Proceedings of the 57th Hawaii International Conference on System Sciences | 2024

Graph-controlled Permutation Mixers in QAOA for the Flexible Job-Shop Problem

Lilly Palackal*
Infineon Technologies AG
Technical University of Munich
lilly.palackal@infineon.com

Leonhard Richter*
Friedrich-Alexander University
Erlangen-Nürnberg
leonhard.richter@outlook.de

Maximilian Hess
Infineon Technologies AG
Leibniz University Hannover
maximilian.hess@infineon.com

*Authors contributed equally

Abstract

One of the most promising attempts towards solving optimization problems with quantum computers in the noisy intermediate scale era of quantum computing are variational quantum algorithms. The Quantum Alternating Operator Ansatz provides an algorithmic framework for constrained, combinatorial optimization problems. As opposed to the better known standard QAOA protocol, the constraints of the optimization problem are built into the mixing layers of the ansatz circuit, thereby limiting the search to the much smaller Hilbert space of feasible solutions. In this work we develop mixing operators for a wide range of scheduling problems including the flexible job shop problem. These mixing operators are based on a special control scheme defined by a constraint graph model. After describing an explicit construction of those mixing operators, they are proven to be feasibility preserving, as well as exploring the feasible subspace.

Keywords: Quantum Computing, Quantum Algorithms, Constraint-Mixer QAOA, Job-Shop Scheduling

1. Introduction

The usage of quantum mechanical phenomena in computing provably enables asymptotic speedups over classical computers in some computational tasks (Shor, 1997, Grover, 1996), and researchers are aiming to discover such algorithmic breakthroughs in others. The reality of quantum computing right now, sometimes called the noisy intermediate scale quantum (NISQ) era, presents additional challenges: While quantum processors are only available with a modest number of noisy qubits, the algorithms under consideration must address these shortcomings. In the case of quantum optimization, hopes for near- to mid-term algorithmic advances rest on the shoulders of variational hybrid algorithms (Cerezo et al., 2021). All of them involve a parameterized quantum circuit $U(\theta)$ which is used to generate an ansatz state

$$|\psi(\theta)\rangle := U(\theta)\,|0\rangle\,. \tag{1.1}$$

Measuring this ansatz state several times in the computational basis provides solution candidates (the bit string representations of the computational basis states) to the (binary) optimization problem. Based on the quality of these samples with respect to the optimization problem, the parameters θ are then updated and fed back into the quantum circuit. In this algorithmic scheme, one qubit roughly corresponds to one binary variable in the optimization problem. Without the need for spatial overhead, such as auxiliary quantum registers, relevant problem sizes can be targeted much sooner, thereby adressing the “intermediate scale” concern. Due to the variational nature of the algorithm, where measurement outcomes inform the choice of parameters and thus the eventual solution, noise is taken into account and filtered out to a certain extent. This is in contrast to pure quantum algorithms, where each step “trusts” the preceding steps to prepare a certain quantum state, thus allowing errors to propagate easily.
The Quantum Alternating Operator Ansatz (Hadfield, 2018) also referred to as the constraint mixer Quantum Approximate Optimization Algorithm (CM-QAOA) (Fuchs et al., 2022) is a framework for such algorithms. Its setup is very similar to the better known Quantum Approximate Optimization Algorithm (Farhi et al., 2014), involving alternating mixing and phase separation layers. The main distinction point of CM-QAOA is to explore only feasible solutions for a given optimization

URI: https://hdl.handle.net/10125/107302
978-0-9981331-7-1

problem, thus drastically reducing the dimension of the search space. This is in contrast to many other variational algorithms like standard QAOA, where infeasible solutions are assigned undesirably high values via penalty terms and the mixing layers create a superposition of all bit strings. In this work, we introduce a concrete construction following this scheme, which is suitable for a possibly wide range of scheduling problems including the *Flexible Job-Shop Problem* (FJSP). To this end we consider a *constraint-graph model* that encodes the constraints of FJSP-instances in a graph, similarly to the work of Koßmann et al., 2022. We use the constraint graph to define a unitary mixing operator for the FJSP which is made up of controlled permutation operators. We prove that this mixing operator is both feasibility preserving and explores the feasible subspace.

In Section 2 we formally introduce the two main ingredients of this work, namely the algorithmic framework CM-QAOA in Section 2.1 and the flexible job shop problem in Section 2.2. Section 3 goes on to construct the constraint graph of the FJSP (Section 3.1). In Section 3.2 we go on to introduce the concept of controlled permutation unitaries and provide an explicit construction of a mixing operator for the FJSP. We prove that this operator is feasibility-preserving in Theorem 1 and that it explores the feasible subspace in Theorem 2. Finally, in Section 4, the results are compared to related work and the practical limitations of our approach are discussed.

2. Preliminaries

2.1. The Quantum Alternating Operator Ansatz

The CM-QAOA, originally proposed by Hadfield (2018), is a framework of variational quantum algorithms generalizing the Quantum Approximate Optimization Algorithm (Farhi et al., 2014). The CM-QAOA allows for a restriction of the search space to the subspace of feasible states, i.e. computational basis states whose bit string representations correspond to feasible solutions to an optimization problem. More precisely, the subspace of feasible solutions is spanned by computational basis states corresponding to bit strings which satisfy all constraints of the given problem instance. This potentially decreases the size of the state space in which the goal function is to be optimized substantially. To achieve this, the main effort in algorithm design is shifted from constructing a cost Hamiltonian representing the goal function as well as the constraints of a problem instance towards the implementation of the constraints directly into mixing operators. A combinatorial optimization problem which can be solved by CM-QAOA is given by a pair (F, f), where, for $N \in \mathbb{N}$, $f : \{0,1\}^N \to \mathbb{R}$ is a cost function to be minimized[1] and

$$F \subset \{0,1\}^N \tag{2.1}$$

is the set of feasible solutions typically characterized by a list of constraints. In the CM-QAOA the choice of alternating operators is more flexible compared to the QAOA. Especially, the mixing operators no longer need to be generated by a mixer Hamiltonian as in the work of Farhi et al., 2014, but is rather required to fulfill certain *design criteria* Hadfield, 2018. The substitute for the QAOA-mixer $\sum_i X_i$ must have two features which ensure that the search of CM-QAOA remains within the feasible subspace $\mathcal{F}$ and at the same time every feasible state can be reached for a suitable choice of parameters.

