]
III. FEASIBILITY STRUCTURE OF FREE JOB-SHOP SCHEDULING

In this chapter we investigate the constraint graph of an FJSP instance. Our focus lies on determining the feasibility-preserving subgroup $F$ and its properties. For brevity, we will not distinguish between the COP and its constraint graph anymore.

1. Feasibility-Preserving Graph Automorphisms

Let $G = (V, E)$ be a graph. Recall that a graph automorphism is a bijection $\varphi : V \rightarrow V$ such that $\varphi$ as well as $\varphi^{-1}$ preserve adjacency. One can actually prove that any bijective graph homomorphism between $G$ and itself is already a graph automorphism. We start with the general observation that graph automorphisms are always feasibility-preserving.

**Proposition 1.** Let $G = (V, E)$ be a graph with solution set $S$. It holds that $\text{Aut}(G) \subseteq F$.

**Proof.** Let $\varphi \in \text{Aut}(G)$ and let $W \subseteq S$ be arbitrary. Since $\varphi : V \rightarrow V$ is bijective, it holds that $|\varphi(W)| = |W| = J$. Let $\varphi(v), \varphi(w) \in \varphi(W)$. Since $W$ is an independent set, it holds that $vw \notin E$. With the isomorphism property of $\varphi^{-1}$, it follows that $\varphi(v)\varphi(w) \notin E$; hence $\varphi(W)$ is again an independent set. This shows that $\text{Aut}(G) \subseteq F$. □

For general graphs $G$, $F$ will be a strict superset of $\text{Aut}(G)$. In case of the FJSP graph, however, we have equality. In order to prove this we utilize a simple auxiliary result.

**Proposition 2.** Let $G = (V, E)$ be a graph with solution set $S$. If for all non-adjacent $v, w \in V$ there exists $W \subseteq S$ so that $v, w \in W$, then $\text{Aut}(G) = F$.

**Proof.** By Proposition 1 it suffices to show that $F \subseteq \text{Aut}(G)$. Let $\rho \in F$ and $v, w \in V$ with $\rho(v)\rho(w) \notin E$. Then there exists $W \subseteq S$ so that $\rho(v), \rho(w) \in W$. Since $F$ is a group, $\rho^{-1}$ is also feasibility-preserving, hence $v, w \in \rho^{-1}(W) \subseteq S$ are non-adjacent. Thus the bijective map $\rho$ fulfills $(\rho(v)\rho(w) \notin E \implies vw \notin E)$ which is an equivalent characterization of a graph automorphism. □

Consider again the FJSP solution set (12). Note that any pair $(m, t, j), (m', t', j')$ corresponding to non-adjacent vertices, i.e., $(m, t) \neq (m', t')$ and $j \neq j'$, can be augmented to form a solution to FJSP. Therefore, we can apply Proposition 2 and conclude that $\text{Aut}(\text{FJSP}) = F$. 

---

**FIG. 1: FJSP(2, 2, 3) constraint graph. In the above graph, the vertices are labeled using the coordinate system $(m, t, j)$. In addition, the three job blocks $\Delta_{(j)}$, $j \in \{3\}$, are drawn in.**
2. Determining \( F \)

Since the relation (11) is divided in two logically disjoint parts, one concludes that the constraint graph is given by the cartesian product\(^{17}\) of the two graphs \( K^J \) and \( K^P \) for \( P := MT \), where \( K^n \) is the complete graph with \( n \) vertices. With some effort one can show the following theorem.\(^{18}\)

**Theorem 3.** Let \( G \) be a graph and \( H_1, \ldots, H_k \) be representatives of the isomorphism types of indecomposable components\(^{17}\) of \( G \).

