EPJ.org

EPJ Quantum Technology

REVIEW

Open Access

Quantum algorithms for scheduling problems: a survey

Tino Werner[1*] and Freyja Ullinger[2]

*Correspondence:
tino.werner@dlr.de
[1]Institute of Systems Engineering for Future Mobility, German Aerospace Center (DLR), Escherweg 2, Oldenburg, 26129, Lower Saxony, Germany
Full list of author information is available at the end of the article

Abstract

Quantum algorithms have the potential to solve combinatorial optimization problems faster than classical algorithms. A particular example for combinatorial optimization problems are scheduling problems. This work provides summarizes quantum or quantum-inspired algorithms for scheduling problems, providing an overview of 20 years of research. We categorize the approaches by problem type and algorithm type. A condensation of the reviewed literature to the main ideas and details about the considered problem size, solvers and evaluation metrics enables a quick comparison with and placement into the current state of research for future works. We further critically assess the comparability of the reviewed literature and present crucial metrics for future comparison.

Keywords: Scheduling Problems; Quadratic unconstrained binary optimization; Mixed-integer linear programming; Quantum genetic algorithms

1 Introduction

Scheduling problems are immensely important for production, healthcare, traffic, logistics, distributed systems, and many other application areas Abdalkareem et al. [1], Ikeda et al. [95], Pinedo [165]. There are many variants of scheduling problems in the literature, such as job shop scheduling problems Xiong et al. [224], open shop scheduling problems Anand et al. [9], flow shop scheduling problems Reza Hejazi and Saghafian [174], and flexible versions where more than one machine is available for at least one operation Chaudhry and Khan [41], Xie et al. [223]. See Allahverdi [7] for an excellent reference on scheduling problems and their classification.

Standard job shop scheduling problems consider a collection of jobs, consisting of operations, and a collection of machines. The operations within a job must be executed in a given order. A job is done once its last operation is done. Each machine is designed to handle one of the operations and can only execute one operation at a time. The goal is to find a schedule of the operations onto the machines such that some metrics are optimized, typically, one aims at minimizing the makespan, i.e., the time until the last job is finished. For such problems, Garey et al. [80] showed that finding a schedule that minimizes the makespan is NP-hard.

Springer

Quantum computing aims at exploiting properties of quantum mechanics in order to achieve advantages over classical computing on particular problem classes. Instead of operating with classical bits that are either in the state 0 or 1, quantum computers work with quantum bits, in short qubits, which can be in superposition of both states. A measurement leads to a manifestation of either state with a certain probability. Therefore, a collection of N qubits can be in a superposition of 2^N binary states. Based on this principle, there is a potential that quantum computers can solve some large-scale problems faster than classical computers. In this sense, quantum computing could have many application areas, such as electric power systems Zhou et al. [242], smart grids Ullah et al. [209], chemistry Motta and Rice [141], nuclear fusion Joseph et al. [100], or finance Herman et al. [92]. However, there are still many open problems, such as decoherence, scalability, or error correction.

Due to the complexity of scheduling problems and the potential to solve complex problems faster, quantum computers could be a promising tool to solve scheduling problems. This survey provides a quick overview of the currently existing quantum or quantum-classical hybrid methods for solving scheduling problems, which may form the basis for improvements and comparison, and is primarily addressed to researchers from industry and academia. It contains papers that only consider small-scale problems that are solved with a quantum algorithm as a proof of concept but also papers that provide hybrid solvers that allow for solving large-scale problems, from which some even mimick the size of real-world problems. As related work, an overview of quantum optimization algorithms in operation research is given in Klug [111], including a collection of selected papers on scheduling problems. A selection of quantum algorithms for scheduling problems are included in the review paper Pooja and Sood [168]. Chen et al. [45] provide a literature overview on quantum approaches in supply chain management, including production scheduling. An overview of quantum algorithms in medicine, finance and logistics is provided by Ciacco et al. [40], in which Sect. 5 is devoted to scheduling problems.

We distinguish between formalizations of the scheduling problems as a quadratic unconstrained binary optimization (QUBO) problem, the encoding via evolutionary algorithms (EAs), which form the majority of the strategies, and a small section with other approaches, which include formalizations as a constrained quadratic model (CQM). QUBO problems are often solved via quantum annealing. See Yarkoni et al. [233] for an overview of quantum annealing in industrial applications, including scheduling problems. As the name suggests, QUBO problems are unconstrained. Hence, the constraints are transformed into penalty terms that are added to the objective function. Consequently, a QUBO solution cannot be expected to satisfy the corresponding hard constraints. CQMs, in contrast to QUBO problems, allow for hard constraints (e.g., D-Wave [61]).

This paper is organized as follows: Sect. 2 provides an overview of different types of scheduling problems and typical constraints that need to be satisfied. Sect. 3 starts with a very short introduction into quantum computing. Then, we present how scheduling problems can be expressed as Ising models, and compiles basic aspects of quantum-inspired evolutionary algorithms (QIEAs). In Sects. 4, 5, and 6, the works from the literature concerning formulations of scheduling problems as quadratic optimization problems, the application of QIEAs, or other types of formalizations and algorithms are reviewed in a chronological order, respectively. For each work, the problem type, constraints, objectives, proposed solvers (optionally, the used commercial hardware), and problems sizes in the

experiments are listed. Usually, additional information about the proposed algorithm, the comparison with classical or other quantum algorithms, and the evaluation metrics are provided. Sect. 7 reviews some quantum hardware requirements and resulting challenges for large-scale optimization problems. Sect. 8 critically discusses the methodology and makes recommendations for future work.

2 Scheduling problems

Scheduling problems are closely connected to sequencing problems and to assignment problems, and essentially is a combination of both Pinto and Grossmann [163]. In sequencing problems, the goal is to find the optimal sequence of n jobs, while disregarding any distribution of the jobs onto different machines or the determination of time points when a job should start. In the special case of single-machine scheduling problems, sequencing and scheduling problems are identical when the time points are defined implicitly. In that sense the whole production starts at some time t_0 and one job starts directly after the previous job has been finished. Assignment problems consider the distribution of n objects onto m slots, where $n \geq m$. In contrast to scheduling problems, which, for multi-machine scheduling, also include the allocation of jobs onto different machines, any type of ordering of the objects is not considered, making assignment problems also weaker optimization problems than scheduling problems.

2.1 Types of scheduling problems

A *job shop scheduling problem* (JSSP; Xiong et al. [224]) considers n jobs, $J_1,\ldots,J_n$, and m machines, $M_1,\ldots,M_m$. The individual jobs J_i are composed by K_i operations $O_i^{(1)},\ldots,O_i^{(K_i)}$, which have to be executed in a certain order. The job J_i is not finished until the last operation corresponding to this job is finished. In the standard JSSP, there is only one machine on which an operation $O^{(k)}$ can be executed. A softer version of the JSSP is the *flexible JSSP* (FJSSP; Chaudhry and Khan [41], Xie et al. [223]), which potentially allows for an operation to be executed on more than one machine. In this case, there is a set $\mathcal{M}^{(k)} \subset \{M_1,\ldots,M_m\}$ of machines on which operation $O^{(k)}$ can be executed. The sets of machines that can handle a specific operation are fixed in advance.

Another relaxation of the JSSP is the *Open Shop Scheduling Problem* (OSSP; Anand et al. [9]), where the order of the operations corresponding to the jobs is arbitrary, but each operation can only be executed on one machine, i.e., $|\mathcal{M}^{(k)}| = 1$ for all k. The *flexible OSSP* (FOSSP) allows for operations to be executed on potentially more than one machine, i.e., $|\mathcal{M}^{(k)}| \geq 1$ for all k.

A more restricted variant of the JSSP is the *flow shop scheduling problem* (FSSP; Reza Hejazi and Saghafian [174]). Here, each job J_i is composed of $K_i = m$ operations, and where the j-th operation can only be executed on the j-th machine. A particular type of FSSP is the *no-wait FSSP* (NW-FSSP), where the processing of the next operation of a job on the next machine must be started immediately after the current operation has been processed on the current machine. Consequently, there should be no waiting time of a job between the machines, which may cause that the whole processing sequence for one job must be delayed in order to meet this constraint. These no-wait constraints may arise due to the necessity to keep temperature or material properties constant Allahverdi [8].

Workflow scheduling problems (WSPs; Wu et al. [219]) generalize job scheduling problems. In this case each operation requires a given amount of resources/workers in order

to start. Job scheduling problems can be interpreted as the special case where one worker or one machine, is sufficient.

The problems above were all **multi-stage job scheduling problems**, indicating that each job is composed of more than one operation. In single-stage job scheduling problems, each job corresponds to exactly one operation.

Single-machine scheduling problems (SMSP; see Allahverdi [7]) consider the special case of a single-stage job scheduling problem where only one machine is available. An *unrelated machines scheduling problem* (UMSP; Durasevic and Jakobovic [52], Pfund et al. [162]) considers multiple machines and allows for individual processing times of each job and each machine, which means, a job can take different processing times if executed on different machines. A more restricted version is the *identical-machines scheduling problem* (IMSP), where the processing time only depends on the operation.

The scheduling problems above assume that all parameters are known in advance, which is often not satisfied in practice. If there are non-deterministic parameters, for example, the processing time, one faces an uncertain scheduling problem. In the case that one can at least assume a probability distribution for the uncertain parameters, the problem is called a stochastic scheduling problem Gu et al. [75].

The different types of scheduling problems are characterized by their assumptions and constraints, not by their objective, for which there may be different alternatives. The objective that can be encountered most frequently in the literature is the minimization of the makespan, which is given by the finishing time of the last job. Early and late delivery costs occur when a job is finished before or after its due time. Production costs can appear in manifold ways, e.g., by having a fixed amount of costs assigned to the execution of a particular job on a particular machine Amaro et al. [13]. In specialized scheduling problems such as nurse scheduling or timetabling, the objective is often just feasibility, that is finding a schedule which satisfies all constraints.

2.2 Typical constraints

Typical constraints that are implied by a scheduling problem include:

Assignment constraints (A): An operation can only be executed on designated machines,

Duration constraints (D): The time between the starting time of two operations on the same machine must be at least the timespan the machine requires for the first of the two operations,

Preemption constraints (E): An operation that has started on a machine must not be interrupted,

Finishing constraints (F): If a deadline is given, all jobs must be finished before this deadline,

Imambiguity constraints (I): An operation must have a unique starting point in the schedule,

Machine constraints (M): A time slot in a machine can be occupied by at most one operation at a time. This constraint seems to be superfluous due to D, however, in many works, time slots for the machines are considered, leading to the constraint that each time slot must be occupied only once in each machine, which is represented by M. Moreover, M can represent the general condition, not bound to job scheduling problems, that an object can only execute one task at a given time step,

Precedence constraints (P): Operations corresponding to one job need to be executed in the correct order. For single-stage scheduling problems, such constraints indicate that the jobs must follow a specified order or that at least some job are to be prioritized,

Resource constraints (R): In WSPs, a job can only start if sufficiently many workers are available. Note that R generalizes M in the sense that M is the special case of R if only one worker is used,

Shutdown constraints (S): Machines cannot be interrupted until the last operation scheduled on them is finished, i.e., there must be no waiting time between the finishing time of one operation and the starting time of the next operation. S is not covered by the special case of D that the difference between the starting points of two consecutive operations is exactly the timespan for the first operation as S also indicates that there must not be an interruption during the execution of an operation.

The constraints I are given in each of the reviewed references, either explicitly, or at least by construction due to the extraction of a schedule from the qubit state. Therefore, when listing the constraint sets that have been considered in each of the papers we review in the following two sections, we omit the constraint set I in the constraint lists. Constraint set E is also found in each corresponding paper. Therefore, we also exclude it from the list.

Constraints S, which indicate that an interruption within or between the execution of operations us forbidden, must not be confused with preemption constraints E, that only forbid that a machine is interrupted during an operation.

2.3 Benchmark data sets

There are several data sets onto which many of the scheduling algorithms reviewed in this paper are applied. The Kacem data (Kacem et al. [109]) includes five data sets, usually referred to as Kacem1 to Kacem5. The largest, Kacem5, has 15 jobs, 10 machines and a total number of 56 operations, up to 4 per job. The Brandimarte data (Brandimarte [30]) includes 10 data sets, referred to as MK01 to MK10. The largest, MK15, has 30 jobs, 15 machines and up to 12 operations per job. The Dauzère-Pérès data (Dauzère-Pérès and Paulli [56]) consists of 18 data sets, referred to as LA01 to LA18. The largest ones, La13 to LA18, have 20 jobs, 10 machines and up to 25 operations per job. The Taillard data (Taillard [200]) includes data sets for FSSPs with up to 500 jobs and 20 machines, JSSPs with up to 100 jobs and 20 machines, and OSSPs with up to 20 jobs and 20 machines. The Carlier data (Carlier [37]) includes 8 data sets for FSSPs, referred to as car1 to car8, where the largest has 8 jobs and 9 machines/operations. The abz data (Adams et al. [3]) consists of 5 data sets, referred to as abz5 to abz9, where the largest, abz9, has 20 jobs, 15 machines and 15 operations per job.

3 Quantum computing

In this section, we provide a brief introduction into quantum computing. We further present two models for solving scheduling problems with quantum and quantum-inspired algorithms.

3.1 Quantum states

A classical bit is the simplest unit of computation which can either take the value 0 or 1 deterministically. In contrast to that, a quantum bit or qubit state

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

can be in a superposition of the computational basis states $|0\rangle$ and $|1\rangle$ with complex-valued coefficients α and β. These coefficients represent the weight of each basis state in the superposition as well as the relative phase and satisfy the normalization constraint $|\alpha|^2 + |\beta|^2 = 1$. Therefore, for a measurement in the computational basis, they correspond to the probabilities of obtaining the basis states $\{|0\rangle, |1\rangle\}$. More precisely, we measure $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$.

In order to form a qubit bitstring out of single qubits we utilize the tensor product $\otimes$. For example, the tensor product of two qubit states $|\psi_1\rangle = \alpha_1 |0\rangle + \beta_1 |1\rangle$ and $|\psi_2\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle$ leads to the two-qubit state

$$|\psi_1\rangle \otimes |\psi_2\rangle = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle.$$

Here, we omitted the tensor product on the right-hand-side and combined both computational basis states into one. As a result, $|00\rangle$ corresponds to $|0\rangle \otimes |0\rangle$, $|01\rangle$ corresponds to $|0\rangle \otimes |1\rangle$ and so forth. Two-qubit states which can be repesented in terms of a tensor product of two single-qubit states are called product states. However, not every two-qubit state can be expressed as a tensor product of two single-qubit states as the state

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

shows. The zero amplitudes for the states $|01\rangle$ and $|10\rangle$ only vanish if, for instance the two qubits have zero amplitude for the state $|0\rangle$, or similarly for each being in $|1\rangle$. This however, is conflicting to the overall form of the state $|\psi_3\rangle$, where the amplitudes for $|00\rangle$ and $|11\rangle$ are non-zero. Hence, the two qubits cannot be represented by a tensor product state. In this case, the states are called entangled. This state in particular is called a Bell state and is maximally entangled, meaning that there is a maximum of correlation between the two qubit systems. Entanglement is a purely quantum property and together with the superposition, it is the main reason why quantum computing could provide advantages over classical computing, as entangled qubits contain correlations that the individual qubits cannot represent. These definitions and properties can be extended to any N-qubit system. In this case, a composite N-qubit state is not entangled if it can be written as a tensor product of N single qubit states.

3.2 Quantum gates

In order to exploit a possible quantum advantage, one has to create and maintain the superposition and entanglement of qubits, during the computation of a quantum algorithm. For this purpose, we utilize specific unitary operators, also called gates, that can be applied to quantum states. In the following, we consider some examples of single and two qubit quantum gates in order to provide a first illustration of quantum algorithms. For a more in depth discussion, we refer to Nielsen and Chuang [144].

To visualize the effect of a quantum gate on a quantum state, we perform a change in the representation of the states and operators. In general, quantum states are denoted using the Dirac notation $|\cdot\rangle$, as presented above. However, it is also possible to identify the basis states of the two-dimensional system with two-dimensional basis vectors of the vector space $\mathbb{C}^2$. In this case, a popular choice is

$$|0\rangle \hat{=} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and

$$|1\rangle \hat{=} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Consequently, we are able to rewrite any single-qubit state $|\psi\rangle$ as a vector

$$|\psi\rangle \hat{=} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

and any two-qubit state as the vector

$$|\psi_{12}\rangle \hat{=} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$$

where α, β, γ and δ denote the complex-valued coefficients of the basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. In principle one can map any N-qubit state to a 2^N-dimensional vector.

In this regard, the unitary operators then correspond to a unitary matrix. For example, the Z-gate is characterized by the matrix

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

From this notation, one can clearly see that the computational basis states $|0\rangle$ and $|1\rangle$ are eigenstates of the Z-gate with eigenvalues ± 1. Hence, when applied on an arbitrary qubit state, it yields a phase factor, which is global in terms of a single basis state and relative when considering a superposition of basis states. Other important operations are the X-gate, represented by

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

which is a qubit flip and the Hadamard-gate, represented by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

which creates an equally weighted superposition of states $|0\rangle$ and $|1\rangle$, when applied on the latter. Next to the single qubit gates, there exist also multi-qubit gates, which act on multiple qubits simultaneously. The most popular one is the controlled NOT-gate (CNOT-gate)

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

which, applied on two qubits, flips the state of the second qubit if the state of the first qubit (which hence serves as "control qubit") is in state $|1\rangle$ and does nothing otherwise.

3.3 Creation of entanglement

One way to create entanglement between two qubits is to employ a combination of Hadamard and CNOT-gate, as displayed in Fig. 1. For this purpose, we prepare two qubits in the state

$$|00\rangle \hat{=} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Then, we apply a Hadamard gate on the first qubit to create an equally weighted superposition of basis states, while doing nothing with the second qubit. Subsequently, we arrive

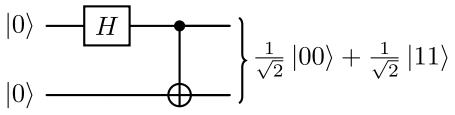

Figure 1 Quantum circuit for creating a Bell state. First two qubits are prepared in the state $|0\rangle$. Then, we apply a Hadamard-gate on the first qubit to create an equally weighted superposition of the two basis states. Finally, we apply a CNOT-gate with the first qubit as control qubit and the second one as target qubit

at

$$(H \otimes I)\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix}.$$

Here

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

denotes the two-dimensional identity matrix. Next, we take the first qubit as a control qubit and apply a CNOT-gate on both qubits. This leads us to our final vector

$$\mathrm{CNOT}(H \otimes I)\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\1\end{pmatrix},$$

which corresponds an entangled state and in particular a Bell state. In this case, the measurement of one system in its computational basis automatically reveals the state of the second system and the two-qubit state collapses into a product state of two computational basis states. Hence, the composite state looses its entanglement, after a measurement in the computational basis.

3.4 Quantum computing for scheduling problems

The main aim for this section is show how a scheduling problem can be formalized in a such way that a quantum algorithm can be applied. For this purpose, we discuss two popular approaches. The first approach is the identification of the scheduling problem with a particular discrete optimization problem onto which quantum algorithms can be applied. In the second approach, we consider the generation of a problem-specific fitness function, which is used for an evolutionary algorithm. In any case, the individuals are represented by qubits rather than bits.

3.4.1 Quantum formalizations for discrete optimization problems

It has been observed in Rieffel et al. [178] that some types of scheduling problems can be identified with graph coloring problems such as vertex coloring problems where in a graph, a color from a finite set of colors is assigned to each vertex such that no adjacent vertices have the same color. In this case, the constraint that a machine cannot handle more than one job simultaneously is equivalent to the constraint that two vertices that are connected by an edge must not have the same color. Graph coloring problems, among many other combinatorial optimization problems, can be formulated via Ising models, which has been shown in Lucas [128].

In the following, we explain why the formulation as Ising model enables the application of quantum algorithms and briefly describe common quantum algorithms to solve

Ising models. Then, we conclude with a discussion on how a scheduling problem can be expressed by such an Ising model.

An Ising model Lucas [128] is given by a Hamiltonian of the form

$$\hat{H}_I = -\sum_{i=1}^{N} h_i \hat{s}_i + \sum_{i \neq j=1,\dots,N} \sum J_{i,j} \hat{s}_i \hat{s}_j,$$

where the $J_{i,j}$ and h_i are real numbers that represent the strength of the interactions between two spins and the strength of the external field, respectively. The eigenstates of the spin operators $\hat{s}_i$ correspond to the computational basis states $|0\rangle$ and $|1\rangle$ with eigenvalues ± 1.