Definition 1 (Design criteria for mixing operators)**.** Let the tuple $(F \subset \{0,1\}^N, f)$ define a constrained optimization problem and let $B_\beta = \{B(\beta) \mid \beta \in \mathbb{R}\}$ be a family of unitaries acting on the Hilbert space $\mathcal{H} := (\mathbb{C}^2)^{\otimes N}$. Denote the subspace of feasible solutions by $\mathcal{F} := \operatorname{span}(\{|x\rangle \, | x \in F\}) \subset \mathcal{H}$. The family B_β is said to ...

... *preserve feasibility*, if and only if

$$\forall \beta \in \mathbb{R}:\; B(\mathcal{F}) := \{B(\beta)\,|\psi\rangle \mid |\psi\rangle \in \mathcal{F}\} \subseteq \mathcal{F}.$$

... *explore the feasible subspace*, if and only if for all $x, y \in F$: there exists a $\beta^* \in \mathbb{R}$, and $r \in \mathbb{N}$ s.t.

$$|\langle y|\, B^r(\beta^*)\, |x\rangle| > 0.$$

Definition 2 (CM-QAOA)**.** A CM-QAOA instance is given by two families of unitary operators, $B_\beta = \{B(\beta) \mid \beta \in \mathbb{R}\}$ and $C_\gamma = \left\{e^{-i\gamma H_f} \,\middle|\, \gamma \in \mathbb{R}\right\}$, and a feasible initial state $|\psi_0\rangle \in \mathcal{F}$, such that each operator in B_β fulfills the criteria in Definition 1. Here $H_f = \sum_{x\in\{0,1\}^N} f(x)\,|x\rangle\langle x|$ is the Hamiltonian representing the function f diagonally.

The operator $B(\beta)$ is referred to as the *mixing operator* and the operator $e^{-i\gamma H_f}$ is called *phase-separation operator*.

2.2. The Flexible Job-Shop Problem

An example of a combinatorial optimization problem, which is relevant to industrial applications, is the *Flexible Job-Shop Problem* (FJSP). A number of jobs, each of which is given by a sequence of operations of varying processing times, have to be scheduled on a number of machines with the goal of optimizing a performance

[1] As maximization of f is the same as minimization of $-f$, it is sufficient to only consider minimization problems.

indicator, e.g. minimizing the overall execution time, often called *makespan*. More concretely, an instance of FJSP comprises the following data:

- n_J jobs $J_i \in J = \{J_1, \ldots, J_{n_J}\}$, for some $n_J \in \mathbb{N}$,
- For each $j \in [n_J]$, a set of p_j operations $O_{j,o}$ in $O_j = \{O_{j,1}, \ldots, O_{j,p_j}\}$, for some $p_j \in \mathbb{N}$
- n_M machines M_m in $M = \{M_1, \ldots, M_{n_M}\}$, for some $n_M \in \mathbb{N}$,
- time-slots T_t in $T = \{T_1, \ldots, T_{n_T}\}$, for some number of time steps $n_T \in \mathbb{N}$.

Each job J_j consists of the operations O_j, that shall be assigned to a machine M_m in M at a given time-slot $T_t \in T$. Denoting the set of all operations by $O = \bigcup_{j,o} O_{j,o}$, an *assignment* of operation $O_{j,o}$ is defined to be a tuple $(O_{j,o}, M_m, T_t) \in O \times M \times T$. A subset $S \subseteq O \times M \times T$ of assignments, is called a *schedule*. The following additional data are needed to define feasible schedules.

- For each $O_{j,o} \in O$ a set $M_{j,o} \subseteq M$ of machines the operation may be executed on.
- For each $O_{j,o} \in O$ and $M_m \in M$, a duration $d_{j,o,m} \in N$ the operation $O_{j,o}$ takes, when being executed on machine M_m. If $M_m \notin M_{j,o}$ the corresponding duration does not matter and is set to infinity $d_{j,o,m} = \infty$.

A schedule is called feasible if it satisfies the following constraints:

1. **Assignment constraint** For every operation $O_{j,o} \in O$, there is exactly one time-slot $T_t \in T$ and exactly one machine $M_m \in M$, such that $(O_{j,o}, M_m, T_t) \in S$. In other words, $(O_{j,o}, M_m, T_t) \in S \Rightarrow (O_{j,o}, M_{m'}, T_{t'}) \notin S$ if $m \neq m'$ and $t \neq t'$.
2. **Order constraint** For all assignments $(O_{j,o}, M_m, T_t)$, $(O_{j,o'}, M_{m'}, T_{t'}) \in S$ with $o < o'$, operation $O_{j,o'}$ is started only after operation $O_{j,o}$ has been finished, i.e. $t + d_{j,o,m} \leq t'$.
3. **Machine constraint** For every machine $M_m \in M$, operations $O_{j,o} \neq O_{j',o'} \in O$ and $T_t, T_{t'} \in T$ with $(O_{j,o}, M_m, T_t), (O_{j',o'}, M_m, T_{t'}) \in S$ we have $t' \notin [t, t+d_{j,o,m})$. That is, at no point in time two operations are being processed simultaneously on the same machine.