Let \( m_i \) be the number of indecomposable components of \( G \) which are isomorphic to \( H_i \). Then it holds

\[
\text{Aut}(G) \cong \prod_{i=1}^k \left( [\text{Aut}(H_i) \times \cdots \times \text{Aut}(H_i)] \rtimes S_{m_i} \right),
\]

where \( \rtimes \) denotes the wreath product.\(^{19}\)

In fact we obtain for \( P = J \) the wreath product\(^{19}\). We can interpret the factors in the direct product (17) of \( \text{Aut}(FJSP) \) as permutations of the position coordinates, and of the job coordinate, respectively. For example consider the element \( g := (\text{id}, \sigma) \in S_P \times S_J \). Then \( g \) acts like

\[
9(m, t, j) = (\sigma(m, t), j)
\]

whereby \( \sigma \) only permutes the machine number and the time slot. In the busy case, i.e., \( P = J \), which is especially fulfilled for the TSP, there is therefore an additional non-trivial automorphism which interchanges between position and job blocks.

3. Blocks

From the relation (11) we further conclude that each of the corresponding blocks is a complete subgraph of the constraint graph. Acting with \( S_P \) (resp. \( S_J \)) stabilizes exactly one of these structures and the opposite group permutes them. For simplicity we consider the busy case, but neglect the non-trivial reflection between positions and jobs. Thereby, we consider only the subgroup

\[
G := S_J \times S_J \leq (S_J \times S_J) \rtimes S_2.
\]

Now one defines blocks in a group theoretical language for a transitive group action\(^{20}\) via a partition of the vertex set into disjoint subsets with the following property.

\[
\Delta \subset V : \quad 1 < |\Delta| < |V|
\]

is a block of \( G \) if

\[
\forall g \in G : \ 9\Delta = \Delta \lor 9\Delta \cap \Delta = \emptyset.
\]

One can readily verify that \( G \) acts transitively on \([M] \times [T] \times [J] \) and we can therefore conclude two analog block structures of this action of \( G \) on \([M] \times [T] \times [J] \). Furthermore, all vertices in a block are adjacent, which implies that a solution is a subset \( s \subset [M] \times [T] \times [J] \) such that each element in \( s \) belongs to exactly one block in each of the two partitions into position blocks and job blocks, respectively. Observe that in the non-busy case more position blocks exist then job blocks. In this situation a solution occupies only a subset of all position blocks.

4. Transitivity

The observation that solutions strongly correspond to blocks in a group-theoretical manner is the key observation for transitivity on the set of solutions. Since any element of a solution exactly corresponds to one position and to one job block, we can capture the action of \( G \) on the solution set \( S \) equivalently as the action of \( G \) on the blocks. Here, \([20], Proposition 1.37\) states that if the blocks are maximal with respect to inclusion (which they are here), then the block-wise action of \( G \) is primitive, i.e., it does not possess blocks on its own. In the busy case, we can conclude that there are exactly \( J \) position or equivalently job blocks and the action of \( S_J \) on the blocks is transitive. Moreover it is the normal action of \( S_J \) on \( J \) elements that is sharp transitive, which means that for every two blocks there is exactly one element in \( S_J \) which transfers between them. The solutions are now...
The unitary phase separator is supposed to render the classical objective function’s behavior but is technically merely required to be diagonal in the computational basis. We want to be more precise and suggest the following definition.

**Definition 5.** A Hamiltonian $H$ is called a phase separator Hamiltonian iff it fulfills the following two conditions:

(i) $H$ is diagonal in the computational basis.

(ii) The eigenspace of $H|_S$ corresponding to its smallest eigenvalue is $S_{\min}$.

Then

$$U_p(H, \gamma) := e^{-i\gamma H}$$

is the corresponding (parametrized) phase separator.

The canonical choice for a phase separator Hamiltonian is the objective Hamiltonian $C$. However, there might be decent approximations of $C$ which are easier to implement and still preserve the optimal solution space. Since the phase separator is itself diagonal in the computational basis it trivially leaves the solution space $S$ invariant!