There is a close correspondence between Ising models and a particular class of discrete optimization problems, namely QUBO (quadratic unconstrained binary optimization) problems. A QUBO is an optimization problem formulated in terms of binary variables x_i such that a quadratic objective function

$$f(x_1, \dots, x_N) = -\sum_{i=1}^{N} p_i x_i + \sum_{i \neq j=1,\dots,N} \sum \tilde{Q}_{ij} x_i x_j$$

is to be minimized. Since, $x_i \in \{0, 1\}$, one can always find coefficients Q_{ii} such that the objective function can be expressed by the more convenient notation

$$f(x_1, \dots, x_N) = \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij} x_i x_j.$$

As the name suggests, there are no hard constraints. If constraints need to be respected, it can only happen via additional penalty terms. For example, an equality constraint set of the form $\sum_{i=1}^{N} a_i x_i = b$ is expressed as $\left(b - \sum_{i=1}^{N} a_i x_i\right)^2$. Usually, inequality constraints can be reduced to equality constraints by introducing slack variables (Boyd and Vandenberghe [36]). However, there are some recent approaches that directly integrate inequality constraints, e.g., Bottarelli et al. [33], Vyskočil et al. [213], Yonaga et al. [230]. Once a QUBO problem is formulated, one can transform it into an Ising model by substituting the binary variables $x_i \in \{0, 1\}$ by spin variables $s_i = 2x_i - 1 \in \{-1, 1\}$. Consequently, one can directly encode the problem into an Ising Hamiltonian $\hat{H}_I$, paving the path for the application of quantum annealing (QA) techniques. General mapping, navigation and scheduling problems and their formulation as a QUBO problem has been considered in Rieffel et al. [179].

Although QA techniques have not shown a quantum advantage yet, they are often favored as they are able to run on currently noise hardware. In a nutshell, they are based on the adiabatic evolution of a state in a time-dependent system. In this process the system Hamiltonian

$$\hat{H}_{\text{tot}} = a(t)\hat{H}_0 + b(t)\hat{H}_I$$

is slowly varied so that the initially prepared quantum state remains in the ground state of the system at all times. More precisely, QA starts with some external field Hamiltonian $\hat{H}_0$ of which a ground state can be prepared easily. The external field, described by $a(t)$, decays while the Ising Hamiltonian is slowly enforced to the system, indicated by $b(t)$. Consequently, the system evolves according to the convex combination of $\hat{H}_0$ and $\hat{H}_I$. The functions $a(t)$ and $b(t)$ are monotonic such that $a(0) = 1$, $b(0) = 0$, $a(T) = 0$, and $b(T) = 1$, for the annealing time T. By the adiabatic theorem (e.g., Morita and Nishimori [138]), if the conditions are met and if the mixed Hamiltonian evolves sufficiently slowly towards the Ising Hamiltonian, the system will always stay in the ground state. Hence, it starts with the ground state of $\hat{H}_0$ and ends with the desired ground state of $\hat{H}_I$. Next to QA, other algorithms such as a variational quantum eigensolver (VQE) or the quantum approximate

optimization algorithm (QAOA; Farhi et al. [69]), have also been applied to Hamiltonians corresponding to scheduling problems, as the remainder of this paper will show.

Coming back to scheduling problems, although they can be identified as graph coloring problem and be formulated as QUBO problems, there can be domain-specific side conditions that do not allow for quadratic binary terms, making a QUBO formulation impossible. An alternative class of optimization problems are mixed-integer linear programs (MILPs). Although they require that the constraints can the objective function can be linearized, they have the advantage that the time, among other variables, can directly be represented as a real-valued variable. More precisely, one would usually use real-valued variables in order to express starting times of operations, resulting in both discrete and real-valued decision variables. As for a QUBO formulation, once the integration of concrete time points are necessary, one could express the integer and real variables using a binary expansion, which has been mentioned in Zhang et al. [235], but the standard strategy is to use a time-index QUBO formalization. Here, the time enters as index, so that in a job shop scheduling problem, a binary variable $x_{o,m,t}$ may indicate whether some operation o starts at time t on machine m. Clearly, this strategy, although very popular in the literature, has two pitfalls. One is the sheer amount of binary variables, which linearly increases with the time horizon. The other one is the necessity to find an upper bound T for the time. This has, for example, been done in the literature by applying a classical algorithm first Zielewski et al. [234] and to extract T as the maximum makespan observed. Given the large number of binary variables that form a QUBO problem, one could ask whether there is a way to reduce the number of qubits in order to encode them. Prior works on a qubit-efficient encoding of QUBO problems exist, e.g., Tan et al. [205], who propose to divide the QUBO problem into sub-problems. This strategy would, in the best case, enable to encode a QUBO of N variables with $\log_2(N)$ qubits and some ancilla qubits, but for the expense that certain correlations can no longer be expressed in this case. Nevertheless, correlations can be expressed by other means, and their strategy has already been applied. However, Perelshtein et al. [166] point out that it remains unclear how to train the corresponding circuits efficiently. Therefore, one can consider qubit-efficient QUBO encoding as an open and active research topic.

Sect. 4 also includes references where a constrained quadratic model (CQM) formulation is used. Although these are related to QUBO problems, and Koniorczyk et al. [110] point out that the CQM solver works by identifying smaller QUBO-sub-problems which are solved by QA. In order to boost the speed of the classical parts of the solver, the mathematical formulation differs from a QUBO formulation. Therefore, we collect them in a separate subsection. Binary quadratic models (BQMs) cover both QUBO problems and Ising models according to the D-Wave documentation.[1] Therefore, BQM formulations are just QUBO problems that we allocate into the QUBO subsection.

3.4.2 Quantum version of evolutionary algorithms

Quantum versions of evolutionary algorithms (EAs) go essentially back to Narayanan and Shmatikov [147], who describe a quantum-inspired genetic algorithm. As for the terminology, quantum-inspired algorithms are no pure quantum algorithm, which is explained

[1] https://test-projecttemplate-dimod.readthedocs.io/en/latest/reference/bqm/binary_quadratic_model.html.

in Narayanan and Shmatikov [147] with the drawback that classical procedures are required in order to assess the candidate solutions. More precisely, the current qubit population must be measured in order to derive a fitness measure. Another early work where a quantum-inspired evolutionary algorithm (QIEA) has been proposed and tested is the work of Han and Kim [89].

A common feature of such algorithms is that one generates a number of individuals, which are iteratively modified, so that at the end, they can be considered as solution candidates. The collection of all individuals is called the population. As for quantum evolutionary algorithms, the individuals are encoded by qubits, more precisely, at time step t, each individual i is represented by an m-qubit vector $q_i(t)$. In the first papers on the application of such algorithms for scheduling problems, such as Wang et al. [221], FSSPs have been considered, only requiring a permutation of the jobs. Therefore, one can define an individual by

$$q_i(t) = \left(\begin{array}{c|c|c} \alpha_{i11}(t)\dots\alpha_{i1m}(t) & \dots & \alpha_{im1}(t)\dots\alpha_{imm}(t) \\ \beta_{i11}(t)\dots\beta_{i1m}(t) & \dots & \beta_{im1}(t)\dots\beta_{imm}(t) \end{array}\right), \tag{1}$$

where the vertical lines just emphasize the partition into m genes. The whole population at time step t is denoted by $Q(t) = \{q_1(t), \dots, q_N(t)\}$. In order to let the quantum individuals evolve, the evolutionary procedures are composed by suitable gates. For example, mutation indicates that for a specific qubit, the two amplitudes are exchanged, which can be realized with a X-gate (e.g., Saad et al. [184]). The updating step for the whole individuals in each generation is done via rotation and Hadamard gates. In order to targetedly update the individuals, the quality of the represented solution must be evaluated. This is done via a fitness function. However, the fitness value can only be evaluated by measuring each individual. This step is not directly possible on a quantum computer, as the measurement would let the superposition of states collapse to an eigenstate of the measurement operation. Of course, one could consider cloning $Q(t)$ and measuring the copy. However, according to the No-Cloning theorem, stating that independent and identical copies of arbitrary unknown quantum states are impossible, it would be impossible to copy $Q(t)$ for a measurement (cf. Sofge [190]).

In order to find the fittest individuals and avoiding the measurement of the qubit population in each iteration, Udrescu et al. [210] proposed the utilization of Grover's Search Grover [84]. To this end, they do not only use a quantum register for the individuals, but also a fitness register. Such a register has already been proposed by Rylander et al. [177]. It is however not clear how one would design the fitness operator from Udrescu et al. [210], which computes the fitness values of the quantum population, in practice. Sofge [190] at least put into question whether the selection and extraction of the fittest individual found by Grover's Search would not let the superposition collapse. According to Lahoz-Beltra [119], the problem how to compute the fitness values from the quantum population without letting the superposition collapse remains an open problem.

When applying a QIEA to a job shop scheduling problem, a crucial question is the representation of the solution, the actual schedule. This problem has already been solved in Bean [14], who propose a random key representation. The idea is to first convert the qubit string that represents an individual into a binary string. Using the representation from Eq. (1), for a randomly selected threshold $\eta \in [0, 1]$, one generates a vector of length m^2.

In each entry j, this vector has the value $I(|(q_i(t))_{1,j}|^2 > \eta)$. Therefore, each gene represents a number in $\{0, 1, \ldots, 2^m - 1\}$ in binary encoding, implying an ordering of the m jobs. For example, if $m = 3$, a measurement could lead to the realization (001|110|011), representing (1|6|3). Here, the substrings of the binary string that correspond to the individual jobs are converted into decimal numbers, whose order represents the order of the jobs. In this example, one executes the first job first, then the third, and finally the second. As this technique covers only problems where the assignment of operations to machines is not necessary, such as FSSPs, Gu et al. [75] proposed an extension that can also handle general JSSPs. In general, for n jobs, m machines, and the maximum number K of operations per job, let an individual $q_i(t)$ be composed by at most $Km(\lfloor \log_2(n) \rfloor + 1)$ qubits. Using an analogous binarization strategy, a string of Km job IDs in $\{1, \ldots, n\}$ is formed. Each job ID i must appear K_i times, where each occurence represents one operation of the job. The final schedule that decides which operation of which job is executed in which order and on which machine however requires additional information about the machine onto which operations can be executed and their execution time. Merging this information with the job ID sequence, Gu et al. [75] can generate the final schedule.

In this review paper, we decided that we include all papers where QIEAs have been proposed for scheduling problems, even though the proposed QIEAs are emulated on classical computers. A possible reason for not using quantum hardware could be the fact that the quantum hardware was not yet available or that preparing the qubit population again in each generation, due to the collapse after the measurement, could have been expensive when using commercially available quantum computers. In fact, many quantum algorithms that we included in the review so far are hybrid algorithms. For instance, hybrid CQM solvers or algorithms such as QAOA where the optimal parameters are found using classical optimization, necessiting the combination of both classical and quantum computing. Clearly, preparing the qubit individuals again in each iteration of a QIEA would be impractical. However, decomposition methods for QUBO problems such as Bender's decomposition (Benders [15]) that already entered scheduling problems, also alternatingly apply a quantum solver and a classical solver, necessiting to prepare the quantum problem anew according to the solution of the classical solver in each iteration. Although some papers call the proposed QIEA "hybrid", we omit this term, since in fact all QIEAs are hybrid in this light.

In the early literature, such as Han and Kim [88], it is argued that the probabilistic nature of the population already allows for more diversity than a classical population and thus maybe making genetic operators obsolete. Nonetheless, many of the more recent works incorporate such evolution operators. In contrast to the application of quantum algorithms that require a problem-related Hamiltonian as in the previous section, here, the main effort is to suitably encode the qubit population and to find a problem-related fitness function.

4 Quadratic problem formulations of scheduling problems

We first provide a graphical overview of the references where a scheduling problem was formulated as a QUBO problem or a CQM and solved via a quantum or hybrid quantum-classical approach. More details about the respective references are given later in the cited sections.

In Fig. 2, the references from Sects. 4.1.1 and 4.2 where single-stage job scheduling problems are formulated as QUBO problems or CQMs are compiled. Fig. 3 contains

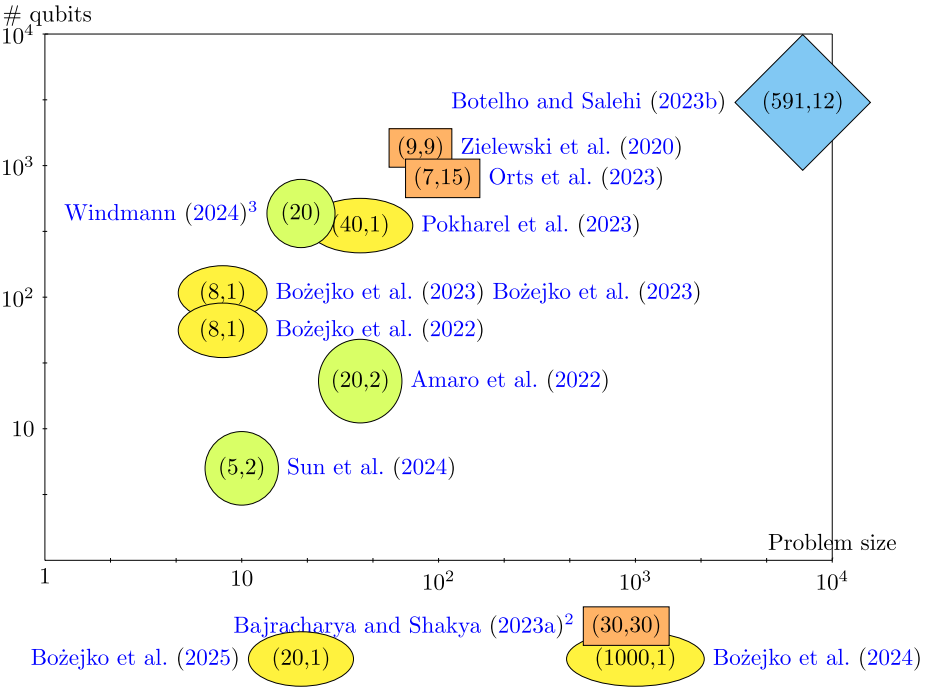

Figure 2 Overview of single-stage scheduling problems that are formulated as a QUBO problem or as a CQM. The horizontal axis refers to the size of the problem, quantified as $n \times m$, for the number n of jobs and the number m of machines. The vertical axis denotes the number of qubits used. The location of the node, with respect to its center, is determined by the largest combination of problem size and number of qubits presented in the corresponding reference. The vectors in the nodes show the largest problem instances addressed in the respective paper, denoted by (n, m). A rectangle refers to an UMSP, a circle to an IMSP, an ellipse to an SMSP, and a diamond to a fixed-interval scheduling problem. References where the number of qubits was not clearly reported are depicted below the horizontal axis

the references that formulate a multi-stage job scheduling problem as a QUBO problem, see Sect. 4.1.2. Fig. 4 depicts all references where a railway or flight scheduling problem (Sect. 4.1.3), a shift scheduling problem (Sect. 4.1.4), or some other type of scheduling problem (Sect. 4.1.5) has been formulated as a QUBO problem.

4.1 QUBO formulations

4.1.1 Single-stage job scheduling problems

Zielewski et al. [234]: **Type:** UMSP, **Constraints:** D, P, **Objective:** Makespan, **Solver:** QA (2000Q), **Size:** Up to 9 jobs and machines; up to 1214 qubits.

They study the performance of QA in dependence of the embedding size of the QUBO, empirically showing that the success probability decreases with the size. They also consider pausing the annealing, which can improve the performance, but which is itself susceptible to the embedding format.

Amaro et al. [13]: **Type**: IMSP, **Constraints:** A, M, P, S, **Objective:** Early delivery costs, late delivery costs, production costs, **Solver:** VQE, Filtering VQE, VarQITE, QAOA, **Size:** Up to 20 jobs and 2 machines (23 qubits).

They consider an IMSP in a steel factory. In their formulation, the time interval is discretized into time slots, so that M and S enforce that is job is assigned to exactly one time slot and that a series of jobs executed on a machine is assigned to consecutive time slots.

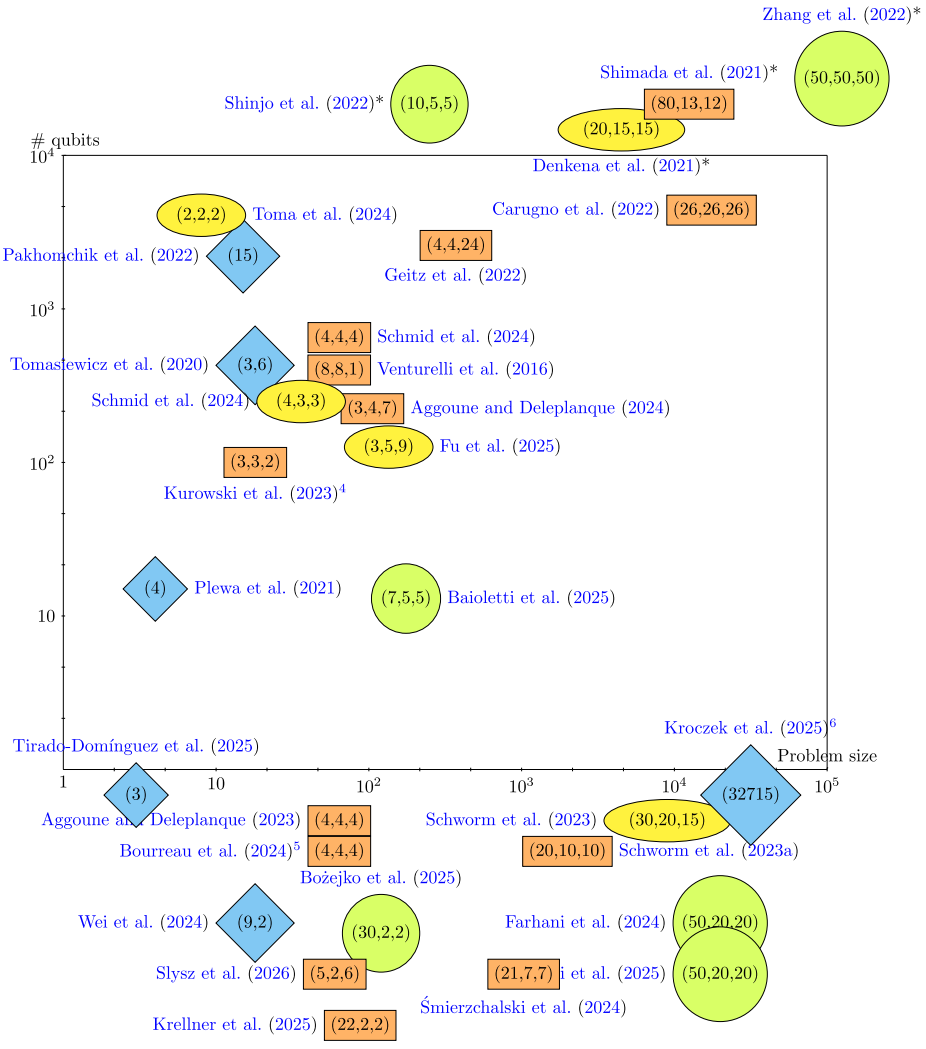

Figure 3 Overview of multi-stage job scheduling problems that are formulated as a QUBO problem or CQM. The horizontal axis refers to the size of the problem, quantified as $n \times m \times K$, for the number n of jobs, the number m of machines and the (in some references, average) number K of operations per job. For WSPs, the size is $n \times m$, where m is again the number of machines and where n is the number of tasks. The vertical axis denotes the number of qubits used. The location of the node, with respect to its center, is determined by the largest combination of problem size and number of qubits presented in the corresponding reference. The vectors in the nodes show the largest problem instances addressed in the respective paper, denoted by (n, m, K). A rectangle refers to a JSSP, a circle to an FSSP, an ellipse to an FJSSP, and a diamond to a WSP. References where the number of qubits was not clearly reported are depicted below the horizontal axis. References where a Digital Annealer was used, which emulates a quantum algorithm using bits, are depicted on top and are marked with a star

Bożejko et al. [21, 28]: **Type:** SMSP, **Constraints:** D, S, **Objective:** Sum of tardiness costs, **Solver:** Hybrid, **Size:** Up to 8 jobs; up to 96 qubits.