A schedule is called *feasible* if each of the constraints is satisfied. The number of time steps n_T must be chosen such that it is guaranteed that a feasible schedule finishing within n_T steps exists. This is usually achieved by constructing a feasible schedule via some computationally cheap heuristic method. The aim of the FJSP is to find a feasible schedule, which optimizes a cost function. Cost functions for the FJSP can be defined in various fashions (Xie et al., 2019). One typical choice which will be our choice in this work is the makespan $C(S) := \max\{T_t + d_{j,o,m} \mid (O_{j,o}, M_m, T_t) \in S\}$, i.e. the total processing time.

We formulate the FJSP as a binary combinatorial optimization in the sense of eq. (2.1) as follows. Every possible schedule $S \subseteq O \times M \times T$ (feasible or not) can be encoded by some binary string $x = x_1 \ldots x_N \in \{0,1\}^N$, where $N := |O \times M \times T|$. Each x_i denotes whether a given assignment is part of the schedule S. More precisely for a given schedule S, set

$$x_i := \begin{cases} 1, & \iota^{-1}(i) \in S \\ 0, & \text{else}, \end{cases} \tag{2.2}$$

where $\iota\colon\ O \times M \times T \to \{1, \ldots, N\}$ is a bijective map, enumerating all possible assignments. Thus ι induces a bijection $I\colon\ \mathcal{P}(O \times M \times T) \to \{0,1\}^N$. The set $F \subseteq \{0,1\}^N$ of feasible binary strings is then the set of binary strings $x \in \{0,1\}^N$, such that $S(x) := I^{-1}(x)$ is a feasible schedule. Together with the cost function $f(x) := C(\iota^{-1}(x))$, (F, f) is a binary combinatorial optimization problem as above. In particular, the bit $x_{\iota(O_o, M_m, T_t)}$ is set to 1 if we assign operation o to machine m at time t.

3. Graph-based Constraint Mixer Operators

Many scheduling problems may be formulated as some variation of a graph-coloring problem (Marx, 2004). The common idea in all such approaches is to consider a graph, where each vertex corresponds to one object (job, assignment of a job to a machine,...), and the set of edges is constructed such that "conflicting" objects share an edge. Roughly speaking, the solution of the scheduling problem then becomes equivalent to finding some coloring of the corresponding graph (Marx, 2004). In many cases, each color represents a time slot or machine for the object to be assigned to. However, this approach is hard to realize for the FJSP, because the available assignments for each job depend on the assignments of all the operations. Nevertheless, the FJSP can be translated to a constraint graph: In contrast to other scheduling problems, not each job but each possible assignment (j, o, m, t) corresponds to a vertex of the graph. Then each pair of vertices, whose corresponding assignments are in conflict with each other, are connected by an edge. The resulting graph $G = (V, E)$ now encodes all the constraints of a given FJSP instance. Further, each feasible schedule corresponds to a selection $S \subseteq V$ of $|S| = k$ vertices, such that no pair of vertices in S share an edge, i.e $(j, k) \in S^2 \Rightarrow (j, k) \notin E$.

3.1. The Constraint Graph of FJSP

Consider a FJSP instance, characterized by the data listed in Section 2.2. Denote the set of assignments by $\mathcal{A} := O \times M \times T$. Note that the constraints 1, 2 and 3 are all formulated in terms of pairs of assignments. We denote these sets of conflicting pairs with respect to assignment, order and machine constraints by

$$\Omega^{(A)} = \{a, a' \in \mathcal{A} \mid a, a' \text{ are in conflict with } 1\}, \tag{3.1}$$

$$\Omega^{(O)} = \{a, a' \in \mathcal{A} \mid a, a' \text{ are in conflict with } 2\}, \tag{3.2}$$

$$\Omega^{(M)} = \{a, a' \in \mathcal{A} \mid a, a' \text{ are in conflict with } 3\}. \tag{3.3}$$

Then the construction of the constraint graph described above results in $G = (V, E)$ with

$$V = \left\{\iota\big(O_{j,o}, M_m, T_t\big) \in [N] \;\middle|\; M_m \in M_{j,o}\right\}, \tag{3.4}$$

$$E = \left\{(j, k) \;\middle|\; \left\{\iota^{-1}(j), \iota^{-1}(k)\right\} \in \Omega^{(\text{total})}\right\}, \tag{3.5}$$

where $\Omega^{(\text{total})} := \Omega^{(A)} \cup \Omega^{(O)} \cup \Omega^{(M)}$ and ι is the bijective enumeration of the set of all schedules. The number k of vertices to be colored is the total number of operations $k = |O|$. Figure 1 shows the constraint graph for a small instance of the FJSP together with a feasible solution.

3.2. Graph-controlled Permutation Mixers

In the following, we construct mixing operators for the CM-QAOA for the flexible job shop problem making use of the constraint graph formalism. More concretely, an N-bit string $x \in \{0,1\}^N$ encoding a schedule S is equivalent to marking $|x|$ vertices of the N-vertex constraint graph, where $|x| = \sum_j x_j$ is the *Hamming weight* of x. The validity of the bit string x corresponds to the property that

- there are no adjacent marked vertices in G:

$$(j, k) \in E \;\Rightarrow\; \neg(x_j \wedge x_k) \tag{3.6}$$

  and

- exactly k vertices are marked:

$$|x| = k. \tag{3.7}$$

Define the set F of feasible solutions satisfying these constraints as

$$F = \left\{x \in \{0,1\}^N \;\middle|\; x \text{ satisfies eq. (3.6) and eq. (3.7)}\right\} \tag{3.8}$$

Consider a quantum register of N qubits in $(\mathbb{C}^2)^{\otimes N}$ and denote its computational basis by $\left\{|\mathbf{x}\rangle \;\middle|\; \mathbf{x} \in \{0,1\}^N\right\}$. We define the *feasible subspace* to be $\mathcal{F} = \text{span}_{\mathbb{C}}\{|x\rangle \mid x \in F\}$. Our goal is to construct a family $B_\beta = \{B(\beta) \mid \beta \in \mathbb{R}\}$ of mixing unitaries (cf. Definition 1) with respect to the constraints in eq. (3.6) and eq. (3.7). To this end, we follow the procedure described by Hadfield (2018). We start by constructing local and classical mixing rules and continue by translating those to unitary operators.