**Mixer**

The unitary mixer is supposed to ‘preserve’ and ‘explore’ the solution space $S$. Preservation of $S$ simply means that $U_M(\beta)(S) \subseteq S$ should hold for all $\beta \in \mathbb{R}$. The exploring condition is defined as follows: For all $z, z' \in S$ there should exist a power $r \in \mathbb{N}$ and a parameter value $\beta \in \mathbb{R}$ so that $(z|U_M^{r\beta})(z') \neq 0$.

Our aim for a refined definition is now to mimic the two properties of the original QAOA mixer Hamiltonian

$$B = \sum_{n=1}^{N} \sigma_z^{(n)}$$

for unconstrained problems, but tailored to the constrained case. $B$, considered as a matrix in the computational basis, is component-wise non-negative and irreducible. Irreducibility means that the matrix $B$ does not leave any non-trivial coordinate subspace invariant. That is, the only two subspaces of $H$ which are a linear span of computational basis states and are left invariant under $B$ are $\{0\}$ and $H$. For brevity, we will address every linear span of computational basis states as a coordinate subspace. Our crucial observation is that the concept of irreducibility is indeed a fundamental mixing property which should be preserved in the constrained case.

In order to establish also ‘sequential’ mixers, we utilize the following result which can be proved by considering each Hamiltonian as the adjacency matrix of a graph.
Proposition 6. Let \( \{ H_i \}_{i \in I} \subset \mathcal{L}(\mathcal{H}) \), \( 0 < |I| < \infty \), be a family of Hamiltonians, component-wise non-negative in the computational basis, such that \( H_i(S) \subseteq S \) holds for all \( i \in I \). Then the following two statements are equivalent:

(i) Any coordinate subspace \( X \subseteq S \) that is left invariant under every \( H_i|_S \), is trivial.

(ii) \( \{ \sum_{i \in I} H_i \}_{|S} \in \mathcal{L}(S) \) is irreducible in the computational basis.

Definition 7. A family of Hamiltonians \( \mathcal{H} = \{ H_i \}_{i \in I} \subset \mathcal{L}(\mathcal{H}) \) fulfilling the conditions in Proposition 6 is called a mixing family. The corresponding (parametrized) simultaneous mixer is defined as

\[
U_{M,0}(\mathcal{H}, \beta) := e^{-i\beta \sum_{i \in I} H_i}.
\]  

Specifying a permutation \( \sigma \in S(I) \), the corresponding (parametrized) sequential mixer is defined as

\[
U_{M,\sigma}(\mathcal{H}, \beta) := \prod_{i \in I} e^{-i\beta H_{\sigma(i)}}.
\]  

We present here a general method for scheduling-type problems for obtaining suitable mixers from the feasibility-preserving subgroup \( F \). Following the permutation representation, we identify each \( g \in F \) via its action on \( Z(N) \) with a linear operator \( \rho(g) \in \mathcal{L}(\mathcal{H}) \), and via its restricted action on \( S(C) \) with a linear operator \( \rho_S(g) \in \mathcal{L}(S) \). Both representations yield permutation matrices in the computational basis. In the same way we have identified \( S(C) \) with \( S \), we may also identify every subset of solutions with coordinate subspaces of \( S \). Then it readily follows that \( F \) acts transitively on \( S \) iff the only coordinate subspaces of \( S \) that are left invariant by every \( \rho_S(g) \), \( g \in F \), are \( \{0\} \) and \( S \).