They combine QA and a branch-and-bound-algorithm (cf. Potts and Van Wassenhove [172]). In the corresponding tree, in each lower node, one additional task position is fixed. In this sense, the SMSP corresponds to the task to find the path in the tree with minimal objective. By computing upper and lower bounds for the objective in each node in the tree, the branch-and-bound strategy is to always select the node with the smallest lower bound or smallest upper bound. These bounds are computed using QA, by formulating

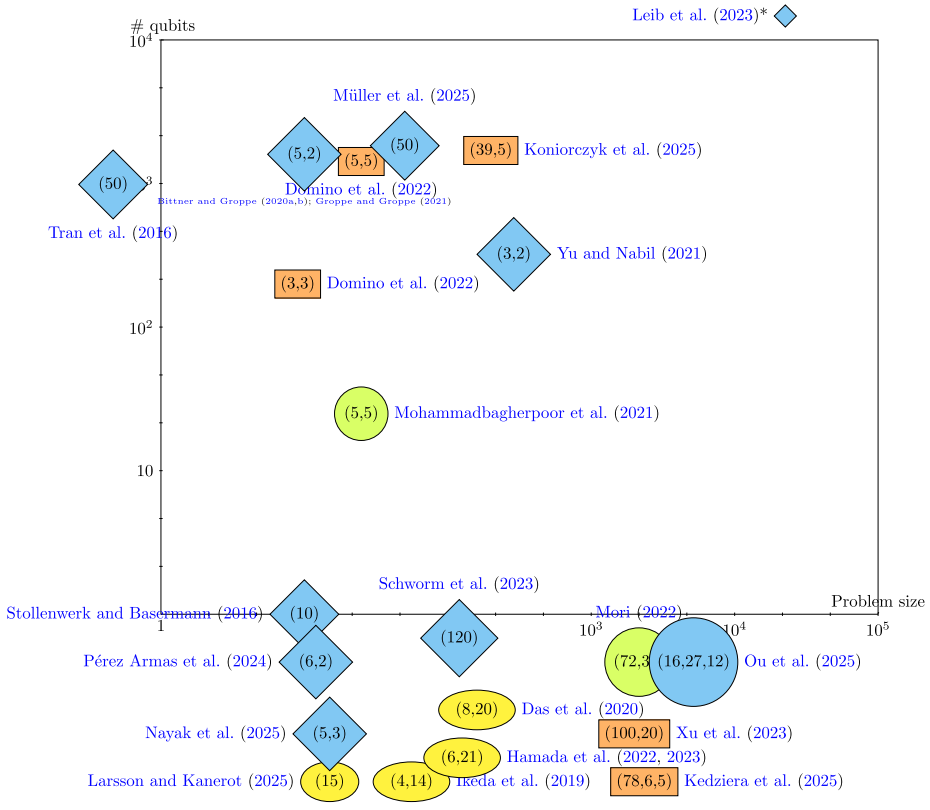

Figure 4 Overview of other types of scheduling problems that are formulated as a QUBO problem or CQM. The horizontal axis refers to the size of the problem, quantified as the number of trains times the number of stations for railway scheduling problems, as the number of aircrafts or gates times the number of flights for gate/flight scheduling problems, as the number of passengers, facilities and time slots in passenger scheduling problems, and as the number of persons times the number of shifts for shift scheduling problems. The vectors in the nodes show the largest problem instances (e.g., (number of trains, number of stations)) addressed in the respective paper. For some other types of scheduling problems, only a number n of tasks or variables is given, therefore, only one number appears. The vertical axis denotes the number of qubits used. The location of the node, with respect to its center, is determined by the largest combination of problem size and number of qubits presented in the corresponding reference. A rectangle refers to a railway scheduling problem, a circle to a flight/gate/passenger scheduling problem, an ellipse to a shift scheduling problem, and a diamond to another type of scheduling problem. References where the number of qubits was not clearly reported are depicted below the horizontal axis. References where a Digital Annealer was used, which emulates a quantum algorithm using bits, are depicted on top and are marked with a star

the corresponding optimization problem as a QUBO problem, which are solved using the hybrid CQM solver.

Bożejko et al. [27]: **Type:** SMSP, **Constraints:** D, S, **Objective:** Sum of tardiness costs, **Solver:** Hybrid, **Size:** Up to 8 jobs; up to 56 qubits.

Here, another divide-and-conquer method (see Bożejko et al. [21]) is proposed, where one considers k-subsets of the task set, and applies QA in order to find the order of execution of these k jobs. The problem is formalized as a BQM.

[2]The number of binary variables in the QUBO formulation is $nm + mn(n-1)/2$, which would be 13,950 for the $(30, 30)$-instances, and even 25,250 for the $(100, 5)$-instances. The exact number of qubits used is however not reported.

[3]They consider a scheduling problem for automated storage and retrieval, therefore, there are only transport jobs but no machines considered.

Bajracharya and Shakya [31]: **Type:** UMSP, **Constraints:** D, F, P, **Objective:** Makespan, **Solver:** QA (Leap), **Size:** Up to 30 jobs and 30 machines.

They consider an application in task scheduling for cloud computing. A comparison with PSO reveals that QA considerably outperforms PSO in terms of total runtime, while the QA solution also has a better quality than the PSO solution, which they explain with the dependence of the PSO on the position and velocity of the particles.

Botelho and Salehi [32]: Fixed interval scheduling problem, **Constraints:** A, M, **Objective:** Sum of weights of assigned jobs, **Solver:** QA (Advantage, Advantage2), **Size:** Up to 591 jobs and 12 machines, 1056 binary variables (up to 3020 qubits).

They consider a fixed interval scheduling problem, where the task is to assign jobs to machines where the jobs are accompanied with both weights as well as fixed starting and ending times and where the total weight of the selected jobs should be maximized. They consider an application in music arrangement with the goal to reduce a piece of music containing the total number of tracks to m tracks. This technique should reduce the number of required instruments while maintaining the main features of the song. First, melodic phrases of the song must be identified, which can be done using an iterative algorithm given in Botelho and Salehi [32, App. B], which also assigns a weight to each phrase, which quantify how much a phrase contributes to the characteristics of the song. Here, the number n of phrases is identified with the number of jobs and the number m is identified with the number of machines. They compare the performance of QA with simulated annealing (SA) and a hybrid annealer in terms of the achieved information entropy (which is computed from the weights) w.r.t. the annealing time. The classical and hybrid annealer always provide feasible solutions on a smaller problem instance with 41 phrases and 4 tracks, while having around 16% infeasible solutions on a larger problem instance with 591 phrases and 12 tracks. In three out of 12 QA configurations on the smaller instance, constraints are violated, while QA does not provide any feasible solution on the larger problem instance.

Orts et al. [154]: **Type:** UMSP, **Constraints:** D, P, switching delay, **Objective:** Makespan, **Solver:** QA (Leap), **Size:** Up to 7 jobs, 250 repetitions of each job, and 15 machines (at least 850 qubits).

The switching delay penalty occurs when the task a machine executes is changed. In their experiments, they test the impact of the problem size and the algorithm parameters, but do not compare their algorithm against a competitor.

Pokharel et al. [164]: **Type:** SMSP, **Constraints:** M, **Objective:** Feasibility, **Solver:** QA, **Size:** Up to 40 jobs; up to around 350 qubits.

They use the equivalence of scheduling problems and graph coloring problems, which are formalized as a QUBO problem. They generate artificial scheduling problems by generating families of Erdös-Renyi-graphs (cf. Erdős and Rényi [65]), and compare the performance of four QA hardwares, namely D-Wave Two, 2X, 2000Q and Advantage. They evaluate the time-to-solution TTS, which is given by the expected annealing time until a solution with 99% success probability has been found, which is given by $TTS = t\ln(1-0.99)/\ln(1-p)$, for the success rate p, given by the relative fraction of ground state

[4] They use a quantum simulator.

[5] They implement their procedure in Qiskit.

[6] Number of binary variables.

solutions among all anneals, and the annealing time t for a single annealing repetition. Advantage outperformed the other algorithms in terms of TTS, and enjoyed the largest improvement when optimizing the ferromagnetic coupling.

Sun et al. [186]: **Type**: IMSP, **Constraints:** A, M, P, S, **Objective:** Early delivery costs, late delivery costs, production costs, **Solver:** VQE, **Size:** 5 jobs, 2 machines; 5 qubits.

They apply differentiable quantum architecture search from Zhang et al. [238] in order to automatically design quantum circuits.

Windmann [216]: **Type:** IMSP, **Constraints:** P, **Objective:** Costs, **Solver:** QAOA, **Size:** Up to 20 jobs; up to 433 qubits.

They consider a job-scheduling problem in the context of an automated storage and retrieval system.

Bożejko et al. [35]: **Type:** SMSP, **Constraints:** Only I, **Objective:** Total weighted tardiness, **Solver:** Local neighborhood search, **Size:** Up to 20 jobs.

They formulate the problem as a QUBO problem and represent the neighborhood search as a CQM and a BQM, which are solved using the LeapHybridCQMSampler. A comparison of the solution quality reveals that for larger problem instances, the hybrid approach yields sub-optimal solutions.

4.1.2 Multi-stage job scheduling problems

Venturelli et al. [212]: **Type:** JSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** QA (Two), **Size:** Up to 8 jobs, 8 machines, 1 operation; up to 400 qubits.

For comparison, the problems are solved via classical branching algorithms, which outperform the QA solver in terms of computational time.

Tomasiewicz et al. [207]: **Type:** WSP, **Constraints:** D, F **Objective:** Overall costs, **Solver:** QA (2000Q), **Size:** Up to 3 tasks and 6 machines; up to 429 qubits.

The QA solutions are evaluated by computing the global optimum with a classical solver and checking whether the quantum solutions attained this optimum. For 8 binary variables, all solutions were feasible and the global optimum was attained, for 15 binary variables, the global optimum was not found and only around 3% of the quantum solutions were feasible, while the QA completely fails on the problem with 18 variables in the sense that the global optimum was not found, however, it was also not found by Gurobi.

Denkena et al. [57]: **Type:** FJSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** Fujitsu Digital Annealer, **Size:** Up to 20 jobs and 15 machines and 15 operations (MK10).

They mention pit-falls when discretizing time-related variables like makespan and execution times, such as inaccuracies due to rounding ratios, and suggest to split the problem into sub-problems once the size becomes large. For the splitting, an iterative procedure is suggested, where one starts with an initial schedule by removing some operations, checking which jobs are, due to the omission of necessary operations, unfinished, and computes a schedule for these jobs, which is merged with the current one, until all jobs are finished. They compare the achieved makespans on several benchmark FJSSPs from the literature, where classical algorithms have been used. More precisely, they collect references where these problems have been solved, so that the achieves makespans are available. Then, they consider the median of the achieved makespans for each problem and the optimal makespan on these problems from the literature, and compare them with the achieved makespan of their digital annealer. On 6 out of 10 problems, the quantum solution was better than the median, and on 2 problems (MK03, MK08), the optimal makespan from

the literature was attained. When comparing the pure annealing time with the computation times of the classical algorithms, the former was lower than 30 seconds, which is often considerably faster than the computation times reported from the literature when applying classical algorithms. As for the number of bits (not qubits because the Fujitsu Digital Annealer works with bits), they prune the binary variables according to logical restrictions such as the fact that not every machine can handle each operation, and set the number of time steps so that the number of binary variables does not exceed the maximum available number of 8192 bits, however, it is not reported whether this limit is attained.

Plewa et al. [169]: **Type:** WSP, **Constraints:** A, F, **Objective:** Overall costs, **Solver:** QAOA, VQE, **Size:** Up to 4 tasks; up to 15 qubits.

They study the performance of QAOA and VQE for different encoding schemes for the discrete variables, namely, one-hot encoding, binary encoding, and domain-wall encoding.

Shimada et al. [194]: **Type:** Extended JSSP, **Constraints:** A, F, M, interval constraints, worker constraints, **Objective:** Makespan, **Solver:** Fujitsu Digital Annealer, **Size:** Up to 80 jobs, 12 workers, 13 machines, 81 time slots.

The standard JSSP is extended by the inclusion of workers that work at the machines. The constraints M include here the necessity to assign a worker to a machine who is capable to operate it. The resulting worker constraints are of the form that each worker can only work at one machine simultaneously, while the interval constraints indicate that the machines and workers must work within pre-scribed intervals. They point out that makespan minimization via QUBO formalization is difficult, and therefore propose a decomposition method, decomposing the JSSP with makespan objective into a master problem and a subproblem, which are solved alternatingly. The main problem with makespan minimization via QUBO is that makespan is not a binary variable, hence a time-indexed formulation is required, where the makespan is either guessed by a pseudo-makespan, penalizing the situation that a job is finished after this threshold, or it is represented by a dummy job, penalizing all jobs that are finished later than the dummy job. Alternatively, one computes the total sum of all ending times, approximating the makespan and leading to a minimization of all ending times. All strategies however only approximate the makespan. The decomposition now considers a makespan estimation problem as master problem, and the constraint satisfaction problem with the estimated makespan as subproblem. If the constraints are satisfied, the makespan is reduced and the sub-problem is tried to be solved again. The sub-problem is formulated as a QUBO problem here. They compare the performance of their approach with that of the Digital Annealer on the standard QUBO formalization with makespan approximation and with Gurobi, applied to a mixed-integer program (MIP) formalization of the JSSP. It turns out that their approach achieves lower makespans than with makespan approximation. For the larger example, the optimal solution was not attained using the Digital Annealer, in contrast to the application of Gurobi. However, the computation time (end-to-end, beginning after the input data were read) was much lower for their decomposition approach than for the Gurobi and the makespan approximation methods. They explain the long computation time of Gurobi by the internal data preparation, in particular, representing the constraints. The number of bits they use is not precisely stated, but they mention that the Digital Annealer can emulate up to 1 million bits.

Carugno et al. [39]: **Type:** JSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** QA, reverse annealing (RA) (Advantage, 2000Q), **Size:** Up to 26 jobs, 26 machines, 26 operations per job; up to 4400 qubits.

They compare the quantum solvers with classical solvers, which are the steepest descent gradient solver, Tabu Search, and SA. On the realistic problems, SA performs best, followed by RA, in terms of best solution energy. QA on D-Wave Advantage outperforms QA on D-Wave 2000Q, which they explain by the better qubit connectivity of Advantage.

Geitz et al. [79]: **Type:** Extended JSSP, **Constraints:** D, F, P, transportation constraints, **Objective:** Makespan, **Solver:** Leap Hybrid v2, Fujitsu Digital Annealer, **Size:** Up to 4 jobs, 4 machines, 24 operations, and 2 vehicles; up to around 2800 qubits.

They consider an extended job shop scheduling problem where workpieces are transported by autonomous ground vehicles in a factory and formulate it as a QUBO problem. Several transportation constraints arise, e.g., that there must be enough time for the transport to the next machine, that the task must be finished before the object is transported, or that the next production step can only start after the transportation is finished. They compare the performance of the quantum solvers with that of Tabu Search and constraint-based optimization (applied to the original problem, not the QUBO problem). The best results, both in terms of computation time and makespan, are achieved with constraint-based optimization.

Pakhomchik et al. [173]: **Type:** WSP, **Constraints:** P, R, **Objective:** Makespan, **Solver:** QuEnc solver (Terra Quantum), QA (D-Wave HSS), Hybrid QuEnc, **Size:** Up to 15 jobs; up to 2206 qubits.

They consider an application in quality control. Due to the problem size, they decompose the QUBO problem into smaller sub-problems, corresponding to the hierarchical structure of the tests performed by the quality control. This is represented by a tree, implying the conditions that children nodes can only start after the jobs corresponding to parent nodes are completed. They point out the disadvantage that this method may produce sub-optimal solutions. Their proposed hybrid QuEnc is the application of QuEnc to the sub-problems into which their algorithm decomposes the original QUBO problem. They compare the performance with several classical algorithms, which are a greedy algorithm, SA, and CPLEX (both applied to the QUBO problem and to the original constrained linear program). On smaller problems, CPLEX lead to the best results in terms of the normalized cost function, while for larger problems, due to a restriction in computational time, CPLEX does not find the optimal solution, in contrast to the Hybrid QuEnc algorithm, while the pure quantum algorithms are not convincing.

Shinjo et al. [189]: **Type:** FSSP, **Constraints:** D, P, changeover, **Objective:** Sum of total completion times, changeover number, **Solver:** Fujitsu Digital Annealer, **Size:** 10 jobs, 5 machines/operations.

They consider changeover, which indicates that machines must be modified when changing the particular task, e.g., for a coloring task, the machine may must be cleaned and filled with a different color. They propose a QUBO model based on a Petri net representation (e.g., Murata [142]) of the problem. Since a Petri net is colored and timed, they first need to unfold the net and then convert it into a QUBO problem.

Zhang et al. [235]: **Type:** FSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** Hybrid, **Size:** Up to 50 jobs and 50 machines/operations (Taillard).

They first formulate the FSSP as a disjunctive QUBO problem and as a time-indexed QUBO problem. They point out that in the literature, one often can only solve very small QUBO problems where the classical approaches outperform the quantum approaches. Therefore, a local neighborhood search strategy is proposed, where multiple one-machine problems and a single multi-machine problem are solved alternatingly. More precisely, the one-machine problems are rank-based, i.e., one has binary search variables $x_{j,q}$, which are 1 if job j has rank q on the machine. The objective of such a problem is given by

$$\sum_{q=1}^{n/2}\sum_{j=1}^{n} x_{j,q} Pos_j (n-q) d_{max} + \sum_{q=1}^{n}\sum_{j=1}^{n} (E_j - T_j)(n-q),$$

where n is the number of jobs, Pos_j the precedence position of the operation j in the job (i.e., if job i has the operations $O_i^{(1)}, \ldots, O_i^{(K_i)}$, then $Pos_{K_i} = K_i$), d_{max} is the maximum time an operation requires, E_j the earliest starting time for operation j, and where T_j is the tail, i.e., the time from the end of the operation to some end point. The number $n/2$ in the first sum is tunable. The first term therefore encourages the Pos_j to be small and the rank q to be small or large simultaneously, i.e., early operations in a job should be executed early (as starting an early operation late would be prone to a larger makespan as all remaining operations of this job need still to be executed). The second term encourages that q and $(E_j - T_j)$ are small or large simultaneously, i.e., if a job can only start late (E_j large and therefore $E_j - T_j$ small), the rank of the operations should be small in order to start the job soon after E_j, since a delay could again lead to a larger makespan. The solutions, i.e., the ranks of the jobs on each machine, enter the multi-machine problem as constraints. The multi-machine problem itself is formulated as constraint-programming problem. If the single-machine problem solutions let some job j_1 be ranked before j_2 (with a specified margin which enters as parameter that essentially defines the size of the neighborhood), this should also be respected for the solution of the multi-machine problem. The alternate solution of both types of problems make the overall approach of Zhang et al. [235] hybrid, where the constraint-programming problem is solved classically, using Gurobi and the IBM ILOG CP Optimizer. The single-machine QUBO problems are solved using the Fujitsu Digital Annealer. Their approach can compete even with standard constraint programming on large instances sizes both in terms of total computation time and mean relative error to the optimum objective.

Aggoune and Deleplanque [4]: **Type:** JSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** QA, **Size:** 4 jobs, 4 operations, 4 machines.

They solve the problem via a D-Wave hybrid solver.

Baioletti et al. [26]: **Type:** FOSSP, **Constraints:** D, F, **Objective:** Makespan, **Solver:** QAOA, **Size:** No experiments.

They restrict themselves to the problem formulation and QAOA concretization, but do not perform experiments.

Kurowski et al. [113]: **Type:** JSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** QAOA, **Size:** 3 jobs, 3 machines, 2 operations; < 100 qubits.

They solved the problem in a quantum simulator.

Schworm et al. [196]: **Type:** Dynamic JSSP, **Constraints:** M, P, S, **Objective:** Makespan, **Solver:** Hybrid BQM solver, **Size:** Up to 20 jobs, 10 machines, 10 operations (MK08).

They consider a dynamic JSSP, where machines can break down and where new jobs can arrive. They argue that solving static JSSP problems is not suitable in order to react to such dynamic settings. They propose to randomly introduce machine unavailabilities

or new jobs and to re-formulate the BQM, as which the problem is formulated, according to the new situation each time a monitoring system detects changes in the environment. As competitor, they use a solver composed by Tabu search and SA, i.e., without the QA component of the hybrid BQM solver. The hybrid BQM solver slightly outperforms the classical solver in terms of makespan, while there is no clear trend in terms of total computation time (the sum of CPU, QPU, and transmission times).

Aggoune and Deleplanque [5]: **Type:** JSSP, **Constraints:** D, M, P, unavailability constraints, **Objective:** Makespan, **Solver:** QA (Advantage), **Size:** 3 jobs, 4 machines, 7 operations; 208 qubits.

They introduce unavailability periods, which are represented by virtual jobs that block the machine during the respective time interval.

Bourreau et al. [16]: **Type:** Indirect JSSP, **Constraints:** A, M, P, **Objective:** Makespan, **Solver:** QAOA, **Size:** 4 jobs, 4 operations, 4 machines.

The considered problem is called indirect JSSP, which indicates that an indirect representation of the solution based on the Bierwirth vector. Such a vector, going back to Bierwirth [20], describes a permutation, which can be mapped into a directed acyclic graph, depicting the schedule. The parameters for QAOA are computed using a GA. The algorithm is implemented in Qiskit.

Farhani et al. [71]: **FSSP**, **Constraints:** D, **Objective:** Makespan, **Solver:** QA (Leap), **Size:** Up to 50 jobs, 20 machines and 20 operations (Taillard).

They formulate the problem as a QUBO problem where the decision variables include waiting times of jobs, idle times of machines and the position of the jobs in the job sequence. In addition, they propose an approximate formulation as traveling salesman problem (TSP). In the TSP, the jobs represent the cities. The makespan is replaced by the sum of the distances between all visited cities. They use different approaches in order to approximate these distances in terms of the individual processing times, proposed by Goh et al. [74]. The approximate TSP is also translated into a QUBO problem.

Schmid et al. [182]: **Type**: JSSP, FJSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** Filtering VQE, **Size:** Random; expected values are up to 4 operations, 4 jobs, 4 machines; up to around 650 qubits (JSSP) resp. around 250 qubits (FJSSP).

They propose an encoding scheme which allows for a reduction of the number of required bit-strings of a factor of at least $N/\log_2(N)$, for a problem with N jobs and N operations. They point out that the time-indexed representation of the problem scales badly. Their idea is to not use a quantum representation of the cost function. Their approach is based on encoding the constraints directly with the representation of the problem. This is done via an operation sequencing list, which contains elements $(O_{\pi(j)}, M_{\pi(j),j})$, where π is a permutation of the operations O_j and where $M_{j,i}$ are the machines on which job j can be executed. They show how to compute makespan from such a list and argue that the sequencing list already implicitly contains some constraints, such as that there must not be idle times of the machines. In other words, by only computing an ordering of the jobs and the assignment to the respective machines, one gets rids of time-related constraints and variables. The only constraint family that can still be violated is P. Therefore, they suggest to encode the number of valid schedules, which for the total number N of operations and sets J_k of jobs, where each J_k contains some operations, is given by $m := N!/(\prod_k |J_k|!)$, using $\lceil \log_2(m) \rceil$ qubits, which allows to discard invalid schedules by mapping schedules to inversion vectors. They further discuss how to reduce the number of required qubits

for a time-indexed formulation, since, due to the constraints P, an operation cannot start until its predecessors are finished, allowing to delete time indices and, therefore, qubits.