Starting with a feasible schedule x, we want the Hamming weight $|x|$ to be invariant under application of the mixing operators. This is ensured by only applying permutations $\pi \in S^N$ to the indices of a bit string, i.e. $\pi(\mathbf{x}) := \big(x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(N)}\big)$. The corresponding permutation unitaries

$$U_\pi |\mathbf{x}\rangle = |\pi(\mathbf{x})\rangle \tag{3.9}$$

can be extended to a family of unitaries

$$U_\pi(\beta) = \cos(\beta)\, \mathbb{1} - i \sin(\beta)\, U_\pi \tag{3.10}$$

which preserves the constraint eq. (3.7). In order to preserve the constraint eq. (3.6) we check whether a permutation will preserve the constraint via a boolean function $\chi\colon \{0,1\}^k \to \{0,1\}$ which decides if an input bit string (with an auxiliary bit attached) is feasible or not. Given a unitary operator U acting on k' qubits, we define the χ-controlled unitary operator (see Hadfield, 2018, p.145) $\Lambda_\chi(U)$ acting on $k + k'$ qubits by the linear extension of

$$\Lambda_\chi(U)\,|x\rangle\,|y\rangle = \begin{cases} |x\rangle\,|y\rangle\,, & \chi(y) = 0 \\ (U\,|x\rangle)\,|y\rangle\,, & \chi(y) = 1 \end{cases} \tag{3.11}$$

defined for computational basis states $|x\rangle$ and $|y\rangle$ given by $x \in \{0,1\}^k, y \in \{0,1\}^{k'}$. If we find a function

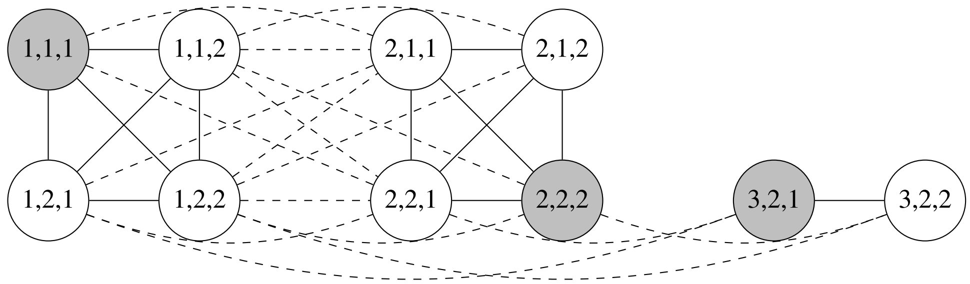


Figure 1. A visualization of an example constraint-graph for the FJSP. Here the instance is given by $J = \{J_1, J_2\}$, $O = \{O_{1,1}, O_{1,2}, O_{2,1}\}$, $M = \{M_1, M_2\}$, $M_{1,1} = M_{1,2} = M$, $M_{2,1} = \{M_2\}$ and $d_{j,o,m} = 1$ for all j, o, m. The nodes are labeled as i, m, t, where $i = 1$ denotes $O_{1,1}$, $i = 2$ denote $O_{1,2}$ and $i = 3$ denotes $O_{2,1}$. Due to the assignment constraint, each operation corresponds to a complete subgraph, where the edges are drawn solid. The rest of the edges, corresponding to other constraints, are drawn dashed. The nodes highlighted in grey correspond to a feasible solution of this FJSP instance. Note that no marked nodes are connected with an edge.

χ_π for each π such that $\chi_\pi(x) = 1$ if and only if $\pi(x)$ satisfies eq. (3.8), we can use it as the control for the permutation unitary $U_\pi(\beta)$. Denote the neighborhood of a vertex $j \in V$ by $\operatorname{nbhd}(j) = \{k \in V \mid (j,k) \in E\}$. If $x_j = 0$ there will be no conflict with the constraint regarding $x_{\pi(j)}$ even if other nodes in $\operatorname{nbhd}(\pi(j))$ are 1. On the other hand if $x_j = 1$ all neighbors of $\pi(j)$ need to be 0 after the permutation is applied in order for π not to raise a conflict. Hence, to prevent conflicts regarding node j we compute

$$\chi_\pi^j(x) = \neg(x_j) \vee \left(\bigwedge_{l \in \operatorname{nbhd}(\pi(j))} \neg(x_{\pi^{-1}(l)}) \right). \quad (3.12)$$

Now we can define the complete classical control predicate

$$\chi_\pi(x) = \bigwedge_{j=1}^{N} \chi_\pi^j(x). \quad (3.13)$$

Definition 3 (Partial graph-controlled permutation mixers)**.** For every permutation $\pi \in S^N$ define a *partial graph-controlled permutation mixer*

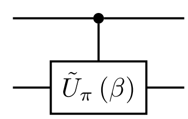
$$\Lambda_{\chi_\pi}\left(\tilde{U}_\pi(\beta)\right), \quad (3.14)$$

where χ_π is the control function defined in eq. (3.13) and

$$\tilde{U}_\pi(\beta) := \cos\beta(\mathbb{1} \otimes \mathbb{1}) - i \sin\beta(X \otimes U_\pi) \quad (3.15)$$

is a version of eq. (3.10) which is controlled on an auxiliary qubit $|c\rangle$, whose purpose is explained below, cf. Figure 2.