As we have shown in Section III, the action of \( F \) on \( S(F\text{JSP}) \) is indeed transitive. In addition, \( F \) consists of bit (value) permutations. Thus, the operator analogs \( \rho(F) \) respect the tensor product structure of the qubit space \( \mathcal{H} \), namely

\[
\rho(g) = \bigotimes_{n=1}^{N} |\psi_{g(n)}\rangle = \bigotimes_{n=1}^{N} |\psi(g(n))\rangle.
\]  

Starting from the unitary operators \( \rho(F) = \{ W_g \}_{g \in F} \), we construct a family of Hamiltonians by taking suitable matrix logarithms \( \{ iL(W_g) \}_{g \in F} \). A direct calculation yields that \( A \subseteq \mathcal{H} \) is a \( W \)-invariant subspace iff \( A \) is an \( L(W) \)-invariant subspace. Thus, the constructed family \( \{ iL(W_g) \}_{g \in F} \) admits the just introduced mixing property and can therefore be used to build simultaneous and sequential mixers.

2. A Quantum Group Optimization Algorithm

We propose here a VQA tailored to FJSP-instances. For brevity, we consider the busy case, i.e., we assume that \( J = P \), thus conceptually considering the TSP. Namely, given a complete graph \( G = (V, E) \) (not the constraint graph!) with \( |V| = J \) vertices, we consider the quadratic objective function

\[
f : Z(J^2) \rightarrow \mathbb{R}; \quad z \mapsto \sum_{\{u,v\} \in E} d_{uv}(z_{u,j}z_{v,j+1} + z_{v,j}z_{u,j+1}),
\]  

where \( d_{uv} \) is the distance between city \( u \) and \( v \). \( z_{u,j} = 1 \) means that city \( v \) is visited at time \( j^{-2} \). Our method above gives now a way to describe the whole solution set \( S \) of this problem in this representation with bit-to-qubit mapping. Namely, the set of all solutions is identified with the symmetric group \( \text{Sym}([J]) \). Fixing one solution \( s \in S \), we can consider the problem of optimizing \( f \) equivalently as optimizing

\[
\tilde{f} : \text{Sym}([J]) \rightarrow \mathbb{R}; \quad g \mapsto f(\sigma g).
\]  

Thus the TSP and, more generally, any generic FJSP-instances are, in fact, optimization problems over symmetric groups. This underlying group structure can be exploited in the following way: Consider an arbitrary \( \sigma \in \text{Sym}([J]) \). There is a well-known representation as product of at most \( \frac{(J-1)}{2} \) transpositions\(^2\). We can further write every transposition as a product of some of the \( J - 1 \) specific transpositions \( \tau_1 = (1, 2), \ldots, \tau_{J-1} = (J-1, J) \) in \( \text{Sym}([J]) \). In summary, we find for \( \sigma \in S(J) \) a binary vector \( \overline{r} \in \{0, 1\}^{\frac{(J-1)}{2} J^2} \) such that

\[
\sigma = \prod_{k=1}^{\frac{(J-1)}{2} J} (\tau_{k_1}^{r_{k_1}} \cdots \tau_{k_{J-1}}^{r_{k_{J-1}}}).
\]  

The representation of the transpositions \( \tau_{j,j+1} \) as elements in \( \mathcal{L}(\mathcal{H}) \) yields sums of disjoint SWAP gates since we have to mirror any transposition on a given block in every other block, that is,

\[
\tau_j = \frac{J}{2} \sum_{j=1}^{J} \text{SWAP}_{(j,i,i+1)} =: B_i.
\]

The summation over \( j \) thus runs through all blocks. Since all SWAP gates are also hermitian, the operator \( B_i \) is again hermitian. Introducing \( \frac{(J-1) J}{2} \) parameters and exponentiating each transposition operator \( B_i \) then yields the following implementation of \( \sigma \) as a parameterized quantum circuit:

\[
U(\vec{\beta}) = \prod_{k=1}^{\frac{(J-1) J}{2}} e^{i \beta_{k_1} B_1 \cdots e^{i \beta_{k_{J-1}} B_{J-1}}}. 
\]  

Due to the fact that \( e^{i \gamma \text{SWAP}} \) is equal to SWAP with a global phase \( i \) for \( \gamma = \frac{\pi}{2} \) and 1 for \( \gamma = 0 \), we can
directly transfer Theorem 4 to the gate implementation of permutations in $\text{Sym}(\mathcal{J})$ to conclude that every solution is achievable with a circuit $U(\vec{\beta})$ for appropriate $\vec{\beta} \in [0, \frac{\pi}{2}]^{(\mathcal{J} - 1)^2/2}$. Therefore the number of gates (not decomposed) is within $O(J^3)$.