Toma et al. [208]: **Type:** Distributed FJSSP, **Constraints:** A, D, **Objective:** Makespan, **Solver:** QA (AdvantageSystem4.1 Annealer), **Size:** Up to 2 jobs/machines/operations (up to 250 binary variables); up to 4067 qubits.

The distributed FJSSP indicates that machines are located at different places. Therefore, transportation time and costs between jobs need to be included. They consider an application from the wool industry. The QA solutions are compared with the SA solutions. The best solutions, quantified by low-energy states of the corresponding Hamiltonian, from SA are at least equal to those of the QA solutions, and often considerably better. The computation time (where the QPU time for QA and the CPU time for SA are compared) is better for SA than for QA for problems with up to 300 variables, and worse for larger problems.

Wei et al. [215]: **Type:** WSP, **Constraints:** M, R, **Objective:** Transmission, training, aggregation and transmission cost, **Solver:** Hybrid, **Size:** Up to 9 servers, 2 models, 3 workers per model.

They consider federated learning and aim at optimizing the assignment of the individual models to the parameter servers. As the learning rate also needs to be adjusted, the formulated problem is a mixed-integer non-linear problem (MINLP). First, they linearize the problem by introducing auxiliary variables representing products of the original variables. Then, the Benders decomposition (Benders [15]) is applied, resulting in a continuous dual subproblem and a master problem. The master problem involved only binary variables, except for the real-valued pricing variable, which is approximated by binary variables. Therefore, the master problem can be converted into a QUBO problem, which is solved using QA. They compare the performance of this hybrid approach with a pure classical approach where classical solvers are applied for both the subproblem and the Master problem. Both approaches lead to similar values of the master problem. The computational time, measured as the accessing time of the QPU and CPU solver, respectively, without any overhead time (e.g., transmission time of parameters) is, in average, lower for the hybrid algorithm.

Baioletti et al. [17]: **Type:** FSSP, **Constraints:** P, **Objective:** Makespan, **Solver:** VQE, **Size:** Up to 7 jobs and 5 machines/operations (up to 13 qubits).

They use Lehmer's coding (Lehmer [120]) in order to represent an FSSP solution with n jobs with $\mathcal{O}(n \ln(n))$ qubits. On most problem instances, the VQE procedure achieved the optimal makespan. On some the larger instances however, the optimal solution is never achieved.

Bożejko et al. [34]: **Type:** FSSP, **Constraints:** A, D, **Objective:** Total delay costs, **Solver:** QA (Leap, Advantage), **Size:** Up to 30 jobs, 2 machines and 2 operations.

They integrate the scheduling problem into a LLM framework where an AI prepares the problem, selects suitable solvers, and returns the solution. A comparison between Leap and Advantage shows that the larger instances are not embeddable into Advantage, and that the computation time is much larger for the Advantage embedding than for the Leap CQM solver. A comparison between the Leap CQM solver, SA and Tabu Search reveals that, for smaller instances, the classical algorithms require less computation time, while the CQM solver and Tabu Search achieve the same solution quality and that of SA is worse. For larger instances, the CQM solver beats Tabu Search and SA in computation time, for the price of a solution quality lying between that of Tabu Search and SA.

Farhani et al. [67]: **Type:** FSSP, **Constraints:** A, M, **Objective:** Makespan, **Solver:** QA (Leap, InfinityQ), **Size:** Up to 50 jobs, 20 machines and 20 operations (Taillard).

They compare different QUBO problem formulations, that arise from software that automatically converts the original hard-constrained optimization problems into QUBO problems, and manually designed QUBO problems. As for the latter, they consider a standard version where the decision variables x_{jm} indicate whether job j is assigned to machine m, as well as an approximation of the FSSP as traveling salesman problem. They experimentally derive that the manually designed QUBO problems lead to better solutions in terms of makespan, but require more computation time for the solver than the automatically generated QUBO problems.

Fu et al. [70]: **Type:** FJSSP, **Constraints:** A, D, M, **Objective:** Makespan, **Solver:** Coherent Ising Machine, **Size:** Up to 3 jobs, 5 machines, 9 operations, up to 126 qubits.

The largest considered problem would initially require 1200 qubits, but they prune qubits which, due to the constraints, are already enforced to be zero, resulting in 126 qubits. They compare the performance of Gurobi, SA, applied to the MIP formulation, and tabu search, applied to the QUBO problem, with their approach. The quantum approach significantly outperforms the competitor methods in terms of computational time, while all methods result in the same makespan.

Tirado-Domínguez et al. [202]: **Type:** WSP, **Constraints:** D, F, M, **Objective:** Feasibility, **Solver:** Modified QAOA (AER), **Size:** 3 tasks.

They argue that the QAOA ansatz requires a complete computation of the Ising Hamiltonian. Therefore, they decompose the Hamiltonian into a sum of two Hamiltonians, where one corresponds to the objective function and a subset of constraints while the second one corresponds to the remaining constraints, allowing for distinct parameter sets for both Hamiltonians. They compare the performance of their QAOA variant with that of standard QAOA and another QAOA variant, and conclude that it shows intermediate performance concerning normalized energy (standard QAOA being worst), while being comparable with standard QAOA in terms of execution time.

Slysz et al. [185]: **Type:** JSSP, **Constraints:** A, D, M, P, **Objective:** Makespan, **Solver:** Binary Bosonic Solver (ORCA PT-1), **Size:** 5 jobs, 2 machines and 6 operations (up to 30 binary variables).

They solve the problem using photonic quantum computing. For the larger problem instance with 30 binary variables, the solver finds feasible solutions, but not the optimal solution.

4.1.3 Railway/flight scheduling problems

Domino et al. [53]: **Type:** Railway scheduling, **Constraints:** D, M, rolling stock circulation, **Objective:** Sum of secondary delays, **Solver:** QA (2000Q), **Size:** 3 trains, 3 stations; 198 qubits.

They consider railway dispatching for single-track railway lines. The constraints D and M represent the conditions that the entrance and exit time of a train in a station must respect a minimal passing time through the station, and that a rail or a station can only be occupied by one or a fixed number of trains at the same time, respectively. On a smaller problem instance with 48 qubits, the optimal solution was achieved in many parameter settings concerning annealing time and coupling strength. However, one out of four cases for the larger problem instance was not embeddable in the topology of the 2000Q annealer, and the solutions in the other three cases were always infeasible.

Domino et al. [54]: **Type:** Railway scheduling, **Constraints:** D, M, safety, minimal stopping time, rolling stock circulation (cf. Domino et al. [53]), **Objective:** Weighted sum of delays relative to maximum acceptable delay, **Solver:** QA (Advantage), hybrid BQM solver, **Size:** Up to 5 trains and 5 stations; up to 1419 qubits.

They propose a higher-order binary optimization (HOBO) problem formulation. However, as QA cannot be applied to a HOBO problem, they eventually recast the problem as a QUBO problem by introducing auxiliary variables representing terms of cubic degree or higher. While the hybrid BQM solver solution comes close to the optimal solution in terms of energy on the larger problem instance, QA provides clearly sub-optimal solutions.

Mohammadbagherpoor et al. [134]: **Type:** Airline gate-scheduling, **Constraints:** M, unique assignment, **Objective:** Total walking distance of the passengers between the gates, **Solver:** VQE, **Size:** Up to 5 flights and 5 gates; up to 25 qubits.

They point out that further possible restrictions can be gate closing constraints or alleyway constraints so that planes at close gates should not depart with a small time gap.

Mori [139]: **Type:** Flight scheduling, **Constraints:** Assignment constraints, delay constraints, swap constraints, pairing constraints, **Objective:** Total operational costs, **Solver:** LeapHybridSampler, **Size:** 2160 variables.

The assignment constraints indicate that exactly one aircraft must execute a given flight. The delay penalties discourage that the delay for a given flight affects other flights, while the swap penalties consider an already given original partial schedule and penalize that a different than originally planned aircraft is assigned to a flight. The pairing constraints consider that two flights are assigned to the same aircraft without the end point of the one and the starting point of the other flight coinciding. In their experiments, they compare the performance of the quantum algorithm with a classical particle swarm optimization (PSO) and CPLEX, but only in terms of execution times for some selected flights, were the quantum algorithm has a tendency to provide better solutions.

Xu et al. [222]: **Type:** Train timetabling, **Constraints:** Allowed number of stops for each train and station, stopping time constraints, flow balancing constraints, delay constraints, safety constraints, maintenance gap constraint, **Objective:** Total travel time costs, total departure delay, number of stops, **Solver:** Coherent Ising Machine, **Size:** Up to 100 trains, 20 stations.

The flow balancing constraints prescribe that the selected arc between origin and destination must be unique, continuous and interruption-free. Among others, QUBO problem formulations are considered. The safety constraints indicate a minimum time interval between the departure of one and the arrival of the next train in a station, as well as a minimum time interval between two trains running on the same arc. They compare the quantum algorithm with SA, concluding that the optimal solution is only attained for the problem with 15 trains for both algorithms, while for larger problem instances, the algorithms end up in a sub-optimal solution. For each problem instance, the computation time of the algorithms is very similar and the solution quality is also comparable.

Kedziera et al. [108]: **Type:** Railway scheduling, **Constraints:** Continuity, capacity, demand, crew, **Objective:** Operational costs, number of deployed units, **Solver:** QA (Advantage), VeloxQ, **Size:** Up to 78 trips, 6 depots and 5 unit types.

They consider scheduling the daily rolling stock circulation on a railway. The continuity constraints balance the incoming and outcoming units for each node. The crew constraints respect driver availability. They compare the performance of QA and the VeloxQ

solver with the classical SCIP solver, applied on an ILP formulation of the problem. The classical solver outperformed the quantum approaches in terms of solution quality, and achieves lower computation time than QA.

Ou et al. [153]: **Type:** Cruise passenger itinerary scheduling, **Constraints:** D, M, R, start and end point, **Objective:** Total travel distance, **Solver:** DA (Compal Quantix), **Size:** Up to 16 passengers, 12 time slots, 27 facilities.

In the passenger scheduling problem, the constraints M indicate that there are mandatory facilities that must be visited. The constraints R indicate that each passenger can chose only a limited number of facilities and that in each facility, there can only be a limited number of passengers simultaneously. The start and end point constraints enforce that each passenger's schedule starts at the starting facility and ends at the chosen final facility. They compare the DA with a greedy algorithm, being aware that the greedy algorithm is prone to only find a local optimum. Across all instances, the solution of DA is better, but DA requires a significantly higher computation time by several magnitudes. This time however is nearly constant across the problem instances, while the greedy algorithm requires more time on larger instances.

4.1.4 Shift scheduling problems

Ikeda et al. [95]: **Type:** Nurse scheduling problem, **Constraints:** Allowed number of breaks, time between two days of duty, number of nurses per shift, **Objective:** Feasibility, **Solver:** QA, reverse annealing (RA) (2000Q), **Size:** Up to 4 nurses and 14 shifts.

This is a particular scheduling problem which is unrelated to job scheduling problems, hence it invokes the specific constraints listed above.

Das et al. [63]: **Type:** Nurse scheduling problem, physician scheduling problem, combined nurse-physician scheduling problem, **Constraints:** Maximum number of shifts per person and time unit, maximum number of persons per shift, maximum number of consecutive shifts per person, day off, **Objective:** Feasibility, **Solver:** Leap v2, QA, RA (2000Q) **Size:** Up to 6 nurses, 2 physicians and 20 shifts; up to 160 binary variables.

They compare the results of the quantum approach with those of simulated annealing (SA) in terms of the achieved energy. RA outperforms QA, while the performance of SA is often between the performances of RA and QA.

Hamada et al. [90, 91]: **Type:** Shift scheduling problem, **Constraints:** Desired workload constraints, group assignment constraints, **Objective:** Feasibility, **Solver:** QA (Advantage), **Size:** Up to 6 workers and 21 shifts.

They consider shift scheduling in call centers and compare the performance of QA with simulated QA (SQA), satisfiability modulo theory (SMT), Tabu search, SA, and the SCIP solver for MIPs. The solutions are compared in terms of energy and of the time-to-solution, which, for the execution time τ of one run, the probability r_1 of obtaining a feasible solution in a single annealing process, and for $r_m = 1 - (1 - r_1)^m$, is defined as $\tau \lceil \frac{\ln(1-r_m)}{\ln(1-r_1)} \rceil$. QA achieves the smallest time-to-solution, but the energy range is large. The lowest energy range is achieved by SA, being comparable with SQA and Tabu Search in terms of mean energy.

Larsson and Kanerot [121]: **Type:** Physician scheduling problem, **Constraints:** Demand, availability, preferences, fairness, breaks, number of shifts per week, **Objective:** Feasibility, **Solver:** QAOA (IBM Quantum, AER), **Size:** Up to 15 physicians (up to 98 binary variables).

The problem formulation encourages several constraints such as fairness, indicating that each physician should have the same number of shifts, or breaks, indicating that the cumulated number of shifts should be kept low. Compared with Gurobi, the quantum approach shows often a worse performance, while requiring dramatically more computation time (at least 2946 seconds, while Gurobi only required 0.1 seconds).

4.1.5 Other scheduling problems

Stollenwerk and Basermann [181]: **Type:** Satellite scheduling problem, **Constraints:** M, charge and memory constraints, **Objective:** Downlink, **Solver:** QA (2X), **Size:** Up to 10 binary variables (not clearly reported).

The satellites can either charge, downlink, or experiment. Charging is only possible when being exposed to sunlight, downlinking only when being near the ground station. The charge and memory constraints enforce the charge and the memory to be inside a given interval, respectively.

Tran et al. [203]: **Type:** Vertex coloring, task scheduling, airport runaway scheduling, **Constraints:** M, unique assignment; potentially P and separation constraints, **Objective:** Feasibility/makespan, **Solver:** QA (2X), **Size:** Up to 50 binary variables; up to 981 qubits.

They propose a hybrid quantum-classical guided tree search approach for job scheduling. The idea is to apply QA in different regions of the input space and to use the output for a tree-based search algorithm, while the problem is formulated as a QUBO problem. They consider three different scheduling problems, namely, vertex coloring (into which scheduling can be transformed, see Rieffel et al. [178]), task scheduling for a Mars Lander, and airport runaway scheduling. The unique assignment constraint, which corresponds to all three examples, enforces that each task must be assigned exactly to one resource. For the mars lander and airport runaway scheduling problem, constraints P become active. For the airport runaway problem, a separation between the departures must be respected as additional constraint. The objective for the airport runaway scheduling problem is to minimize the sum of the departure times, while for the other two problems, the objective is to find a feasible solution.

Bittner and Groppe [18, 19], Groppe and Groppe [73]: **Type:** Transaction scheduling, **Constraints:** D, blocking constraints, **Objective:** Weighted sum of execution times, **Solver:** QA (2000Q) (Bittner and Groppe [18, 19], Grover's search (Groppe and Groppe [73]), **Size:** Up to 5 transactions and 2 machines; up to around 1600 qubits.

They consider transaction scheduling in databases, which differs from job scheduling problems due to additional blocking constraints in the sense that certain transactions block other transactions during their execution.

Yu and Nabil [231]: **Type:** Charging scheduling problem of electric buses, **Constraints:** M, S, state of charge (SoC) constraints, **Objective:** Electricity costs, **Solver:** QA (2000Q), **Size:** Up to 320 qubits.[7]

They apply the Hubbard-Stratonovich transformation to a QUBO problem. This idea goes back to Ohzeki [152], who point out that the 2000Q solver from D-wave embeds the problem into a chimera graph, which is sparsely covered by artificial spins. Since the embeddable size of the chimera graph was low at this time, not allowing for fully-connected

[7] The 320 qubits are required in order to embed the transformed problem with 3 buses, 2 loading stations, and 48 time steps on the 16×16 chimera graph of the 2000Q annealer. They point out that the original QUBO would not be embeddable, and that it would require 2499 qubits on an 24×24 chimera graph.

Ising models. Therefore, they propose to apply the Hubbard-Stratonovich transformation from statistical mechanics, which translates the quadratic term in the Ising model to a linear term, and providing a dual Hamiltonian. In Yu and Nabil [231], this technique allows to include a penalty term of fourth order into the objective. The application of the transformation reduces the order of such a term to two. The constraints M and S are to be understood in the sense that a loading station can only charge one bus simultaneously and that the charging process must not be interrupted, respectively. The SoC constraints pre-scribe that the SoC must lie in a given interval for each bus. Experiments confirm that using the Hubbard-Stratonovich transform, larger problems can be solved, which would not be embeddable on the given hardware. They compare the results of the two solvers and derive that the time to a success rate of 99% is considerably lower for QA than for SA, while the energy of the dual Hamiltonian is generally larger for QA than for SA. They also derive experimentally that the number of iterations can be substantially reduced when using adaptive moment estimation (ADAM) instead of a fixed learning rate when updating the Lagrange multipliers of the dual Hamiltonian.

Leib et al. [127]: **Type:** Transport robot scheduling problem, **Constraints:** M, transportation constraints, no-delay constraints, time interval constraints, **Objective:** Sum of completion times, **Solver:** QA (AdvanceSystem4.1), Fujitsu digital annealer, Fujitsu digital annealer hybrid framework, **Size:** Up to 22,692 binary variables.

The constraints M not only indicate that a machine can only handle one sample simultaneously, but that also the transport robot can only carry one sample at the same time. The transportation constraints prescribe that the robot must pick up the sample from the rack or a machine, carry it to another machine or the rack and to drop it. The no-delay constraints indicate that a machine must immediately start the operation once the sample has been dropped, and the time interval constraints necessitate that a sample must be re-operated again after a certain time interval. The competing Gurobi solver is applied to a MIP-formulation of the problem, both in a sequence and a time-index formulation. The classical solvers outperform the quantum approaches at least in the quality of the solutions or in the running time. The running time has to be understood as end-to-end running time, i.e., from the submission of the problem to the solver until the return of the solution. The large instances with more than 10,000 variables are only solved using Gurobi and the Fujitsu Digital Annealer, hence they are not represented by qubits. The quantum solvers were used for instances with up to 8080 (binary) variables.

Pérez Armas et al. [155]: **Type:** Resource-constrained project scheduling, **Constraints:** D, P, R, **Objective:** Makespan, **Solver:** QA, RQA (Advantage 6.3), **Size:** Up to 6 tasks and 2 resources.

They review 12 different MILP formulations of such a problem from the literature and analyze the complexity of the corresponding QUBO problem formulation, measured in the number of qubits. Here, the qubits represent target variables and slack variables. The authors consider the most efficient QUBO formulation for their experiments. They compare the performance of the quantum algorithms with Random Sampling, SA, Gurobi, Coin-CBC, and GLPK. They conclude that QA and RQA is superior than CBC and GLPK in terms of time-to-target (i.e., the runtime required in order to obtain a solution whose energy is lower than some specified energy threshold) and solution energy. Although inferior to Gurobi on high energy quantiles, it outperforms Gurobi on low energy quantiles.

Jayaraman et al. [98]: **Type:** Task scheduling in cloud computing, **Constraints:** None, **Objective:** Energy consumption or makespan, **Solver:** QAOA, VQE, **Size:** Not clearly reported.

They apply QAOA in order to find task-to-machine assignments, while VQE optimizes the parameters of the ansatz. In a comparison with a GA, an ant colony optimization algorithm and a PSO, the QAOA-VQE approach it best in terms of objective values, convergence speed, resource utilization and solution accuracy, however, the instance size is not clearly reported.

Müller et al. [135]: **Type:** Energy unit scheduling, **Constraints:** Demand, loading capacity, **Solver:** QA (Advantage), **Size:** Up tp 50 binary variables (up to 1837 qubits).

They first formulate the problem as standard QUBO problem where inequality constraints, arising from loading capacity bounds, are represented via equality constraints with slack variables. Alternatively, they use an unbalanced penalization approach where, for a constraint of the form $ax \leq b$, the indicator function $I(ax > b)$ is approximated with a smooth function, which is added to the objective. Due to this representation, the number of variables and therefore qubits is significantly reduced. They experimentally show that the unbalanced penalization approach leads to better solutions and a higher probability of finding the optimal solution, on average.

Nayak et al. [146]: **Type:** Transaction scheduling, **Constraints:** D, P, **Objective:** Maximum processing time, **Solver:** QAOA, **Size:** Up to 5 transactions and 3 machines (up to 68 binary variables).

They propose to enhance the QAOA solver with a locking mechanism, where several transactions are already fixed, resulting in a reduced QUBO problem.

4.2 Constrained quadratic model formulation

4.2.1 Single-stage job scheduling problems

Bożejko et al. [29]: **Type:** SMSP, **Constraints:** D, S, **Objective:** Sum of tardiness costs, **Solver:** Hybrid, **Size:** Up to 1000 jobs.