We present one possible approach for implementing the partial graph-controlled permutation mixers (3.14). We assume that an implementation of $\tilde{U}_\pi(\beta)$ controlled by a single qubit (see below) is known and focus primarily on the implementation of the graph-control.



We explicitly implement the logic of χ_π (3.13) using additional $3N + 2$ auxiliary qubits. Take some $x \in \{0,1\}^N$ and consider 3 N-qubits auxiliary quantum registers y, a, b with initial states $|0\rangle_y$, $|0\rangle_a$ and $|0\rangle_b$ and one-qubit registers with initial states $|0\rangle_c$ and $|0\rangle_z$ of one qubit each. The register y will contain information on the main working register x. Each of the remaining auxiliary registers will represent truth values of intermediate results in the calculation of $\chi_\pi(y)$. An overview of the purpose of each register can be found in table 1. Following the construction of sequential mixers presented in Hadfield, 2018 we state the following definition for the mixer operators $B(\beta)$.

Definition 4 (Graph-controlled permutation mixers)**.** Define the sequential mixer

$$B(\beta) := \left(\prod_{\pi \in P \subset S^N} B_\pi(\beta) \right) A, \quad (3.16)$$

where A and $B_\pi(\beta)$ defined as in Figure 2 and P is a suitable subset of permutations; a natural choice is presented in Lemma 1. More precisely, in order to implement $B(\beta)$ apply …

(a) … A as defined in Figure 2, i.e. for each $j = 1, \ldots, N$ apply $\mathrm{CNOT}_{j;j}\, |\psi\rangle_x \otimes |0\rangle_y$, where $|\psi\rangle_x$ is a superposition of feasible states in the main register. This step copies an unmodified version of the working register into the y register.

(b) ...a Pauli X_j on each qubit b_j, $j = 1, \dots, N$ in the register b, flipping the qubits in the b register.

(c) ...$C_j(\pi)$ for each $j = 1, \dots, N$ as defined in eq. (3.17). This step first uses each qubit a_j in the a register to store whether there are any neighbors of $\pi(j)$ which have the value 1 in the constraint graph corresponding to the schedule which applying π would yield. Then the bit b_j is flipped back to 0 if a_j has the value 0 and y_j has the value 1, i.e. if a conflict (two neighboring vertices in the constraint graph both have value 1) regarding the j-th variable arises. This implements the logic in eq. (3.12).

(d) ...a single multiqubit $\text{CNOT}_{1,\dots,N,\neg N+1;1}$ applied to the registers b and c (control qubits) and z. This stores in the register z whether all qubits in b_j have the value 1, i.e. if applying the permutation would yield a feasible schedule. This is the implementation of eq. (3.13).

(e) ...a single qubit controlled $\tilde{U}_\pi(\beta)$ as defined in eq. (3.15), applied to the registers c and x with $|z\rangle$ as control qubit. This applies the parameterized permutation, conditioned on register z (which indicates whether the permutation is feasible) and register c, which ensures that the permutation is only applied to the original $|\psi\rangle_x$ portion of the working register.

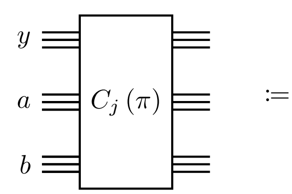
(f) Uncompute the auxiliary registers as depicted in Figure 2, i.e. apply the inverse operator of (b)-(e).

(g) Repeat (b)-(f) for every $\pi \in P \subset S^N$.

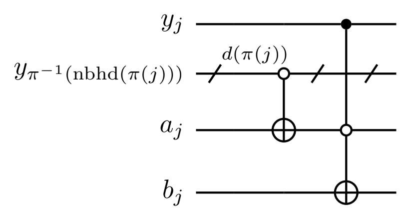
Here $\text{CNOT}_{l_1,\dots,l_n;k}$ denotes the multi qubit CNOT acting on qubits $l_1, \dots, l_n$ as control and on qubit k as target. A $\neg$ in an expression like $\text{CNOT}_{i,\neg j;k}$ denotes that the j-th qubit should act as negated control qubit, while the i-th qubit should act as normal control. The number of neighbors of a node j in the graph is denoted by $d(j)$. The whole procedure is illustrated in Figure 2, where $C_j(\pi)$ is defined in eq. (3.17)f.

Table 1. Quantum registers and their purpose

y	Aux. register: Copy of the initial state before the application of any permutations in the main register	N qubits
a	Aux. register: For each variable x_j: would rescheduling according to $\pi \in S_N$ lead to the assignment $x_l = 0$ for every neighbor l of $\pi(j)$?	N qubits
b	Aux. register: If $a_j = 1$ (else set $b_j := 1$): would rescheduling according to $\pi \in S_N$ lead to the assignment $x_{\pi j} = 1$?	N qubits
z	Aux. register: Are all of the variables in register b equal to 1	1 qubit
c	Aux. register: One variable ensuring that each layer only applies permutations to the original state	1 qubit
x	Main register: Modified by every time we apply $B_\pi(\beta)$ for a permutation yielding a legal schedule	N qubits



(3.17)



(3.18)

Note that the uncomputation leads to the states $|0\rangle$ only in registers a and b. For the auxiliary registers y, c and z, we require a fresh register for each layer of mixers. Thus, this protocol requires $2N + (2 + N)k$ auxiliary registers in total for N logical qubits and k layers in CM-QAOA.