The resulting VQA is now

1. Start in a solution state $|\iota\rangle \in \mathcal{S}$.
2. Apply the parameterized quantum circuit (35).
3. Variationally optimize $\vec{\beta} \in [0, \frac{\pi}{2}]^{(\mathcal{J} - 1)^2/2}$ using a classical optimization rule.
4. Measure the final outcome state in the computational basis.

V. NUMERICAL RESULTS

We demonstrate our just introduced VQA on an FJSP(2,2,4)-instance. Figure 3 displays its constraint graph.

1. Automorphisms, Group Action, and Blocks

Due to Theorem 3, the group of feasibility-preserving permutations is given by

$$ F = \text{Aut}(\text{FJSP}(2,2,4)) \cong (S_4 \times S_4) \rtimes S_2. $$ (30)

The first copy of $S_4$ is identified with $\text{Sym}([M] \times [T])$, while the second one corresponds to $\text{Sym}([J])$. We discard the non-trivial group automorphism in the semi-direct product in order to preserve the block structure.

After assigning a bit to each of the 16 elements in $[2] \times [2] \times [4]$ via

$$(m, t, j) \mapsto 4(m - 1) + 2(t - 1) + j, \quad (31)$$

(compare Figure 3 and Figure 4), we can explicitly generate $\text{Sym}([M] \times [T])$ with three permutations:

$$ \text{Sym}([M] \times [T]) = \langle (1, 2)(5, 6)(9, 10)(13, 14), (2, 3)(6, 7)(10, 11)(14, 15), (3, 4)(7, 8)(11, 12)(15, 16) \rangle. $$ (32)

Consequently, we have to implement three distinct mixer Hamiltonians $B_1, B_2, B_3$. As a concrete example, consider the permutation

$$(1, 2)(5, 6)(9, 10)(13, 14)$$

which corresponds to the Hamiltonian

$$ B_1 = \sum_{m, t=1}^{2} \text{SWAP}^{(m, t)}(1, 2) + \text{SWAP}(5, 6) + \text{SWAP}(9, 10) + \text{SWAP}(13, 14). $$

2. Cost Function

Using again the enumeration (31), we consider the following objective function

$$ f: \{0, 1\}^{16} \to \mathbb{R}, \quad z \mapsto \sum_{m, t, j} \omega_{mtj}z_{mtj}. \quad (33) $$
TABLE I: Every solution bit strings to FJSP(2,2,4) is listed with its objective value under the objective function (33) with weights (34).

| bit string | value |
|------------|-------|
| 0010000110000100 | 5 |
| 0010000101001000 | 5 |
| 0010010000000100 | 7 |
| 0100001100100100 | 7 |
| 0010100001010100 | 7 |
| 0100010000110000 | 8 |
| 0100001100000010 | 8 |
| 0010100001000001 | 8 |
| 0010100010000001 | 8 |
| 1000010101000000 | 9 |
| 0010100101000010 | 9 |
| 0010100010000010 | 9 |
| 0010010001000000 | 9 |
| 0100010000010100 | 10 |
| 0100100010000100 | 10 |
| 0010100001100000 | 10 |
| 0001001001001000 | 10 |
| 0001100000100100 | 10 |
| 1000000101000010 | 11 |
| 1000010000100000 | 11 |
| 0001100001000010 | 11 |
| 0001010010000001 | 11 |
| 0001100010000010 | 11 |

We collect the weights in two machine-specific weight matrices

\((\omega)_{1tj} = \begin{pmatrix} 3 & 2 & 2 & 3 \\ 2 & 2 & 3 & 0 \end{pmatrix}, \ (\omega)_{2tj} = \begin{pmatrix} 2 & 2 & 4 & 2 \\ 1 & 1 & 4 & 2 \end{pmatrix}\) (34)

Table I displays the objective values for all feasible solutions of FJSP(2,2,4).