They propose a similar branch-and-bound strategy as Bożejko et al. [21], but the problem to find the bounds is directly formalized as a CQM. The experiments made in Bożejko et al. [29] compare the performance of their branch-and-bound strategy with that of a classical branch-and-bound strategy and with the optimal bounds determined by Gurobi. It turns out that both the total computation time and the QPU time are nearly constant over the problems when using the hybrid branch-and-bound algorithm, while growing exponentially when applying the classical algorithms. The relative error of the bounds computed by the proposed strategy and Gurobi is nearly 6% in one problem instance, while in other problem instances, the optimal solution has been retrieved. There is no clear tendency that the relative error grows with the problem size.

4.2.2 Multi-stage job scheduling problems

Bożejko et al. [23]: **Type:** Two-machine scheduling problem, **Constraints:** P, S, D, **Objective:** Gain for competing tasks in time, **Solver:** Hybrid, **Size:** No experiments.

They propose a similar branch-and-bound strategy as Bożejko et al. [21] did for the SMSP. They focus on calculating the lower bounds for the objective. This problem is formulated as a CQM, while calculating the upper bounds is considered as a constrained linear program which is solved using QA.

Schworm et al. [197]: **Type:** FJSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** Leap hybrid solvers, **Size:** Up to 30 jobs, 20 operations, 15 machines.

They provide BQM, CQM and discrete quadratic model (DQM) formulations of the problem and apply the corresponding Leap hybrid solvers. The hybrid CQM solver does not find a solution on problems with 8 jobs, 8 operations and 8 machines or larger due to exceeding the maximum allowed number of variables. The iterative classical hybrid solver significantly outperforms all other solvers in the computation time, while the achieved makespans are comparable with those of the other solvers.

Śmierzchalski et al. [191]: **Type:** JSSP, **Constraints:** D, M, P, minimal headway, deadlock, **Objective:** Weighted sum of completion times, **Solver:** QA, hybrid BQM solver, hybrid CQM solver, **Size:** Up to 21 vehicles and 7 zones; up to 1302 (sum of binary and integer) variables.

They consider the problem to schedule a fleet of automated vehicles in a factory, which is identified as JSSP in the sense that the vehicles represent the jobs and the locations where conflicts can happen represent the machines. Constraints D, M, and P are understood in the sense that there is a minimal passing time for the vehicles to subsequent zones and within the zones, that at most one vehicle can be located in a zone at a time, and that vehicles cannot overtake, respectively. They compare the performance of QA, the hybrid BQM and the hybrid CQM solver with CPLEX. The only competitive quantum solver is the hybrid CQM solver, as the relative number of infeasible solutions of BQM and QA is high. For small problem instances, CQM which achieves similar objective values as the CPLEX solver, while requiring more computational time. The total computation time remains nearly constant throughout all experiments, while for CPLEX, it becomes eventually much larger than for the hybrid CQM solver. However, for larger problem instances, the solutions of CQM have worse objective values than those of CPLEX.

Krellner et al. [104]: **Type:** JSSP, **Constraints:** A, D, I, P, **Objective:** Makespan, **Solver:** Hybrid, **Size:** Up to 22 jobs, 2 machines, 2 operations.

They include the delivery of the jobs from one machine to another in their formulation and formalize the problem as MILP as well as a CQM, including quadratic constraints. They first experimentally derive that the Leap Hybrid solver leads to better results when using the MILP formulation than the CQM formulation. Then, they compare the performance of Gurobi and the Leap Hybrid solver on the MILP model. On the MILP model, it turns out that Gurobi considerably outperforms the hybrid solver in terms of solution quality and that the gap increases with increasing runtime. They interpret this finding in the sense that Gurobi improves the solution with increasing runtime, which the hybrid solver does not necessarily. They also compute the time to solution, which is, on average, up to 27 times larger for the hybrid solver than for Gurobi. On the CQM model, the hybrid solver outperforms Gurobi (which does not solve the CQM directly but linearizes the quadratic constraints) on the majority of instances in terms of solution quality.

Kroczek et al. [116]: **Type:** WSP, **Constraints:** D, F, **Objective:** Overall costs, **Solver:** Hybrid, **Size:** Up to 32,715 binary variables.

They formulate the problem as a CQM and apply D-Wave's hybrid CQM solver. For selected problem instances, they report an increase in the achieved objective value of 1.2% up to 15.8% compared to the Gurobi solution, where the highest gap is achieved on the second-largest problem instance and the lowest gap on the largest problem instance.

4.2.3 Other scheduling problems

Glos et al. [81]: **Type:** Vehicle test scheduling problem, **Constraints:** M, group constraints, capacity constraints, **Objective:** Feasibility, **Solver:** Hybrid CQM solver, **Size:** 911 binary variables.

They formulate the optimization problem for scheduling tests for vehicles as a CQM and compare the performance of the CQM solver with Gurobi and CBC. In the given problem, the number of vehicles to be tested decreases over time, making later iterations cheaper. The average runtime is up to 2 magnitudes lower for Gurobi than for CBC and the CQM solver, where the runtime for CBC is larger than for CQM in early iterations, but lower in later iterations (less vehicles). The solution quality is comparable. For a problem where not one vehicle but a bunch of five vehicles has to be scheduled, the CQM solver was faster than Gurobi (around one magnitude), but the solution quality was worse, while CBC did not find a solution in the allowed time.

Koniorczyk et al. [110]: **Type:** Railway scheduling problem, **Constraints:** D, M, P, dwell time, safety, rolling stock circulation, **Objective:** Weighted sum of secondary delays, **Solver:** Hybrid CQM solver, **Size:** Up to 39 trains and 5 stations; up to around 1650 variables.

The constraints D and M have to be understood in the sense that there must be enough time to reach the next assigned station, and that single tracks, station tracks, and intersection areas can be occupied by at most one train at the same time, respectively. On synthetic experiments with up to around 1650 variables, CQM required more computational time than CPLEX in order to find optimal solutions. In the most challenging scenario, CQM could not find optimal, but only feasible solutions. When limiting the computational time of CPLEX to the time CQM required to find these feasible solutions, CPLEX also only found sub-optimal solutions of comparable quality to that of the CQM solutions. In realistic examples with up to 817 variables, the hybrid CQM solver only provides sub-optimal solutions for the larger instances, where the total computation time is comparable, while requiring much more time than CPLEX on the smaller instances with up to 711 variables.

Schworm et al. [198]: **Type:** Energy supply scheduling problem, **Constraints:** Capacity constraints, demand constraints, storage constraints, **Objective:** Energy costs, **Solver:** Hybrid CQM solver, **Size:** Up to 120 time steps.

They apply the hybrid CQM solver to the formulated problem for different use cases, but do not compare it against a competitor.

5 Evolutionary algorithms for scheduling problems

We first provide a graphical overview of the references where a scheduling problem was solved using a quantum-inspired genetic algorithm. More details about the respective references are given later in the cited sections.

In Fig. 5, Fig. 6, Fig. 7 and Fig. 8, the references from Sect. 5.1 and 5.2 where standard single-stage job scheduling problems and multi-stage job scheduling problems are considered, respectively.

[8] The iteration number of 226 is an average.

[9] The number 446 does refer to masks, not to operations. In the graphic, the node location is computed by considering the number of machines to be 56+446.

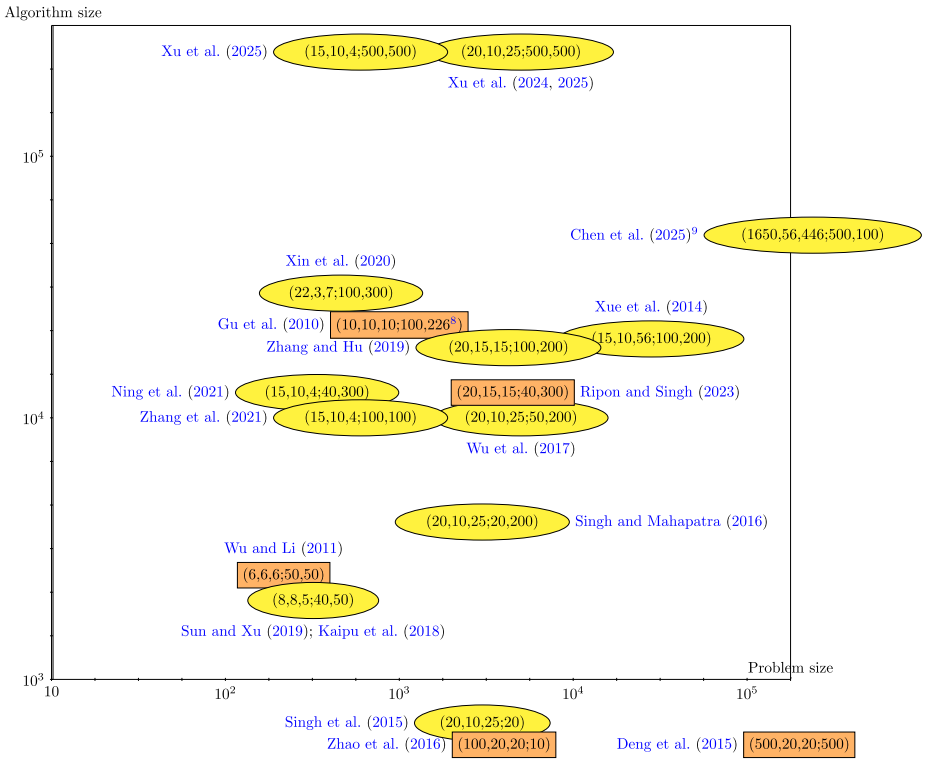

Figure 5 Overview of scheduling problems that are solved via a quantum genetic algorithm. The *x*-axis refers to the size of the problem, quantified as $n \times m \times O$, where O is the number of operations per job. The *y*-axis denotes the size of the algorithm, quantified as the population size times the number of generations. The largest combination (problem size, algorithm size) in the reference is always given by the mid-point of the node or the tip of the arrow, respectively. The vectors in the nodes show the largest problem instances addressed in the respective paper. A semicolon separates the problem size from the algorithm size. In some algorithms, there was no fixed number of generations but another stopping criterion such as convergence or a time limit. If no average or maximum generation number could be extracted, the references are depicted below the *x*-axis, and the number behind the semicolon denotes the population size. A rectangle refers to a JSSP, an ellipse to an FJSSP

5.1 Single-stage job scheduling problems

Niu et al. [149]: **Type:** Hybrid UMSP, **Constraints:** D, **Objective:** Total completion time, **Solver:** Quantum-inspired immune algorithm (QIIA), **Size:** Up to 40 jobs and 10 stages.

In a hybrid UMSP,[10] the factory consists of different stages and on at least one stage, there is more than one machine. Each job can be processed on any machine at each stage. Each individual consists of an $S \times n$-qubit matrix, where S is the number of stages and n the number of jobs. Each individual is converted into a job permutation. Their QIIA is compared with a classical immune algorithm (IA) and a QIGA as in Narayanan and Shmatikov [147]. QIIA outperforms its competitors on all problem instances. On selected problem instances, QIIA converges faster in terms of generations, but the computational times are not reported.

Fang [68]: **Type:** UMSP, **Constraints:** A, M, **Objective:** Makespan, total earliness, total tardiness, **Solver:** QIIA, **Size:** 15 jobs, 4 machines.

[10] They call the problem a hybrid FSSP, but the jobs do not seem to consist of operations, therefore, we categorize this paper as well as Niu et al. [150] as SMSP paper.

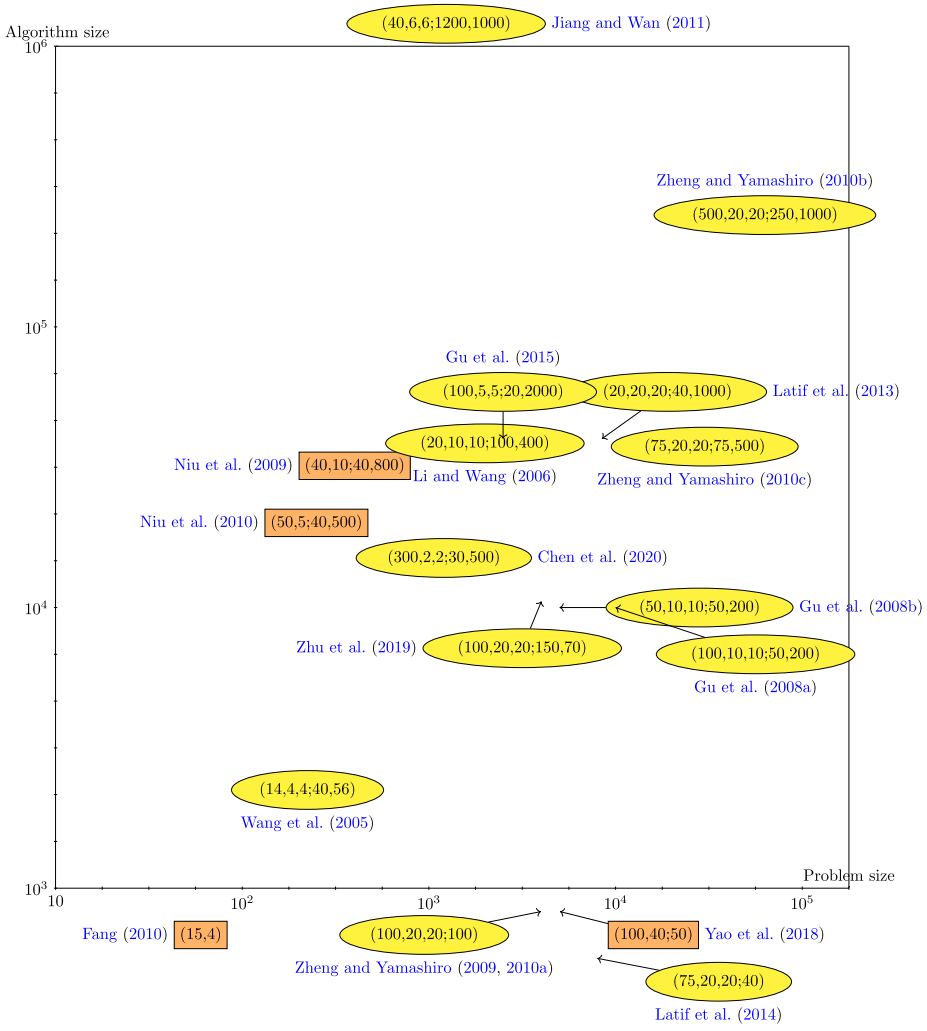

Figure 6 Overview of scheduling problems that are solved via a quantum genetic algorithm. The *x*-axis refers to the size of the problem, quantified as $n \times m \times O$, where O is the number of operations per job. The *y*-axis denotes the size of the algorithm, quantified as the population size times the number of generations. The largest combination (problem size, algorithm size) in the reference is always given by the mid-point of the node or the tip of the arrow, respectively. The vectors in the nodes show the largest problem instances addressed in the respective paper. A semicolon separates the problem size from the algorithm size. In some algorithms, there was no fixed number of generations but another stopping criterion such as convergence or a time limit. If no average or maximum generation number could be extracted, the references are depicted below the *x*-axis, and the number behind the semicolon denotes the population size. An ellipse to an FSSP, a rectangle to a single-stage scheduling problem

They consider an application in the context of textile industry. They compare their algorithm with a classical one and the QIGA-PGA from Li and Wang [129] (see Sect. 5.2) and achieve a better performance than both competitors in terms of objective and coverage.

Niu et al. [150]: **Type:** Hybrid UMSP, **Constraints:** M, **Objective:** Total completion time, **Solver:** QIGA, **Size:** Up to 50 jobs and 5 stages.

The individuals are converted to a job permutation at each stage, as in Niu et al. [149]. A comparison with a standard GA and a QIGA in the spirit of Narayanan and Shmatikov [147] reveals that the proposed algorithm outperforms its competitors in terms of average and best total completion time.

Yao et al. [232]: **Type:** UMSP, **Constraints:** D, blocking, **Objective:** Makespan, total carbon emissions, **Solver:** QIEA, **Size:** Up to 100 jobs and 40 machines.

The blocking constraints indicate that a finished job cannot leave the respective machine before the machine that executes the subsequent job is available. They compare their algorithm with classical GAs in terms of the number and the ratio of non-dominated solutions, stopping all three algorithms after the same running time. It turns out that their algorithm leads to better solutions in almost all problem instances.

5.2 Multi-stage job scheduling problems

Wang et al. [220, 221]: **Type:** FSSP, **Constraints:** D, **Objective:** Makespan, **Solver:** Quantum-inspired genetic algorithm (QIGA), **Size:** Up to 14 jobs and 4 machines/operations.

In Wang et al. [220], the QIGA is enhanced with permutation GA (PGA), which is also run and needs to be completed until the next iteration starts.

Li and Wang [129]: **Type:** FSSP, **Constraints:** M, **Objective:** Makespan, mean weighted completion time, maximum tardiness, mean weighted tardiness, maximum earliness, mean weighted earliness, maximum flow time, mean weighted flow time, number of tardy jobs, **Solver:** QIGA-PGA, **Size:** Up to 20 jobs and 10 machines/operations

They consider multiple objectives, which are formed into a single objective function using a randomly weighted sum. The idea is to let QIGA and PGA run in parallel, but to let the initial population in each iteration for PGA be composed by the union of the best half of the PGA solutions from the previous iterations and the population from QIGA from the previous iteration. They compare the performance of QIGA-PGA with pure PGA and pure QIGA. The results indicate that QIGA-PGA slightly outperforms PGA in terms of computation time and makespan, and considerably outperforms QIGA. QIGA-PGA is also competitive w.r.t. the other components of the objective function.

Gu et al. [77]: **Type:** FSSP, **Constraints:** D, E, F, **Objective:** Earliness, tardiness, **Solver:** QIGA, **Size:** Up to 100 jobs and 10 machines.

They apply QIGA and GA to the problem instances. Both algorithms lead to comparable objective values and makespans.

Gu et al. [78]: **Type:** Stochastic FSSP with breakdown, **Constraints:** D, S, **Objective:** Expected makespan, **Solver:** QIEA, **Size:** Up to 50 jobs and 10 machines/operations.

They consider random processing times and random machine breakdown, leading to unavailabilities of machines. Moreover, the processing time of a job in a machine as well as the repair time in the case of a breakdown are stochastic.

Gu et al. [75]: **Type:** Stochastic JSSP, **Constraints:** A, D, P, **Objective:** Expected makespan, **Solver:** Co-evolutionary QIGA, **Size:** Up to 10 jobs/machines/operations.

In their co-evolutionary algorithm, multiple populations are considered simultaneously. They propose three different competetive evolution schemes. A comparison with a GA and QIGA from Wang et al. [221] reveals that their algorithm often provides better expected makespans on the larger problems than the competetors, but for the price of often requiring much more computation time.

Zheng and Yamashiro [244, 245]: **Type:** FSSP, **Constraints:** M, **Objective:** Total flowtime, **Solver:** Quantum-inspired swarm evolutionary algorithm, **Size:** Up to 100 jobs and 20 machines/operations (Taillard).

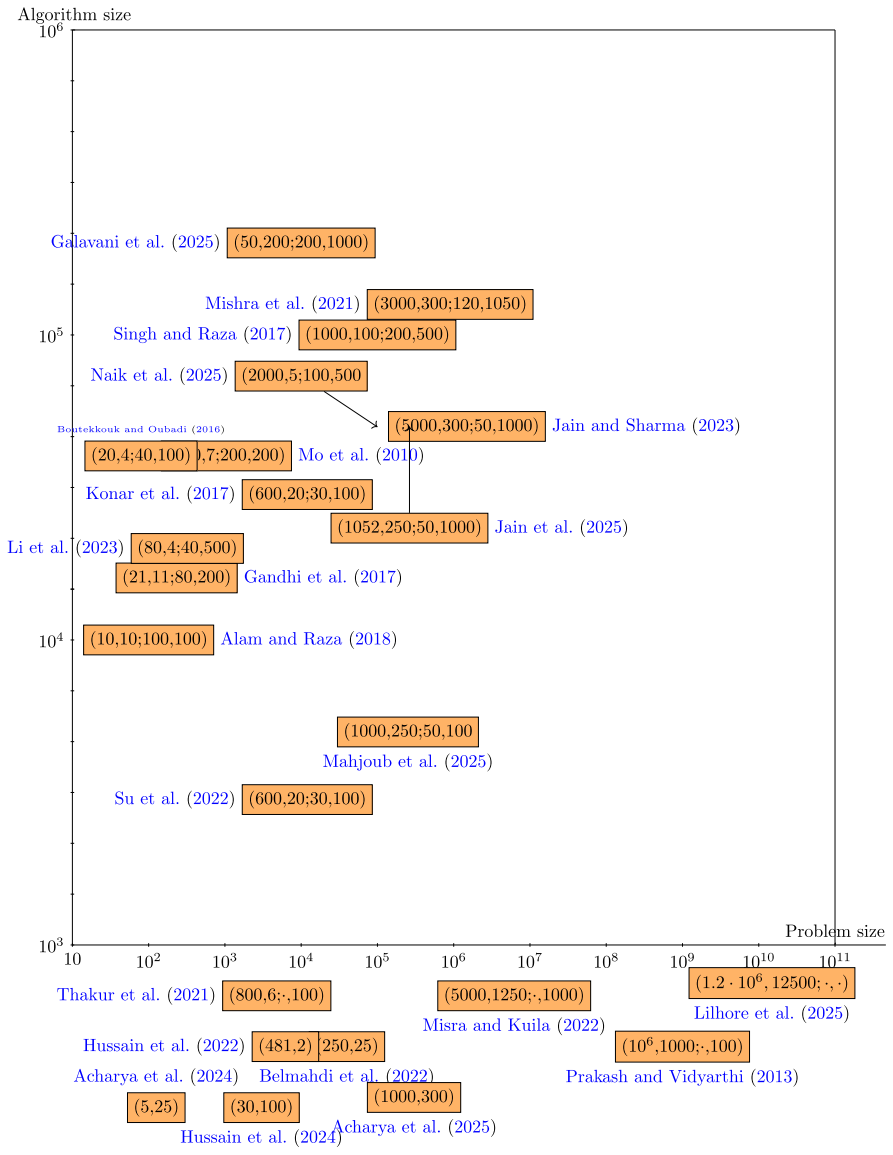

Figure 7 Overview of task scheduling problems that are solved via a quantum genetic algorithm. The *x*-axis refers to the size of the problem, quantified as the number of tasks times the number of processing nodes. The *y*-axis denotes the size of the algorithm, quantified as the population size times the number of generations. The largest combination (problem size, algorithm size) in the reference is always given by the mid-point of the node. The vectors in the nodes show the largest problem instances addressed in the respective paper. A semicolon separates the problem size from the algorithm size. In some algorithms, there was no fixed number of generations but another stopping criterion such as convergence or a time limit. If no average or maximum generation number could be extracted, the references are depicted below the *x*-axis, and the number behind the semicolon denotes the population size. If no population size but a generation number can be extracted, a · is depicted after the semicolon. A rectangle refers to a task scheduling problem, an ellipse to a railway scheduling problem, a circle to a maritime scheduling problem, a diamond to another type of scheduling problem

Their quantum-inspired swarm evolutionary algorithm makes PSO accessible to a population represented by qubits. They compare their algorithm with heuristic and classical evolutionary algorithms, resulting in an equal or better performance of their algorithm concerning total flowtime on all problem instances.