There are several options to choose a suitable set of permutations P. Considering the entire set S^N fulfills all requirements of Definition 1 but is practically infeasible since it would require checking $|S^N| = N!$,

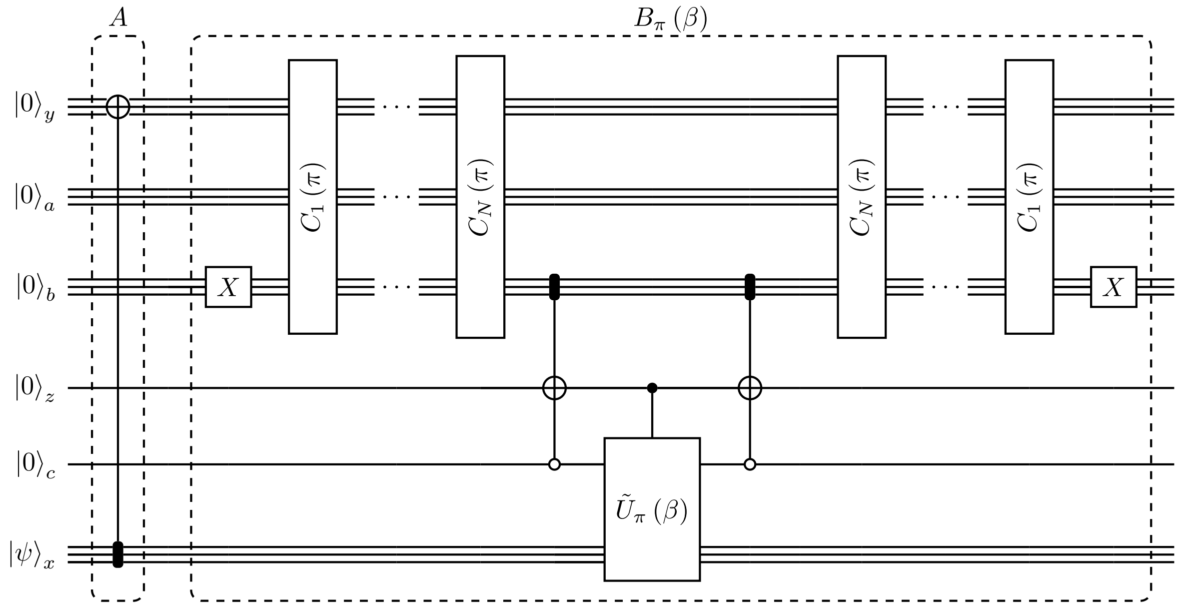


Figure 2. The quantum circuit of $B_\pi(\beta)\,A$ used for graph-controlled permutation mixers $B(\beta)$ from Definition 4. The $C_j(\pi)$ are defined in eq. (3.17) and $\tilde{U}_\pi(\beta)$ is defined in eq. (3.15). The auxiliary registers are initialized in the zero state, the working register contains a superposition of feasible states $|\psi\rangle_x$.

i.e. factorially many permutations. The following lemma states that it is sufficient to restrict to transpositions.

Lemma 1. Given feasible schedules $x, y \in F \subset \{0,1\}^N$, there are transpositions $\tau_1, ..., \tau_k \in S_N$ such that

$$\left(\prod_{i=1}^{k} \tau_i\right)(x) = y \text{ and } \left(\prod_{i=1}^{l} \tau_i\right)(x) \in F \qquad (3.19)$$

for each $1 \leq l \leq k$.

Proof. Assume that the makespans $C(x)$, $C(y)$ of schedules x and y satisfy $\max(C(x), C(y)) < \frac{1}{2} n_T$, else choose a higher cutoff for number of steps n_T.

Let $j \in J$ and $o \in O_j$ specify a job and an operation. Without loss of generality, we assume that the bijective enumeration function $\iota : O \times M \times T \to \{1, \ldots, N\}$ is built such that $\iota(o, m, 2t) = 2\iota(o, m, t)$. Define indices $b_{j,o}, c_{j,o} \in \{1, \ldots, N\}$ such that

$$\iota^{-1}(b_{j,o}) = (j, o, m, t) \text{ for some } m, t \text{ and } x_{b_{j,o}} = 1 \qquad (3.20)$$

and

$$\iota^{-1}(c_{j,o}) = (j, o, m', t') \text{ for some } m', t' \text{ and } y_{c_{j,o}} = 1. \qquad (3.21)$$

In other words $b_{j,o}$ and $c_{j,o}$ are the indices corresponding to operation o of job j whose entry in the schedule x resp. y is 1. As every feasible schedule assigns an operation to exactly one machine and starting time, the properties (3.20) and (3.21) characterize $b_{j,o}$ and $c_{j,o}$ uniquely.

Define transpositions $\tau_{j,o}^{1\to2}, \tau_{j,o}^{2\to1} \in S_N$ via

$$\tau_{j,o}^{1\to2}(a) = \begin{cases} 2a & \text{if } a = b_{j,o} \\ \frac{a}{2} & \text{if } \frac{a}{2} = b_{j,o} \\ a & \text{otherwise} \end{cases} \qquad (3.22)$$

and

$$\tau_{j,o}^{2\to1}(a) = \begin{cases} c_{j,o} & \text{if } \frac{a}{2} = b_{j,o} \\ 2b_{j,o} & \text{if } a = c_{j,o} \\ a & \text{otherwise.} \end{cases} \qquad (3.23)$$