3. Implementation and Results

In addition to the just constructed mixers, we also implement the QAOA-phase separator. Mixer and phase separator together result in the following parametrized quantum circuit

\[ U(\beta, \gamma) = e^{i\beta_1 B_1} e^{i\beta_2 B_2} e^{i\beta_3 B_3} e^{i\gamma_1 C}, \]

\[ e^{i\beta_4 B_1} e^{i\beta_5 B_2} e^{i\beta_6 B_3} e^{i\gamma_2 C}, \ldots e^{i\beta_6 B_1} e^{i\beta_7 B_2} e^{i\beta_8 B_3} e^{i\gamma_6 C}. \]

Choosing an initial state \(|z_0\rangle\), the variational optimization task now reads

\[ \min\{\langle z_0|U(\beta, \gamma)^*CU(\beta, \gamma)|z_0\rangle : \beta \in [0, \pi/2]^{18}, \gamma \in \mathbb{R}^6 \}. \] (36)

As a concrete feasible initial state, we choose

\[ |z_0\rangle = |1000010000100001\rangle. \]

The whole quantum circuit is implemented in Qiskit. For the classical parameter adaptation we use COBYLA as a global optimization routine. Figure 5 and Table II display the numerical results.

We observe a dominating sampling of one of the optimal solution states (791 out of 1024 counts), indicating a fast convergence behavior. Bit strings which have a small Hamming distance to it are also sampled, but with a significantly smaller amplitude. This is due to the fact that the construction of the optimal computational basis state is merely approximate. Any deviation from it is reflected in the measurement statistics as the projective measurements then do not reliably give the correct bit values. These byproducts are mostly infeasible states, which is a direct consequence of the underlying feasibility structure: Distinct solution bit strings differ at, at least, four positions since reassigning one job to a position occupied by another job, one also has to reassign the latter. Obtaining distributions with some non-solution bit strings in hardcoded instances is, however, very common as the final measurement is insensitive to any feasibility structure.
VI. OUTLOOK AND CONCLUSION

In this paper we presented a general approach for characterizing the feasibility structure of job-shop scheduling problems. We utilized the constraint graph model as a general interplay between graph and group theory for determining symmetries for certain problem instances. This additional perspective allowed us to find feasibility-preserving mapping as graph automorphisms. For the free job-shop case, we calculated the whole group of such feasibility-preserving functions and proved that it equals the automorphism group of the constraint graph. This is a very strong statement which unfortunately is not transferable to all other types of job-shop scheduling problems. For example, for a generic flexible job-shop instance, where each job is subdivided into an ordered sequence of operations, we can similarly construct a constraint graph and study its automorphism group. However, in this case, due to additional edges which break certain symmetries, the automorphism group is generally way smaller. In many cases, there is no possible identification between group elements and solutions anymore. The remaining symmetries can still be incorporated into QAOA-mixers but have to be supplemented with additional elements that do not correspond to classical bit permutations.

We emphasize again that the FJSP is equivalent to optimizing over a particular group (compare (27)). We solved this type of problem by representing a symmetric group with exponentials of SWAP gates. The resulting algorithm has the advantage that we can estimate the number of gates in a polynomial regime in comparison of the problem size. The prior discussed transitive action of the group on the solution set guarantees reachability of all solutions. In principle, we can generalize this procedure to an arbitrary (finite) group $G$ acting transitively on a given set of solutions $S$. Then the permutation representation would still yield permutation operators on the qubit space which, however, will not correspond to actual qubit permutations. It would be interesting to characterize the obtained operators and the performance of our algorithm for more exotic cases. However, operators that do not respect the tensor product structure of $\mathcal{H}$ will be generally very difficult to implement.