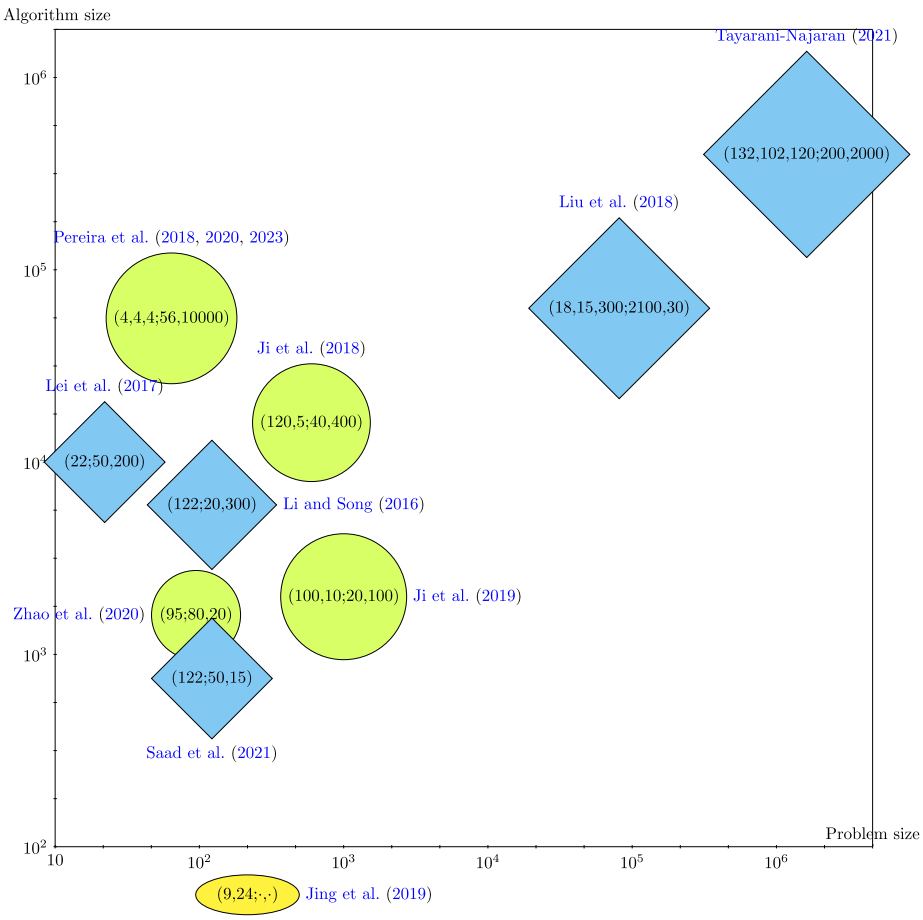

Figure 8 Overview of other scheduling problems that are solved via a quantum genetic algorithm. The *x*-axis refers to the size of the problems. The *y*-axis denotes the size of the algorithm, quantified as the population size times the number of generations. The largest combination (problem size, algorithm size) in the reference is always given by the mid-point of the node. The vectors in the nodes show the largest problem instances addressed in the respective paper. A semicolon separates the problem size from the algorithm size. In some algorithms, there was no fixed number of generations but another stopping criterion such as convergence or a time limit. If no average or maximum generation number could be extracted, the references are depicted below the *x*-axis, and the number behind the semicolon denotes the population size. If no population size but a generation number can be extracted, a · is depicted after the semicolon. An ellipse refers to a railway scheduling problem, a circle to a maritime scheduling problem, a diamond to another type of scheduling problem

Zheng and Yamashiro [246]: **Type:** FSSP, **Constraints:** M, **Objective:** Makespan, total flowtime, maximum lateness, **Solver:** Quantum-inspired differential evolutionary algorithm, **Size:** Up to 500 jobs and 20 machines/operations (Taillard).

They show how to execute differential evolution on the quantum population. Moreover, they propose a variable neighborhood search (VNS) in order to enhance the local search. They compare their algorithm and different VNS strategies with a plethora of other classical and hybrid evolutionary algorithms. Concerning the attained objective, their algorithm often provides the best results.

Zheng and Yamashiro [247]: **Type:** NW-FSSP, **Constraints:** M, S, **Objective:** Makespan, **Solver:** QIEA, **Size:** Up to 75 jobs and 20 machines/operations (rec37, rec39, rec41).

They compare their algorithm with classical evolutionary algorithms, local search strategies, and the QIEA from Zheng and Yamashiro [246]. In terms of a relative error measure w.r.t. the optimal makespan, their algorithm outperforms most of the classical algorithms, and is competitive to the QIEA from Zheng and Yamashiro [246].

Jiang and Wan [101]: **Type:** Parallel FSSP, **Constraints:** M, P, S, **Objective:** Makespan, **Solver:** QIEA, **Size:** Up to 40 jobs, 6 machines/operations, 5 flow shops.

In the parallel FSSP, there are multiple flow shops, so that one has to additionally decide in which flow shop a job should be done. They compare the performance of their algorithm with a genetic algorithm, concluding that the quantum algorithm slightly outperforms the GA in terms of optimal values, computation time and the relative number of optimal solutions found, in nearly all five problems considered.

Wu and Li [217]: **Type:** JSSP, **Constraints:** A, M, P, **Objective:** Makespan, **Solver:** QIEA, **Size:** 6 jobs/machines/operations.

They propose an elitist strategy, which ensures that the best candidates are not modified in the next iteration. Their QIEA is applied to the problem, but no comparison with other algorithms is made.

Latif et al. [132]: **Type:** FSSP, **Constraints:** M, **Objective:** Makespan, **Solver:** QIGA-EDA, **Size:** Up to 20 jobs and 20 machines/operations (Taillard).

They combine two types of evolutionary algorithms: QIGA and estimation of distribution (EDA). The main idea is to let both algorithms run in parallel and to compare the best candidate solutions of each algorithm after each iteration. If there are solutions from EDA among these best candidates, they are integrated into the QIGA population. Their algorithm outperforms QIGA, standard GA, and standard EDA in terms of fitness value and average percentage deviation from the observed best solution.

Latif et al. [133]: **Type:** FSSP, **Constraints:** M, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 75 jobs and 20 machines/operations (rec37, rec39, rec41).

They essentially combine the QIGA from Han and Kim [88] with VNS. The performance is compared with other evolutionary algorithms, with and without VNS. All algorithms with VNS generally outperform algorithms without VNS in terms of makespan, while requiring much more computation time. Among the algorithms with VNS, the proposed QIGA turns out to be competitive.

Davarzani and Akbarzadeh-T [48]: **Type:** FJSSP, **Constraints:** M, **Objective:** Makespan, total workload of most loaded machine, total workload of machines, **Solver:** QIIA, **Size:** Up to 20 jobs, 15 machines/operations (MK10).

A comparison with classical evolutionary algorithms reveals a better performance of the proposed QIIA in terms of the objective.

Xue et al. [228]: **Type:** FJSSP, **Constraints:** M, **Objective:** Makespan, total workload of most loaded machine, total workload of machines, **Solver:** QIIA, **Size:** Up to 15 jobs, 10 machines, up to 56 operations in total (Kacem5).

In their QIIA, the antibody is encoded by an integer sequence, representing the ordering of the operations. The proposed QIIA outperforms classical evolutionary algorithms in terms of the objective.

Deng et al. [62]: **Type:** NW-FSSP, **Constraints:** D, S, **Objective:** Makespan, **Solver:** Co-evolutionary QIGA, **Size:** Up to 500 jobs and 20 machines/operations (Taillard).

They compare the performance of their algorithm with classical algorithms. Each algorithm is stopped after a pre-determined CPU time. Their algorithm achieves the best performance in terms of average relative deviation from the best known solution.

Gu et al. [76]: **Type:** FSSP, **Constraints:** D, pickup and delivery constraints, **Objective:** Makespan, **Solver:** Mutualism QIGA, **Size:** Up to 100 jobs and 5 machines/operations (Taillard).

The problem is first formulated as MIP. In their setting, they consider the pickup of the unprocessed jobs to the machines and the delivery of the finished jobs to the costumers, implying additional constraints such as that the delivery time cannot be earlier than the finishing time or that space constraints within the delivery trucks must be satisfied. They include the pickup of the materials and the delivery of the finished goods in their problem. They apply a mutualism QIGA, which allows to adjust the size of the population dynamically in contrast to QIGA. The proposed algorithm is compared with QIGA from Wang et al. [221] and a classical mutualism GA. It turns out that the proposed algorithm achieves lower makespans than the competitors, in particular in the larger problem instances, while requiring more computation time.

Singh et al. [188]: **Type:** FJSSP, **Constraints:** M, P, unavailability, **Objective:** Makespan, relative makespan difference, expected total tardiness, expected flowtime, **Solver:** QIPSO, **Size:** Up to 20 jobs, 10 machines, 25 operations (LA18).

They consider random machine breakdowns, so that unavailability slots occur, resulting from breakdown time and repair time. Experiments reveal that the realized makespan is improved in nearly all test instances, compared with that achieved by a classical PSO.

Singh and Mahapatra [187]: **Type:** FJSSP, **Constraints:** D, P, **Objective:** Makespan, **Solver:** QIPSO, **Size:** Up to 20 jobs and 10 machines, 25 operations (LA18).

They compare the performance of their QIPSO with several classical evolutionary algorithms in terms of the relative deviation of the makespans between QIPSO and each of the competitors. Their algorithm achieves better makespans in most of the problem instances.

Zhao et al. [241]: **Type:** NW-FSSP, **Constraints:** D, setup times, release times, **Objective:** Makespan, **Solver:** QIEA, **Size:** Up to 100 jobs and 20 machines/operations.

They consider sequence-independent setup times and release dates, indicating that each machine requires a setup time that depends on the job that should be executed on the machine, and that a machine that has been set up for a particular job stays idle until this job has been released. They propose two local search methods. One is an interchange search where always two entries of the current job permutation are swapped in order to find an improved permutation. The other is an insert search where always one entry of the current permutation is inserted iteratively into each position of this permutation. They compare their algorithm with a classical genetic algorithm, a QIEA, and a heuristic method from the literature. All algorithms get the same runtime limit. Their algorithm shows superior behavior in terms of relative improvement over the average makespan.

Wu et al. [218]: **Type:** FJSSP, **Constraints:** A, M, P, **Objective:** Makespan, **Solver:** QIEA, **Size:** Up to 20 jobs, 10 machines, 25 operations (LA18).

As Wu and Li [217], they propose an elitist strategy. Moreover, they enhance the algorithm with a local search. Their algorithm turns out to be competitive in terms of makespan, computation time, and success rate that have been obtained with classical EAs from the literature.

Kaipu et al. [114], Sun and Xu [199]:[11] **Type:** FJSSP, **Constraints:** A, P, **Objective:** Makespan, **Solver:** QIGA, **Size:** 8 jobs, 8 machines, 5 operations.

They propose to iteratively adjust the rotation angle, which is required to update the individuals (cf. Sect. 3.4.2), using a stochastic cloud model, which is a Gaussian distribution. The goal is to create realizations from this Gaussian distribution and to process these numbers in order to sample a rotation angle. The goal is to decrease the amplitude of the rotation angle for individuals with a high fitness, while increasing it for those with small fitness. Their algorithm achieves slightly better makespans than a standard QIGA in the spirit of Narayanan and Shmatikov [147].

Zhang and Hu [237]: **Type:** FJSSP, **Constraints:** D, F, P, **Objective:** Makespan, **Solver:** QIPSO, **Size:** Up to 20 jobs, 15 machines, 15 operations (MK10).

A comparison with two classical and a heuristic algorithm shows that their algorithm often retrieved the best known solutions.

Zhu et al. [240]: **Type:** NW-FSSP, **Constraints:** M, **Objective:** Makespan, **Solver:** Quantum-inspired cuckoo co-search, **Size:** Up to 100 jobs and 20 machines/operations (Taillard).

They propose an encoding of the population via Bloch coordinates, i.e., the qubits are described by the corresponding coordinates on the Bloch sphere. More precisely, each gene corresponds to one job, whose order is determined by the amplitude of the representing qubits. They prove convergence of the proposed algorithm. In terms of the relative deviation to the currently known best solution, their algorithm outperforms classical evolutionary algorithms and a simpler variant of their algorithm, a quantum-inspired cuckoo search.

Chen et al. [47]: **Type:** FSSP, **Constraints:** D, batch constraints, capacity, **Objective:** Makespan, **Solver:** Quantum-inspired ant colony optimization, **Size:** Up to 300 jobs, 2 machines/operations.

They consider the situation that the jobs are gathered in different batches and that those batches are delivered to the machines, resulting in capacity constraints in the sense how many jobs can be stored, and in batch-related constraints, for example that the finishing time of a batch equals the finishing time of the last job of the batch. Constraint set D therefore refers to the processing time of both the individual jobs and the batches. In particular, their proposed always considers two ant populations, where one population is evaluated on data where the jobs are sorted in an ascending order according to their size, the other on descending ordered data. They compare their algorithm with a hybrid discrete differential evolution and a batch-based hybrid ant colony optimization algorithm. While their algorithm beats the former competitor in terms of the achieved makespan and often in the running time, it clearly beats the second competitor in terms of running time, while achieving slightly better makespans.

Xin et al. [226]: **Type:** FJSSP, **Constraints:** D, P, **Objective:** Expected makespan, expected maintenance cost, **Solver:** QIGA, **Size:** Up to 22 jobs with 3 operations, 7 machines.

They consider machine-specific failure rates and the option to do preventive maintenance if there is currently no machine failure and maintenance after a failure. In their

[11] The papers are almost identical, just the authors are different.

QIGA, they consider binary variables which indicate whether a certain operation of a certain job is scheduled in a particular position of a certain machine.

Ning et al. [148]: **Type:** FJSSP, **Constraints:** D, **Objective:** Makespan, carbon emissions, total costs, **Solver:** Quantum-inspired bacterial foraging optimization, **Size:** Up to 15 jobs, 10 machines, up to 56 operations in total (Kacem5).

They model the position and state of the individuals by a wave function, letting it evolve using Schrödinger's equation, so that individual motion equations can be realized using Monte Carlo simulation. They consider an application in a manufacturing process and also take the carbon emissions of the machines into account. As a machine also produces emissions when being idle, one can interpret the carbon emission objective as a soft counterpart of the hard shutdown constraints S from other scheduling problems. Their algorithm slightly outperforms a quantum particle swarm optimization algorithm and considerably outperforms classical bacterial foraging algorithms in their experiments in terms of the objectives.

Zhang et al. [236]: **Type:** FJSSP, **Constraints:** D, P, R, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 15 jobs and 10 machines, 56 operations in total (Kacem5).

They consider a dual-resource constrained FJSSP. Here, one has a set of equipment and a set of workers, with the condition that for each job, only one worker can execute it simultaneously and that the equipment can also only be used by one worker simultaneously. They apply a niche initialization of the population where individuals from different categories are spawned in order to maintain diversity. In addition to that, they utilize an adaptive rotation angle adjustment based on the relative difference of the fitness of the current and the fitness of the best individual. They compare their algorithm with a standard QIGA in the spirit of Narayanan and Shmatikov [147], a GA and a PSO. Their algorithm achieves slightly better makespans and converges quicker w.r.t. the number of generations.

Ripon and Singh [176]: **Type:** JSSP, **Constraints:** D, P, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 20 jobs, 15 machines, 15 operations (abz9).

In their experiments, their QIGA achieves a better performance in terms of makespan than a classical GA.

Xu et al. [225, 229]: **Type:** FJSSP, **Constraints:** A, D, E, M, P, **Objective:** Makespan, **Solver:** QIPSO, **Size:** Xu et al. [229]: Up to 20 jobs, 10 machines, 25 operations (LA18), Xu et al. [225]: Up to 15 jobs, 10 machines, up to 56 operations in total (Kacem5).

The problem is first formulated as MILP. They propose a chaotic encoding scheme in order to simplify the representation of a solution. The real-valued position of a particle is initialized using Monte Carlo random simulation. Each particle represents an operation. As for the correspondence of the particle positions and the actual scheduling tasks, they propose an encoding scheme in the sense that the integer part of the position represents the priority of the corresponding operation. The machine selection is based on this priority, i.e., operations with a higher priority a scheduled to the machines that can finish these operations more quickly. After machine selection, the fractional part of particle positions for individuals which are operated on the same machine defines in which order they are processed. Here, the operations are processed on this machine w.r.t. the descending order of the corresponding fractional parts, i.e., the operation with the highest fractional part is processed first on this machine. Xu et al. [225] argue that the standard encoding scheme has disadvantages such as sensitivity to the encoding parameters and large computational overhead due to requiring one dimension that represents the operations and one

that represents the machine assignments. Xu et al. [229] compare their algorithm with the QIPSO from Singh and Mahapatra [187], another QIPSO from the literature, a heuristic method and two classical EAs, on the Kacem (Kacem1 to Kacem5), Brandimarte (MK01 to MK10), Dauzère-Pérès (L01 to LA18) data and a data set from the industry. Their algorithm achieves better makespans and deviations from the best known makespan in most of the problem instances. Moreover, their algorithm converges quicker w.r.t. the number of iterations. They claim that is also requires less computing time than the competitors in most of the problems, but the running times are not reported. In Xu et al. [225], they compare the proposed encoding scheme with different parameters with the standard encoding scheme on the Kacem (Kacem1 to Kacem5), Brandimarte (MK01 to MK10) data and two data sets from the industry, confirming quicker convergence, better makespan values and a lower runtime when using a chaotic encoding scheme compared to using the standard encoding scheme.

Xu et al. [227]: **Type:** FJSSP, **Constraints:** D, E, M, P, **Objective:** Makespan, **Solver:** QIPSO, **Size:** Up to 20 jobs, 10 machines, 25 operations (LA18).

They use the chaotic encoding scheme proposed in Xu et al. [229] in order to represent the schedule. Their algorithm differs from that in Xu et al. [229] by including adaptive crossover and mutation probabilities, depending on the fitness value, and a Gaussian distribution used for the Monte Carlo random simulation. They compare their algorithm with a classical PSO, a classical GA, a quantum-inspired PSO and the algorithm from Xu et al. [229]. All algorithms run with the same number of individuals and generations. On the Kacem data (Kacem1 to Kacem5), the Dauzère-Pérès data (LA01 to LA18) and some industry data sets, the proposed QIPSO achieves the best results. On the Brandimarte data (MK01 to MK10), it often achieves the best results, except for MK04, where the GA is slightly better.

Chen et al. [38]: **Type:** FJSSP, **Constraints:** A, D, M, P, arrival time, **Objective:** Resource utilization, makespan, **Solver:** GA with QA, **Size:** Up to 1650 jobs, 56 machines and 446 masks.