$\tau_{j,o}^{1\to2}$ is the transposition which shifts the assignment of operation (j, o) according to schedule x into the second half of the available time window. The transposition $\tau_{j,o}^{2\to1}$ shifts the assignment of operation (j, o) back into the first half of the available time window, but into the position specified by the goal schedule y. Now, we define an order in which we apply the transpositions $\tau_{j,o}^{1\to2}$ and $\tau_{j,o}^{2\to1}$ for each job and operation (j, o). We first apply $\tau_{j,o}^{1\to2}$ and within every job j, we go from last to first operation, thereby ensuring that the order constraint is satisfied at all times.

$$\pi := \prod_{j\in J} \prod_{o=1}^{O_j} \tau_{j,o}^{1\to2}. \qquad (3.24)$$

In the second step, we apply the transpositions $\tau_{j,o}^{2\to 1}$, this time going from first to last operation within each job.

$$\rho := \prod_{j\in J}\prod_{o=O_j}^{1} \tau_{j,o}^{2\to 1}. \tag{3.25}$$

In total, we get the desired result

$$\rho\pi(x) = y. \tag{3.26}$$

□

Now we will show that the mixing operators given in Definition 4 preserve feasibility with respect to the constraints (3.8). This implies that, when starting in a feasible state and running CM-QAOA, the evolution stays in the feasible subspace of the given problem. There will be no overlaps with non-feasible states and the algorithm is exclusively focused on optimizing with regard to a given goal function. In the following, we denote the Hilbert space of the auxiliary registers by $\mathcal{H}_{\text{aux}} := (\mathbb{C}^2)^{\otimes(3N+2)}$ and $|0\rangle_{\text{aux}} := \bigotimes_{i=1}^{3N+2}|0\rangle$.

Theorem 1. (*Preserving feasibility*). The mixer operators $B(\beta)$ given in Definition 4 preserve feasibility with regard to the constraints given in eq. (3.8), i.e. for every $x \in F$, $\beta \in \mathbb{R}$ we have

$$B(\beta)\left(|x\rangle \otimes |0\rangle_{\text{aux}}\right) \in \mathcal{F} \otimes \mathcal{H}_{\text{aux}}.$$

Proof. Given any list of permutations $\pi_1, ..., \pi_n \in S_N$, the quantum state after applying $\prod_i B_{\pi_i}(\beta)A$ to a feasible initial basis state $|x\rangle$ with $x \in F$ is given by

$$\left(\prod_{i=1}^{n} B_{\pi_i}(\beta)\right) A\left(|x\rangle_x \otimes |0\rangle_{\text{aux}}\right) =$$

$$\left(\prod_{i=1}^{n} B_{\pi_i}(\beta)\right)\left(|x\rangle_x \otimes |x\rangle_y \otimes |0\rangle_{a,b,z,c}\right) =$$

$$\alpha\left(|x\rangle_x \otimes |x\rangle_y \otimes |0\rangle_{a,b,z,c}\right) +$$

$$\sum_{i=1}^{n} \alpha_i \left(|\pi_i(x)\rangle_x \otimes |x\rangle_y \otimes |0\rangle_{a,b} \otimes |\text{trash}(i)\rangle_{z,c}\right),$$

where $\alpha_i = 0$ iff $\pi_i(x)$ is not a feasible solution. This is ensured by construction of the control logic with auxiliary registers a, b, y and z. The control on auxiliary register c ensures by construction that a permutation π_i is only ever applied to the initial $|x\rangle$ portion of the superposition in the main register, cf. Figure 2. □

The second criterion B needs to meet in order for being a suitable mixer is that every feasible basis state in $\mathcal{F}$ can be reached by a number of repetitions. This is necessary for the phase-separator to have access to all possible solutions in order to find the best possible one. In fact, B as given in Definition 4 does have this property.

We define $|0\rangle_{\text{aux,k}} := \bigotimes_{i=1}^{2N+(N+2)k}|0\rangle$.

Theorem 2. (*Exploring the feasible subspace*). Let $x \in F$, denote the initial state in the auxiliary registers as $|0\rangle_{\text{aux},k} \in \mathcal{H}_{\text{aux},k}$, the mixer $B(\beta)$ as given in Definition 4 and $|\psi_{\text{mix},k,\beta}(x)\rangle := (B(\beta))^k\left(|x\rangle \otimes |0\rangle_{\text{aux,k}}\right)$. Further define $\rho^x_{\text{mix},k,\beta} := \text{Tr}_{\text{aux}}\left[|\psi_{\text{mix},k,\beta}(x)\rangle\langle\psi_{\text{mix},k,\beta}(x)|\right]$. Then for every $x, x' \in F$ there are some $k \in \mathbb{N}$ and $\beta \in \mathbb{R}$ s.t.

$$\text{Tr}\left[|x'\rangle\langle x'|\,\rho^x_{\text{mix},k,\beta}\right] > 0.$$

Proof. Let $x, x' \in F$. Denote the transpositions provided by Lemma 1 by $\tau_1, ..., \tau_k \in S_N$ and recall their properties

$$\left(\prod_{i=1}^{k}\tau_i\right)(x) = x' \text{ and } \left(\prod_{i=1}^{l}\tau_i\right)(x) \in F \tag{3.27}$$

for each $1 \le l \le k$. Then

$$\begin{aligned}|\psi_{\text{mix},k,\beta}(x)\rangle &= (B(\beta))^k\left(|x\rangle \otimes |0\rangle_{\text{aux,k}}\right)\\ &= \alpha\,|\tau_k\cdots\tau_1(x)\rangle \otimes |0\rangle_{a,b} \otimes |\text{trash}\rangle^{\otimes k}_{y,z,c} + |\text{rest}\rangle\\ &= \alpha\,|x'\rangle \otimes |0\rangle_{a,b} \otimes |\text{trash}\rangle^{\otimes k}_{y,z,c} + |\text{rest}\rangle\,,\end{aligned}$$