ACKNOWLEDGMENTS

We thank Michael Cuntz, Tim Heine, Max Hess, Jan Rasmus Holst, Lauritz van Luijk, Tobias J. Osborne, Leonhard Richter, Marco Tomamichel and Timo Ziegler for helpful discussions. The authors thank and are deeply indebted to Benjamin Sambale for many long and fruitful discussions. GK acknowledges financial support by the DAAD and IIT Indore (Kapil Ahuja) for a guest stay. RS acknowledges financial support by the Quantum Valley Lower Saxony and by the BMBF project ATIQ.

1M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Nature Reviews Physics 3, 626 (2021).
2J. Marshall, F. Wudarski, S. Hadfield, and T. Hogg, IOP SciNotes 1, 025208 (2020).
3A. Peruzzo, J. R. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, Nature Communications 5, 4213 (2014).
4P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, Quantum 4, 256 (2020).
5J. Larkin, M. Jonsson, D. Justice, and G. G. Guerreschi, Quantum Science and Technology 7, 045014 (2022).
6J. Preskill, Quantum 2, 79 (2018).
7K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Reviews of Modern Physics 94 (2022).
8G. Kofmann, L. Binkowski, L. van Luijk, T. Ziegler, and R. Schwonnek, Deep-Circuit QAOA (2022), arXiv:2210.12406 [quant-ph].
9E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm (2014), arXiv:1411.4028 [quant-ph].
10J. S. Baker and S. K. Radha, Wasserstein Solution Quality and the Quantum Approximate Optimization Algorithm: A Portfolio Optimization Case Study (2022), arXiv:2202.06782 [quant-ph].
11P. D. de la Grand’rive and J.-F. Hullo, Knapsack Problem variants of QAOA for battery revenue optimisation (2019), arXiv:1908.02210.
12W. van Dam, K. Eldefrawy, N. Genise, and N. Parham, Quantum Optimization Heuristics with an Application to Knapsack Problems (2021), arxiv:2108.08805 [quant-ph].
13S. Hadfield, Z. Wang, B. O’Gorman, E. Rieffel, D. Venturelli, and R. Biswas, Algorithms 12, 34 (2019).
14F. Rossi, P. v. Beek, and T. Walsh, Handbook of Constraint Programming (Elsevier Science Inc., 2006).
15T. Leighton, Journal of Research of the National Bureau of Standards 84, 489 (1977).
E. C. Freuder and M. J. Quinn, in *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'85 (1985) p. 1076–1078.

W. Imrich and S. Klavžar, *Product graphs* (Wiley-Interscience, New York, 2000).

We owe the theorem and its proof to personal correspondence with Benjamin Sambale.

J. J. Roman, *An Introduction to the Theory of Groups* (Springer New York, NY, 1995).

B. Sambale, *Endliche Permutationsgruppen* (Springer Fachmedien Wiesbaden, 2017).

B. E. Sagan, *The Symmetric Group* (Springer New York, NY, 2001).

L. Binkowski, G. Koßmann, L. van Lujik, T. Ziegler, and R. Schwonnek, in preparation.

More precisely, in a first step, one can interpret the matrix representation of each Hamiltonian as the adjacency matrix of a weighted directed graph. However, since we are not interested in actual weights and all matrices are hermitian, it suffices to consider simply a graph. Furthermore, since all components are non-negative, no cancellation happens when summing up all the matrices.

Since we only have finitely many unitary matrices $W_g$, we can always find a common branch $L$ of the complex logarithm for the union of all their eigenvalues.

S. Hadfield, Quantum algorithms for scientific computing and approximate optimization (2018), arXiv:1805.05265 [quant-ph].

J. E. Humphreys, *Reflection Groups and Coxeter Group* (Cambridge University Press, 1992).