They consider photolithography scheduling in thin-film transistor-liquid crystal display and semiconductor manufacturing. In the photolithography stage of this procedure, several tasks such as photo engraving and cleaning have to be processed on the respective machines with the correct photomask. Therefore, in this problem, one not only has to assign operations to machines but also masks to machines and to operations. The arrival time constraints indicate that the operation and the mask is ready at the given time step. In their algorithm, the individuals are represented by bits, not by qubits. Each individual consists of 3 chromosomes, which encode the jobs, machines and masks by integers, respectively. The job and the mask chromosome are combined into a job-mask chromosome, representing the available masks for each job, where one tries to match the earliest jobs with the masks with the highest priority. In the next step, masks are assigned to the machines. The quantum part of the algorithm is located only in the mutation step, where QA is applied in order to compute a transfer matrix according to which the current individuals are updated. This is done by defining a Hamiltonian which aims at maximizing the difference in the fitness values of the current best and worst individual, encouraging diversity of the population and avoiding to be stuck in local optima. They compare their algorithm with classical GA and a baseline solution from practice. In nearly all scenarios

and metrics (execution time, mask-move time, resource utilization), the results from their GA is significantly better than the competing solutions.

5.3 Task scheduling problems

Mo et al. [143]: **Type:** Task scheduling on computational grids, **Constraints:** R, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 150 tasks and 7 nodes.

Each gene corresponds to a computational resource, and each individual is composed of as many genes as there are tasks. They compare their QIGA with a GA and the min-min-algorithm. Their QIGA achieves a lower or equal number of discarded tasks and, in most cases, a lower makespan than its competitors.

Prakash and Vidyarthi [171]: **Type:** Task scheduling on computational grids, **Constraints:** D, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 10^6 tasks and 1000 nodes, up to 4000 million instructions per task.

The genes correspond to the nodes, their values represents the task. They compare their QIGA with a GA. In particular for large-scale experiments, the makespan achieved by QIGA is nearly always lower than that of GA, and QIGA leads to a quicker makespan convergence in terms of generations. They explain this observation by the large search space, which is easier to explore by QIGA than by GA due to higher diversity of the population.

Boutekkouk and Oubadi [25]: **Type:** Task scheduling on distributed systems, **Constraints:** F, **Objective:** Response time, number of tasks missing their deadlines, **Solver:** QIGA, **Size:** 20 tasks, 4 nodes.

They distinguish between periodic and aperiodic tasks, and assume that the periodic ones are synchronous in the sense that their arrival time is identical. For those tasks, they further consider soft constraints in the form of average case execution times, and hard constraints in the form of worst-case execution times which need to be satisfied. They assume that aperiodic tasks always correspond to soft constraints and that the arrival of such tasks is Poisson distributed. They consider the chromosomes as qubit matrices, where each cell corresponds to one pair consisting of a processor and a task.

Gandhi et al. [72]: **Type:** Task scheduling on distributed systems, **Constraints:** D, P, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 21 tasks and 11 nodes.

In a task scheduling problem on distributed systems, the goal is to schedule computing tasks onto different computers. In this particular situation, there can be communication costs between tasks that have been sent to different computers. Moreover, some tasks may need the result from other tasks, implying precedence constraints. A comparison with two classical evolutionary algorithms reveals that the proposed QIGA achieves the best makespans.

Konar et al. [105]: **Type:** Task scheduling on distributed systems, **Constraints:** D, F, **Objective:** Deadline satisfaction, earliest deadline first, shortest computation time first, **Solver:** QGIA, **Size:** Up to 600 tasks, up to 20 nodes.

Each gene corresponds to a processor node, and its value represents the task. Their algorithm is compared with standard GA and hybrid PSO in terms of the fraction of scheduled jobs and the running time. The QIGA shows superior behavior in nearly all problem instances, which they explain by a better diversity of the QIGA population.

Singh and Raza [192]: **Type:** JSSP, **Constraints:** D, P, **Objective:** Job turnaround time, **Solver:** QIGA, Quantum-inspired binary gravitational search (QIBGS), **Size:** Up to 1000 tasks and 100 nodes, and 4000 million instructions per task.

They consider job application in the computational mobile grid. The idea is to harness available computational power of mobile devices, so that computational tasks should be scheduled across those mobile devices. Here, the execution time and precedence constraints among sub-tasks of such jobs have to be respected, leading to the constraints D and P. A comparison with a proposed QIGA reveals that the achieved turnaround time decreases often faster w.r.t. the number of iterations for the QIBGS algorithm than for QIGA.

Alam and Raza [12]: **Type:** Task scheduling on distributed systems, **Constraints:** D, P, **Objective:** Load imbalance, load balancing cost, **Solver:** QIGA, **Size:** Up to 10 tasks, up to 100 million instructions per task, 10 nodes.

In their QIGA, each individual in the population is a qubit string where each gene represents a processor node. As each gene corresponds to a module of a task, all such modules are scheduled to the processor nodes during decoding. The performance of their QIGA is compared with that of three classical evolutionary algorithms in terms of the hypervolume difference w.r.t. the achieved Pareto fronts and the Pareto fronts themselves. QIGA shows a better performance.

Mishra et al. [140]: **Type:** Task scheduling in cloud computing, **Constraints:** D, P, **Objective:** Resource utilization, makespan, load balancing, energy consumption, **Solver:** Quantum-inspired binary chaotic salp swarm optimization, **Size:** Up to 3000 tasks and 300 nodes, up to 15,000 million instructions.

Each salp is given by a qubit matrix whose dimension equals the number of tasks and machines, so each qubit can collapse into one of the binary states indicating that the respective task is either scheduled on the corresponding machine or not. They compare their algorithm with classical EAs. The proposed algorithm shows superior behavior on nearly all problem instances in the four metrics of which the objective is composed and the response time, which they explain with the higher diversity of the population in their algorithm. All algorithms have been executed until a limit of 0.5 seconds for a fair comparison.

Thakur et al. [201]: **Type:** Task scheduling on distributed systems, **Constraints:** D, P, **Objective:** Makespan, load balancing, resource utilization, **Solver:** QIBGSA, **Size:** Up to 800 tasks, 200 million instructions per task, 6 processing nodes.

In their algorithm, each gene corresponds to a task, and the qubits of this gene encode the processing node. They compare their algorithm with the QIGA from Alam and Raza [12] and classical EAs. In terms of all three objectives, their algorithm often performs slightly better.

Belmahdi et al. [24]: **Type:** Task scheduling in cloud computing, **Constraints:** P, R, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 250 tasks, 25 nodes, 1000 million instructions per task.

The genes correspond to the tasks, and their value represents the node. They compare their algorithm with simple first come first serve and with a GA. Their algorithm always achieves the best makespan, which they explain with a higher diversity of the QIGA population.

Hussain et al. [93]: **Type:** Task scheduling in hybrid clouds, **Constraints:** F, P, **Objective:** Makespan, energy consumption, **Solver:** QIGA, **Size:** Up to 481 tasks and 2 clouds.

In their encoding, the genes are given by quantum matrices where the columns represent the tasks and the rows the machines, so each cell corresponds to exactly one partic-

ular machine-task-pair. A comparison with classical EAs reveals superior Pareto fronts of makespan and energy consumption over those achieved by its competitors, which they explain with a higher diversity of the QIGA population.

Misra and Kuila [136]: **Type:** Task scheduling in cloud computing, **Constraints:** D, **Objective:** Makespan, energy consumption, **Solver:** QIGA, **Size:** Up to 5000 tasks and 1250 nodes.

Each gene corresponds to a task, its value represents the task. In a comparison with several EAs and the min-min-algorithm, their algorithm outperforms the competitors in terms of energy consumption and produces comparable results in terms of makespan for medium-sized problems, while achieving better makespans on large-scale problems.

Su et al. [195]: **Type:** Task scheduling on distributed systems, **Constraints:** D, F, **Objective:** Earliest deadline first, shortest computation time first, **Solver:** Quantm-inspired simplified swarm optimization, **Size:** Up to 600 tasks, 20 nodes.

Each gene represents a processor node and the value the task. They compare their algorithm with classical GA, a hybrid PSO and the QIGA from Konar et al. [105] in terms of computation time and the fraction of scheduled jobs. Their algorithm slightly outperforms the QIGA on all problem instances, and as the QIGA already outperformed GA and PSO in Konar et al. [105], also the classical competitors.

Jain and Sharma [99]: **Type:** Task scheduling in cloud computing, **Constraints:** D, F, budget, **Objective:** Makespan, profit, service level agreement violation rate, **Solver:** Quantum-inspired salp swarm grey wolf optimization, **Size:** Up to 5000 tasks and 300 nodes.

As for the constraints F that correspond to deadline violations, they can be stated as either hard or soft constraints, where the latter allows for certain delays. In their algorithm, the population is composed by salps, where each salp is an allocation of each task to a machine, accompanied by checking whether the allocation would violate a hard deadline constraint, which results in a rejection of a considered task. A quarter of the salps are initialized randomly, while the remaining salps are initialized from a given wave function and updated using the Monte Carlo method. In order to prevent the population from being stuck in a local optimum due to the leading salp that attracts the population in salp swarm optimization, they propose an alternative updating scheme inspired from grey wolf optimization, where the alpha and beta salp (i.e., the one with the best and second-best fitness value) are selected. Then, one half of the salps update their position only according to the alpha, the other half according to both the alpha and the beta salp. They compare their algorithm with classical EAs in terms of several quality metrics such as makespan, profit, or resource utilization. The quantum salp swarm optimization without the grey wolf optimization nearly always outperforms the classical competitors, while the combined algorithm always outperforms its competitors, often even considerably.

Li et al. [124]: **Type:** Task scheduling on distributed systems, **Constraints:** R, **Objective:** Earliest completion time, **Solver:** QIGA, **Size:** Up to 80 tasks, 4 nodes.

Each gene corresponds to a task, and their values represent the processor. They compare the performance of their algorithm with a GA and to ant colony optimization algorithms in terms of makespan and speedup (ratio of computing costs on the distributed system and those on a single processor). It turns out that their algorithm outperforms the competitors in both metrics, which they explain with a higher diversity of the QIGA population. As

for the computational time, they point out that their algorithm is cumbersome due to handling many binary matrices in the encoding and decoding steps.

Galavani et al. [85]: **Type:** Task scheduling in mobile edge computing, **Constraints:** F, R, **Objective:** Weighted sum of execution time, energy consumption and resource utilization, **Solver:** QIGA, **Size:** Up to 50 servers and 200 users.

Each individual consists of mn qubits, where qubit q_{ij} indicates whether task i is assigned to node j. They compare the QIGA with a heuristic algorithm, a quantum-inspired differential evolution algorithm and a quantum PSO algorithm. The QIGA solutions achieve the lowest energy consumption and makespan, while achieving highest resource utilization. In contrast to all other methods, only the QIGA solutions always respect the deadlines.

Acharya et al. [11]: **Type:** Task scheduling in multi-processor computing, **Constraints:** D, **Objective:** Makespan, resource utilization, **Solver:** Quantum-inspired binary chaotic salp swarm optimization (Mishra et al. [140]), **Size:** Up to 5 tasks and 25 nodes.

They compare different classical evolutionary algorithms and the quantum-inspired binary chaotic salp swarm optimization in terms of makespan. The performance of the quantum-inspired algorithm is comparable to that of the competitors.

Hussain et al. [94]: **Type:** Workflow scheduling in hybrid clouds, **Constraints:** D, F, P, R, **Objective:** Makespan, total execution cost, **Solver:** QIGA, **Size:** Up to 30 tasks and 100 nodes.

Each qubit indicates whether the corresponding task is assigned to the corresponding node. They compare their QIGA with a PSO, a whale optimization algorithm and two GAs, all purely classical. The achieved makespans of QIGA are often the best or at least among the best results, compared to those of the other four algorithms. When minimizing the cost subject to a makespan constraint, the QIGA solutions always lead to the lowest costs. The authors argue that QIGA can, due to qubit representation and rotation and mutation operators, achieve better population diversity and thus can better explore the search space.

Acharya et al. [10]: **Type:** Task scheduling in cloud computing, **Constraints:** D, **Objective:** Makespan, resource utilization, **Solver:** Quantum-inspired binary chaotic salp swarm optimization (Mishra et al. [140]), **Size:** Up to 1000 tasks and 300 nodes.

The quantum algorithm is compared with different classical evolutionary algorithms. In terms of the objective, load imbalance and response time, the quantum algorithm provides better solutions than three classical competitors, but is constantly outperformed by a whale optimization algorithm proposed in Acharya et al. [10], which they explain by the interplay of local and global search, enabling a better exploration of the search space.

Jain et al. [96]: **Type:** Task scheduling in cloud computing, **Constraints:** D, F, budget, **Objective:** Makespan, profit, service level agreement violation rate, task rejection rate, **Solver:** Quantum-inspired firefly algorithm, hybrid, **Size:** Up to 1052 tasks and 250 nodes.

They propose a quantum-inspired firefly algorithm and a hybrid algorithm where a heuristic first rejects infeasible tasks, and the quantum-inspired firefly algorithm is applied afterwards. The hybrid algorithm showed the best performance among a classical PSO, a classical SSA, the algorithm from Jain and Sharma [99] and the pure quantum-inspired firefly algorithm, in terms of makespan.

Lilhore et al. [117]: **Type:** Task scheduling in cloud computing, **Constraints:** D, **Objective:** Energy consumption, makespan, resource utilization, **Solver:** Hybrid, **Size:** Up to 1.2 million jobs and 12,500 nodes.

They propose a method where a QIEA is used in order to schedule tasks on virtual machines, minimizing energy consumption, resource utilization and makespan. The solution is further enhanced with a hybrid deep reinforcement learning (RL) algorithm that predicts the actual workload of the machines, and triggers a task migration from overloaded to underloaded machines. Finally, a multi-objective optimization algorithm further encourages finding a Pareto-optimal solution. They argue that the QIEA part allows for efficient search space exploration, which is confirmed by experiments where they apply their method with and without the QIEA part. The complete method outperforms classical competitors in terms of the individual objectives.

Mahjoub et al. [137]: **Type:** Task scheduling in cloud computing, **Constraints:** D, F, M, R, **Objective:** Makespan, resource utilization, load imbalance, throughput, energy consumption, efficiency, **Solver:** Hybrid, **Size:** Up to 1000 tasks with up to 900,000 million instructions, 50 nodes and 5 data centers.

Their algorithm is based on the arithmetic optimization algorithm (AOA, Abualigah et al. [6]), which starts with a randomly initialized population. In the exploration phase, the current individuals are, based on the current best solution, updated with a randomly chosen procedure. With a certain probability that increases with the number of iterations, the exploration phase ends and an exploitation phase starts. In order to increase the diversity of the population, Mahjoub et al. [137] combine this scheme with a Gaussian quantum mutation operator. In addition, they use Q-learning (Watkins and Dayan [214]), a reinforcement learning (RL) technique, in order to learn a policy that decides in each iteration whether an exploration or exploitation phase should be done. Their algorithm is compared with AOA, Q-learning, a grey wolf optimizer, a whale optimizer and the sine-cosine algorithm. For all individual objective components, their algorithm achieves better results than the competitor algorithms, sometimes even statistically significant at 5% level.

Naik et al. [145]: **Type:** Task scheduling in edge computing, **Constraints:** D, P, R, **Objective:** Makespan, resource utilization, energy consumption, **Solver:** QIPSO, **Size:** Up to 2000 tasks and 5 nodes.

In their quantum-inspired PSO, each individual consists of $n\lceil \log_2(m) \rceil$ qubits, where the second factor results from the binary representation of the node number. Each individual therefore indicates for each task-node pair whether the task is assigned to the respective node. The performance of the proposed QIPSO is compared to classical algorithms, a quantum-inspired differential evolution algorithm, and the quantum-inspired cuckoo co-search algorithm from Zhu et al. [240]. The proposed method often achieves the best results.

5.4 Railway/maritime scheduling problems

Ji et al. [102]: **Type:** Lock scheduling, **Constraints:** A, restrictions concerning starting times of lockage operations and their minimal duration, **Objective:** Waiting time, water consumption, **Solver:** Hybrid, **Size:** 120 ships, five chambers.

They propose a hybrid algorithm for a co-scheduling problem of cascaded locks with multiple chambers. This problem is decomposed into an inner and outer problem, which consider the ship placement and lockage numbers, respectively, and the inner scheduling problem. This scheduling problem is solved using a hybrid algorithm that combines binary gravitational search with modified moth-flame optimization. The constraint set A has to be understood in the sense that the direction in which the lock transfers next must match

with the desired direction of the ship. The direction is represented by an agent of the quantum-inspired gravitational binary search algorithm, while the starting times of the locking operations are represented by a moth of the modified MFO algorithm. In their experiments, two of the five chambers can only transfer ships into one direction, which enforces additional directional constraints apart from the usual constraint that ships can be transferred in only one direction simultaneously in each lock.

Ji et al. [103]: **Type:** Lock scheduling, **Constraints:** A, D, P, S, mooring constraints, lockage starting times, minimal lockage duration, **Objective:** Delay time of ships, time spent at the quay, **Solver:** Hybrid, **Size:** Up to 100 ships and 10 berths.

They consider lock and quay co-scheduling. They first formulate the problem as MILP. The constraints A, P, D, S have to be understood in the sense that a ship can be assigned to only one lock, that some ships must be prioritized over other ships, that a certain distance must be satisfied for two ships and that there must not be empty lockages, respectively. However, the lock scheduling and the berth problem are solved classically. Only the main problem, which is to select the modes for the ships, which are the lock mode (enter the lock) or the transshipment mode (transfer the loading to another ship at the quay), is solved via the quantum algorithm, which is a fuzzy-controlled QIBGSA. The idea is to adjust the neighborhood of the individuals with which they interact by gravity by a fuzzified diversity measure, i.e., instead of using the true values of a diversity measure of the population, a jittered value is used.

Pereira et al. [158, 159, 161]: **Type:** Oil crude scheduling, **Constraints:** Capacity, task uniqueness, idle times, crude flow, **Objectives:** Downtime of a distillation unit, number of switchovers, the downtime of the pipelines, the volume of oil crude that was not offloaded during a certain time window, **Solver:** Quantum-inspired grammar-based linear genetic programming algorithm, **Size:** Up to one docking point, four terminal tanks, four refinery tanks, four ships, one subsea pipeline, one standard pipeline, two distillation units, and 480 hours of scheduling horizon.

The capacity constraints refer to the maximum volume of the terminal tanks, the task uniqueness constraints indicate that either a tank receives ship offload or it delivers oil into the pipeline. The idle times are necessary in order to remove water impurities or sediments from refinery tanks, and the crude flow constraints define the necessary transfer time given the maximum transfer rate. As for the encoding of a schedule, they first define several functions, such as transferring a proportion of the load of a particular ship compartment to a particular tank, or transferring crude from a terminal tank to a refinery tank. Each quantum individual therefore consists of a predefined number of genes, which encode the instructions that form the schedule. More precisely, a tree-type structure is given, where the top level corresponds to the functions. Then, for each individual function, the arguments have to be defined, e.g., the tank number or the ship compartment number. In the quantum genes, therefore, there is not only a probability distribution on the functions on the top level but also a probability distribution for each argument in the deeper level. Measurement then manifests concrete functions and arguments, which finally define a schedule, onto which the fitness function can be evaluated. Therefore, in their algorithm, the term "grammar-based" indicates that the functions and arguments are properly represented so that the underlying tasks can be executed. Pereira et al. [159] consider different scenarios in order to test their algorithm, but do not compare it with other algorithms. In Pereira et al. [161], a non-dominated sorting strategy is proposed,

where individuals are only compared according to their fitness value if they both satisfy all constraints. Otherwise, the individual which violates less constraints is ranked higher. In Pereira et al. [158], the algorithm from Pereira et al. [159] is modified in order to enable a decomposition strategy of a multi-objective problem into single-objective sub-problems. For each sub-problem, there is a sub-population of individuals that are primarily designed in order to solve the respective sub-problem, from which one is considered as a centroid. After measurement, the fitness functions of the centroid individual and its neighboring individuals are compared, and the best one serves as the centroid in the next iteration. They compare this algorithm with those from Pereira et al. [161] and Pereira et al. [159], and conclude that it beats them in terms of ratio of acceptable solutions, and diversity, measured by the hypervolume and the number of different solutions found on the Pareto front.

Jing et al. [97]: **Type:** Railway scheduling, **Constraints:** M, demand, **Objective:** System revenue, **Solver:** Quantum-inspired immune clonal algorithm, **Size:** Up to 9 stations and 24 time steps.

Initially, they have a dynamic scheduling problem, indicating that supply and demand fluctuate over time. They propose to integrate a timeline, which partitions the dynamic scheduling problem into a series of static scheduling problems, where the individual static problems consider the traffic flows and demands in the respective time interval. The constraint M implies that the traffic capacity of the zones and the lines cannot be exceeded, while the demand constraints indicate that the demand of empty wagons and specific wagon types is satisfied. The system revenue is given by the aggregated benefits that cars of a certain type arrive at a certain station, reduced by the costs of the cars. In their algorithm, each antibody in the population consists of as much genes as the product of time steps, starting points and destinations of empty wagons, respectively, and wagon types, so that each gene represents a binary variable indicating whether an empty wagon of a certain type is scheduled from a certain place to a certain destination at a certain time.

Zhao et al. [239]: **Type:** Ship lift and lock problem, **Constraints:** A, D, P, S, lockage starting and finishing times, ship placement constraints, maximum costs, **Objective:** Maximize utilization, minimize tardiness or elapsed time, **Solver:** Hybrid, **Size:** Up to 95 ships.