where $\alpha = \prod_{j=1}^{k}\alpha_j$, $\alpha_j > 0$ being the probability that we apply τ_j in the j-th layer. The remaining sum of feasible states $|\text{rest}\rangle$ may possibly contain a summand with $|x'\rangle$, so we choose β such that it does not cancel out the coefficient α. Denote the state in the auxiliary registers corresponding to x' by $|\psi(x')\rangle_{\text{aux}} := |0\rangle_{a,b} \otimes |\text{trash}\rangle^{\otimes k}_{y,z,c}$ as derived above. Then the probability of obtaining $|\psi(x')\rangle_{\text{aux}}$ when measuring the auxiliary registers of $|\psi_{\text{mix},k,\beta}(x)\rangle$ in the computational basis is given by $P(|\psi(x')\rangle_{\text{aux}}) > 0$. Thus, for a suitable choice of β and k as chosen in eq. (3.27) we have

$$\text{Tr}\left[|x'\rangle\langle x'|\,\rho^x_{\text{mix},k,\beta}\right] = P\left(|\psi(x')\rangle_{\text{aux}}\right)\alpha^2 > 0.$$

□

4. Discussion and Outlook

We have introduced mixer operators for the FJSP and have proven that they preserve feasibility and explore the entire feasible subspace. While the considered constraint graph model is the same as the one considered by Koßmann et al. (2022), the general approach is very different. The work by Koßmann et al. (2022) has implicitly shown, that for the FJSP permutation unitaries are not sufficient to construct mixing unitaries in the sense of Hadfield (2018), when no auxiliary qubits are used. In this work, we employ several auxiliary registers in order to ensure that the emerging mixing unitaries preserve the feasible subspace. In general, the described control protocol may be applied to all problems that can be modelled by a constraint graph. Additionally, our construction includes a few tricks which can be reused in other protocols, such as the auxiliary register c which stores the information whether an operator has been applied to the quantum state, thus avoiding that a product of permutations is applied. The direct implementation of constraints in the operators C_i can be used whenever constraints admit a boolean formulation as in eq. (3.12). The practicality of the presented mixing operators can be debated critically. The direct implementation $C_1 \ldots C_N$ of the control logic adds an overhead of $\mathcal{O}\left(N^3\right)$ Toffoli, CNOT and single qubit gates to the single qubit controlled U_π. In total, the mixer operator $B\left(\beta\right)$ applies at least $\mathcal{O}(N^2)\left(\mathcal{O}\left(N^3\right)+n_p\right)=\mathcal{O}\left(N^5\right)+n_p\mathcal{O}\left(N^2\right)$ Toffoli, CNOT and single qubit gates, where $n_p = \min\left\{n_\pi \mid \pi \in S^N\right\}$ denotes the minimal number of basic gates necessary to implement the single qubit controlled U_π. This results in a polynomial gate count, which we suspect, however, to be quite demanding on current quantum systems, considering their limited coherence times. A direction for further research could be to exploit the logical structure of specific constraints in order to enforce these in a more efficient way. For certain types of constraints, this has been achieved by Fuchs et al., 2022. Also, one could employ a hybrid approach, where some constraints are enforced via efficiently constructed constraint mixers, while for others the more common penalty term method is kept in place. Further, numerical simulations may give more insight on the potential of approaches using constraint mixers. Also, a combination of both the direct approach presented in this work and the abstract analysis of the constraint-graph (Koßmann et al., 2022) could reduce the number of necessary gates and result in an applicable set of mixing unitaries for solving the FJSP with the CM-QAOA. This can be possibly achieved following Meyer et al., 2023, which presents a concrete method to design variational circuits which take into account problem symmetries.

Another way to reduce the number of auxiliary qubits as well as the gate complexity of one single B_π (3.14), might be a deeper analysis of the logarithm of the fractional permutation unitaries (3.10). Finding a general logarithm for these unlocks a different control implementation by considering the exponential of the tensor product of the logarithm and the Hamiltonian simulating the classical control logic.

Acknowledgements

The authors thank L. Binkowski, J. R. Finžgar, C. B. Mendl, T. Osborne and B. Poggel for fruitful discussions and comments. LP and MH acknowledge financial support by the BMBF project QuBRA.

References

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., & Coles, P. J. (2021). Variational quantum algorithms. *Nature Reviews Physics*, *3*(9), 625–644. https://doi.org/10.1038/s42254-021-00348-9

Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. https://doi.org/10.48550/ARXIV.1411.4028

Fuchs, F. G., Lye, K. O., Møll Nilsen, H., Stasik, A. J., & Sartor, G. (2022). Constraint preserving mixers for the quantum approximate optimization algorithm. *Algorithms*, *15*(6). https://doi.org/10.3390/a15060202

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search.

Hadfield, S. (2018). Quantum algorithms for scientific computing and approximate optimization. https://doi.org/10.48550/ARXIV.1805.03265

Koßmann, G., Binkowski, L., & Schwonnek, R. (2022). Free job-shop scheduling with hardcoded constraints. https://doi.org/10.48550/ARXIV.2211.05822

Marx, D. (2004). Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering*, *48*, 11–16.

Meyer, J. J., Mularski, M., Gil-Fuster, E., Mele, A. A., Arzani, F., Wilms, A., & Eisert, J. (2023). Exploiting symmetry in variational quantum machine learning. *PRX Quantum*, *4*(1). https://doi.org/10.1103/prxquantum.4.010328

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, *26*(5), 1484–1509. https://doi.org/10.1137/s0097539795293172

Xie, J., Gao, L., Peng, K., Li, X., & Li, H. (2019). Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*, *1*. https://doi.org/10.1049/iet-cim.2018.0009