They consider a ship lift and ship lock problem at the Three Gorges Dam. They decompose the problem into an assignment problem of the ships to either the ship lift or the locks and a scheduling problem. While the second problem is solved classically, they propose a hybrid algorithm for the first problem, where a quantum-inspired binary PSO and QIBGSA are applied. It turns out that QIBGSA provides the best results for this problem.

5.5 Other scheduling problems

Li and Song [126]: **Type:** Project scheduling problem, **Constraints:** D, P, R, **Objective:** Makespan, **Solver:** QIGA, **Size:** Up to 122 activities.

They compare their algorithm with another (not clearly specified) QGA from the literature, and conclude that their algorithm performs better in terms of average deviation from the best known solution.

Lei et al. [123]: **Type:** Robot scheduling problem, **Constraints:** D, processing time interval, recurrence, **Objective:** Throughput rate, **Solver:** QIEA, **Size:** Up to 22 machines.

In a robot scheduling problem, the goal is to find a schedule for a robot that transports goods across different machines in a factory. The constraint D has to be understood in the

sense that there must be enough time for the robot to reach the next machine within the pre-scribed time. The recurrence constraint indicates that the robot must eventually return to the starting point at the end of a cycle. They compare their algorithm with CPLEX and Yan's algorithm. With increasing problem size, the improvement in comparison with Yan's algorithm increases, while the computation time is often higher. As for the comparison with CPLEX, the proposed algorithm leads to inferior solutions on larger instances, but the computation time is considerably lower.

Liu et al. [131]: **Type:** Integrated process planning and scheduling problem, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** QIEA, **Size:** Up to 18 jobs, 15 machines, and 300 operations.

The individuals consist of two bit strings and one qubit string. One bit string represents the total number of jobs and machines in the problem instance, the other one represents the machines on which each of the operations is performed. The qubit string is converted into decimal job numbers for decoding and then represents the job permutation. In the evolution, one-way crossover is applied to the bit strings and mutation to the qubit strings. In terms of best achieved makespan, their algorithm outperforms its competitors, which are four classical EAs, except for the two largest problem instances.

Saad et al. [183, 184]: **Type:** Project scheduling problem, **Constraints:** D, P, R, **Objective:** Makespan, **Solver:** Quantum-inspired differential evolution algorithm (Saad et al. [183]), QIGA (Saad et al. [184]), **Size:** Up to 122 activities.

The goal is to solve a resource-constrained project scheduling problem, where one has a set of activities, accompanied with required resources and duration times. The two proposed algorithms show a similar performance in terms of the average relative deviation from the best known solution from the literature, and the performance of both lies in the performance ranges of some classical competitors.

Tayarani-Najaran [206]: **Type:** (Nurse) timetabling problem, **Constraints:** Assignment, capacity, interruption, soft constraints, **Objective:** Feasibility, **Solver:** QIEA, **Size:** Up to 132 teachers, up to 102 classes; up to 120 nurses and 8 weeks planning horizon.

The assignment constraints are standard constraints such that a teacher can hold only one lecture at the same time or that a room can only be occupied by one class at the same time. Capacity constraints indicate the number of lectures a teacher has to give and a class has to attend, while interruption constraints prescribe that free time slots for a class can only be in the last time slot of a day. Soft constraints correspond to teacher-individual wishes. In their QIEA, each binary variable indicates whether a particular teacher holds a lecture on a certain subject, attained by a certain class, on a given period of a given day. Therefore, each individual must consist of sufficiently many qubits in order to represent all combinations. Due to the difficulty to satisfy all constraints, they consider a repair method that essentially randomly picks some subject, teacher, period and day, and checks whether the teacher is available. If yes, one randomly picks a class and checks whether the class is available. If not, the binary variable that indicates that this class should attend the chosen lecture with the chosen combination of teacher, period and day is set to zero. Moreover, they propose two novel operators, namely the reinitialization operator which reinitializes the population after convergence in order to achieve diversity, and the diversity-preserving operator, which re-initializes individuals which are searching around the same local optimum. In total, they have proposed three QIEAs, where the one with the reinitialization operator and the one with the diversity-preserving operator count as

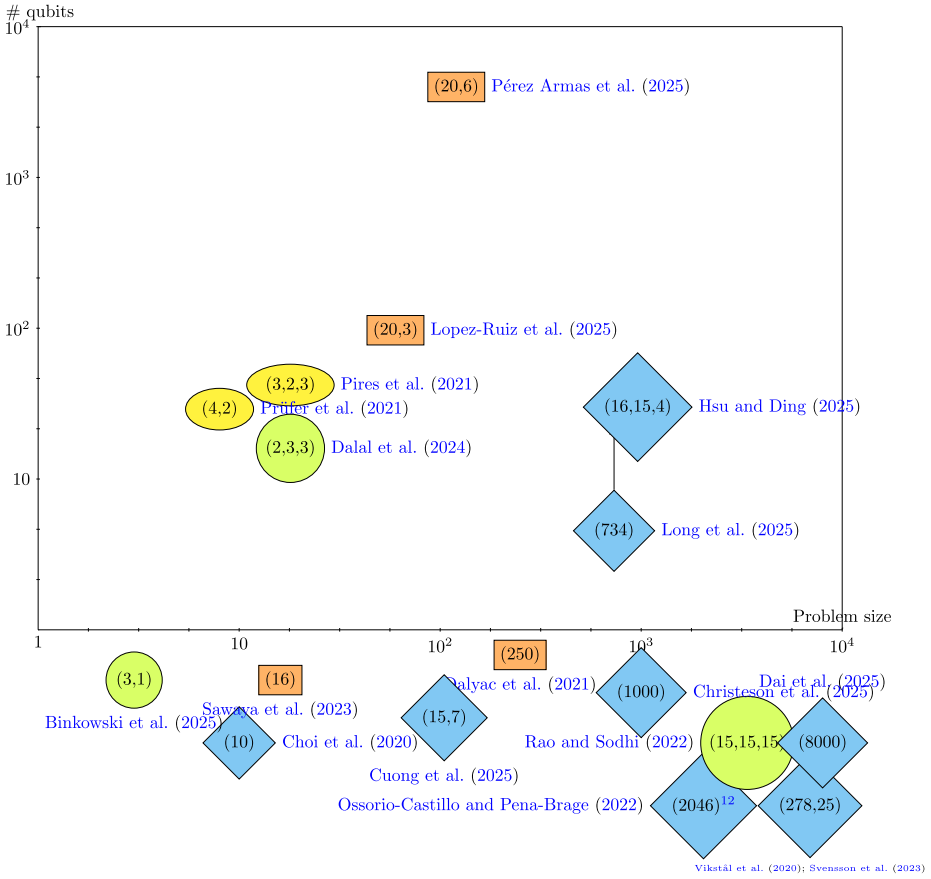

Figure 9 Overview of scheduling problems that are solved neither directly via a QUBO/CQM solver nor a QIGA. The horizontal axis refers to the size of the problem. The vectors in the nodes show the largest problem instances addressed in the respective paper. The vertical axis denotes the number of qubits used. The location of the node, with respect to its center, is determined by the largest combination of problem size and number of qubits presented in the corresponding reference. A rectangle refers to a single-stage job scheduling problem, a circle to a multi-stage job scheduling problem, an ellipse to a shift scheduling/timetabling problem, and a diamond to another type of scheduling problem. References where the number of qubits was not clearly reported are depicted below the horizontal axis

individual algorithms in the experiments. On school timetabling problems, one of their algorithms achieves the best solutions, in terms of a cost function where the constraints are replaced by penalty terms, on nearly each problem instance. They also evaluate the Friedman rank, which shows that their three algorithms are superior than the competing classical EAs. On the nurse scheduling problems, one of their algorithms attains the best solution in many instances. The Friedman rank indicates their algorithm with the diversity-preserving operator may be best or at least second best, only inferior to a branch-and-bound algorithm.

6 Other quantum approaches for scheduling problems

This section compiles all works that neither directly use a quadratic problem formalization nor an evolutionary algorithm. See Fig. 9 for an overview.

[12] 1776 continuous and 270 binary variables.

Choi et al. [44]: **Type:** Wireless scheduling problem, **Constraints:** Mutual exclusivity constraints, **Objective:** Sum of realized weights, **Solver:** QAOA, **Size:** 10 links.

The wireless scheduling problem considers the flow of information along wireless channels. They formulate it as a maximum weight independent set problem, where the weight is the backlog in a channel. First, they construct a conflict graph where a vertex represents the situation that the corresponding links suffer from interference, which implies that both channels cannot be selected simultaneously. They compare the performance of QAOA with different iteration numbers with greedy search and random search. In terms of the normalized realized weights, QAOA outperforms the classical competitors, where the performance increases with growing alternations number.

Svensson et al. [180], Vikstål et al. [211]: **Type:** Tail assignment problem, **Constraints:** M, **Objective:** Feasibility, **Solver:** QAOA, **Size:** Up to 25 routes and 278 flights (Vikstål et al. [211]).

Vikstål et al. [211] consider the tail assignment problem, which is a scheduling problem where aircrafts have to be assigned to a set of flights. Normally, the objective is to minimize the overall costs. Here, they consider only a simplified version where costs are ignored and the goal is to only find a feasible solution, which can be identified with an exact cover problem. Svensson et al. [180] extend their result by first applying a branch-and-price strategy, whose corresponding relaxed master problem can be either an exact cover problem or a set partitioning problem.

Pires et al. [160]: **Type:** School timetabling problem, **Constraints:** Assignment constraints, equipment constraints, **Objective:** No lectures at the last period of the day, more than one lecture a day, no more than two consecutive lectures, **Solver:** QAOA, **Size:** 3 teachers, 2 classes, 3 rooms (42 qubits).

The assignment constraints indicate that a teacher cannot hold more than one lecture simultaneously, that a class cannot have more than one lecture simultaneously and that there cannot be more than one lecture simultaneously in a room. The equipment constraints indicate that some lectures require special equipment, which may not be available in each room. They propose a two-stage approach, where QAOA is first applied to the graph coloring problem representing the hard constraints. If a feasible solution has been found, QAOA is applied again in order to maximize the satisfaction of the soft constraints.

Prüfer et al. [170]: **Type:** Shift scheduling problem, **Constraints:** M, S, workload constraints, partial schedule constraint, fairness constraints, unavailability constraints, **Objective:** Feasibility, **Solver:** Grover's Search, **Size:** Up to 4 operators and 2 positions (up to 29 qubits).

The fairness constraints indicate that all workers should work a comparable amount of days, while the partial schedule constraints refer to an already available partial on-call schedule. They apply Grover's Search by letting the Grover oracle count the number of violations in a schedule, trying to find schedules with minimal number of violations.

Dalyac et al. [51]: **Type:** UMSP / Interval scheduling problem, **Constraints:** D, S, **Objective:** Weighted average completion time, balancedness, **Solver:** QAOA, **Size:** Up to 250 jobs.

They consider smart-charging problems, i.e., scheduling electric vehicles to load stations, which can be identified with an UMSP. They reformulate the problem as weighted Max-k-Cut problem. The other problem considers load tasks that are represented as intervals, and each load task corresponds to a given group. This problem is an interval schedul-

ing problem, which is reformulated as maximum independent set problem. The balancedness objective of the latter problem indicates that no group should be over-represented among the selected tasks. For the UMSP, they derive that QAOA performs better than random search in terms of the approximation ratio.

Ossorio-Castillo and Pena-Brage [151]: **Type:** Refinery scheduling problem, **Constraints:** D, M, R, transfer rate constraints, demand constraints, transfer constraints, capability constraints, **Objective:** Total costs, **Solver:** Hybrid, **Size:** Up to 1776 continuous and 270 binary variables.

In their scheduling problem, there is a fleet of vessels and a set of storage and charge tanks and distillation units. The volume of the storage tank can differ within a certain interval, similarly, the crude transferred from a ship to a storage tank, from the storage tank to the charge tank and from the charge tank to the distillation unit. The scheduling problem is tailored to T discrete time units ahead. The constraints D have to be understood in the sense that the unloading time must be long enough to guarantee that the initial material can be completely transferred if the transfer rate is maximal. The constraints M indicate that material from one charging tank can be transferred to at most one distillation unit and that for each distillation unit, material is transferred from exactly one charging tank. The constraints R indicate that unloading can be completed only after the ship has arrived, respectively. They formulate the problem as a MILP and apply the Dantzig-Wolfe decomposition (Dantzig and Wolfe [60]). This allows for iteratively solving a linear continuous master problem, a linear continuous sub-problem and a binary sub-problem. The latter is solved using QA, while the continuous problems are solved classically.

Rao and Sodhi [175]: **Type:** FJSSP, **Constraints:** D, P, S, **Objective:** Tardiness, **Solver:** Leap hybrid solver, **Size:** Up to 15 machines, 15 jobs, 15 operations.

They consider dispatch rules, which induce an ordering of the remaining operations required by the machine, e.g., by selecting the job with the largest processing time first. They argue that dispatch rules immediately react to changes in the environment and allow for rescheduling. Therefore, they aim at developing a quantum formulation of a FJSSP combined with dispatch rules. The idea is to consider a set of dispatch rules and to assign exactly one rule to each machine, i.e., the goal is to find an optimal assignment of rules to machines.

Dominguez et al. [59]: **Type:** JSSP / Nurse scheduling problem, **Constraints:** D, M, P / balancing constraints, constraints regarding workload and consecutive shifts, **Objective:** Makespan / number of overall shifts, **Solver:** None, **Size:** No experiments.

They apply different encoding schemes, including that from Sawaya et al. [193], to different optimization problems.

Sawaya et al. [193]: **Type:** SMSP, **Constraints:** M, **Objective:** Weighted lateness, **Solver:** None, **Size:** Up to 16 tasks.

They propose a discrete quantum intermediate representation (DQIR) for combinatorial optimization problems in order to facilitate the interpretation and allow for a more efficient implementation. They propose such an encoding for single machine scheduling among other problems. The goal of their work is not to apply quantum algorithms, but to study the necessary resources, in particular the circuit depths, for different encoding schemes. They mainly focus on the encoding of matrix exponentials of the form $\exp(-i\beta H)$, for a Hamiltonian H and $\beta \in \mathbb{R}$, as their implementation is vital for algorithms such as QAOA.

Dalal et al. [55]: **Type:** FSSP, **Constraints:** A, D, P, **Objective:** Makespan, **Solver:** DCQO, hDCQO, **Size:** 2 jobs, 3 machines/operations, up to 16 qubits.

They consider scheduling a robot in a laboratory. As for the quantum algorithms, they use the digitized counterdiabatic quantum optimization (DCQO) algorithm proposed by Hegade et al. [86] and the hybrid DCQO (hDCQO) algorithm proposed by Cadavid et al. [43]. In addition, digitized QA and standard QAOA are applied. The QUBO problem is decomposed into sub-QUBO problems that require at most 16 qubits to be represented. The DCQO an hDCQO achieve success probabilities (the probability to find an exact solution) of around 1%, surpassing that of the digitized QA and QAOA, respectively.

Pérez Armas et al. [156]: **Type:** IMSP, **Constraints:** D, M, **Objective:** Makespan, **Solver:** Hybrid, **Size:** Up to 20 jobs and 6 machines (up to 3969 physical qubits).

The problem is formulated as MILP and decomposed via a Dantzig-Wolfe decomposition. The subproblem is formulated as a QUBO problem. Due to its complexity, they further divide the subproblem into a selection and a sequencing problem, so that only the sequencing problem is solved via QA. They compare the hybrid method with two competitor methods, where the sequencing problem is solved via SA and a commercial classical solver, respectively. The proposed method achieves, in average, the best solutions concerning makespan, while the computation time is often lower than a pure classical decomposition method, but significantly higher than the method where the sequencing problem is solved via SA.

Binkowski et al. [22]: **Type:** OSSP, **Constraints:** D, M, **Objective:** Total execution costs, **Solver:** Constraint-preserving QAOA (IBM Q System One), **Size:** 1 machine, 3 jobs.

Based on the constraint set, they apply the constraint graph model and prove that for OSSPs, one can find feasibility-preserving mappings. They show how, given a feasible initial state, one can express such mappings with a suitable mixer Hamiltonian. In a noiseless simulation, their approach achieves a better approximation ratio than soft-constrained QAOA implementations. On real hardware however, the noisy background dominates.

Christeson et al. [46]: **Type:** Resource scheduling problem, **Constraints:** Several limit constraints, **Objective:** Total operation cost, **Solver:** Hybrid, **Size:** Up to 1000 units (3000 binary variables).

They consider the unit commitment problem in power system management, aiming at optimally scheduling power generating units such that economic, environmental and physical constraints are satisfied. They formulate the problem as a QUBO problem and apply a Benders decomposition (Benders [15]), where the online/offline status, startup and shutdown of the units is determined by solving the Master problem, while the power generation is determined in the subproblem. The Master problem is solved using the LeapHybridCQMSolver on D-Wave Advantage. The solution time for the hybrid algorithm quickly becomes significantly lower than for SA and a genetic algorithm, while in comparison with an MINLP solver, the hybrid solver has comparable computation time for smaller systems, but lower computation time for systems with more than 400 units. In addition, they use the MINLP solution as baseline, w.r.t. to which the optimality gap, in terms of the relative absolute difference of the objective values, is computed. It turns out that the hybrid solution achieves a much lower gap than SA and the genetic algorithm, which is largest on problem instances with 40 units, where the gap is around 2%. The optimality gap decreases with increasing problem size.

Cuong et al. [42]: **Type:** Task scheduling in container terminals, **Constraints:** A, D, arrival time, **Objective:** Makespan, **Solver:** RL-assisted QAOA, **Size:** Up to 15 jobs and 7 machines.

They consider job scheduling in container terminals. First, they formulate the problem as a QUBO problem. In their RL-assisted QAOA, the QAOA-parameters are considered as the actions that the RL-policy can determine. For a given parameter, QAOA is applied and the current state is measured. The observation is evaluated by the reward function (here, the expected energy of the QUBO Hamiltonian), so that the RL agent can learn to optimize the parameter selection. A comparison with a classical GA shows that the proposed algorithm achieves, in average, lower makespans. They also consider a dynamic setting with stochastic processing times. In this setting, their algorithm achieves, in average, the lowest makespans compared to heuristic methods, GA and RL.

Dai et al. [58]: **Type:** Task scheduling in cloud computing, **Constraints:** D, **Objective:** Fraction of non-delayed tasks, actual response time, **Solver:** Quantum RL (TensorFlow Quantum), **Size:** 8000 jobs.

They formulate the problem as Markov decision process where for each job j, the corresponding state vector contains the differences between the idle time of each machine and the arrival time of job j, as well as the type of job j. A transition of such a state is the mapping onto another state after scheduling job j to some machine. The reward after scheduling job j is given by $\exp(R_j/W_j)$ if the job is completed before the expected response time and zero otherwise, where R_j is the actual response time and W_j the waiting time. In their Quantum RL algorithm, they approximate the Q-function via a quantum neural network instead of a classical neural network, using the temporal difference criterion. The experiments reveal that their Quantum RL approach significantly outperforms classical approaches in terms of average response time and success rate. The performance is similar to a deep Q-network and the QIGA from Hussain et al. [93] for low and medium arrival rates of tasks, but significantly better for high arrival rates. In the experimental settings however, they make transparent that the convergence time of their algorithm is significantly higher than that of the deep Q-network. In their experiments, they use an idealized QPU with near-perfect gate fidelity.

Grange et al. [83]: **Type:** FSSP, **Constraints:** D, F, release date, **Objective:** Makespan, **Solver:** Hybrid, **Size:** No experiments.

They propose a hybrid variant of a dynamic programming across the subset (DPAS) scheme. In the classical part, the optimal total completion time for a subset of $\lfloor n/4 \rfloor$ of the jobs, in dependence of the starting time t of the first job in this collection, is computed. This subset is enlarged in the sense that by applying Quantum Minimum Finding (Durr and Hoyer [50]), one finds first the optimal total completion time for a subset of $\lfloor n/2 \rfloor$ jobs, in dependence of t, and eventually of all jobs, starting at time 0. For the application on job scheduling problems, they propose a composed variant of this algorithm due to the fact that the optimal schedule for one subset of jobs is necessary in order to define the earliest possible starting times for the other jobs. The release date constraints indicate that a job cannot be processed before its release date.

Hsu and Ding [87]: **Type:** User scheduling in broadband systems, **Constraints:** Number of antennas, **Objective:** Achievable sum rate, **Solver:** VQA, **Size:** 16 antennas, 15 users, 4 subband (30 qubits).

In their VQA, they use a parametrized linear entanglement ansatz, where they combine an R_{YY} and an R_{ZX} gate in order to create entanglement, instead of a CNOT gate. As for the parameter optimization, they first use a Bayesian optimization approach in order to identify candidate solutions, which are refined via iterative local optimization. In comparison with several classical algorithms, their algorithm shows superior and near-optimal performance in terms of the solution quality.

Long et al. [130]: **Type:** Task scheduling in container terminals, **Constraints:** D, P, workload balance, **Objective:** Makespan, **Solver:** RL-enhanced QAOA, **Size:** 734 containers (up to 21 qubits).

They consider quay crane scheduling and vehicle routing. They formulate the scheduling problem as a QUBO problem and apply QAOA. As post-processing step, they apply classical optimization algorithms in order to remove infeasible or undesired solutions from the solution space, which may result from imperfect encoding. They only represent the quay crane movements as qubits, therefore, they manage to have such a few number of qubits. As for the vehicle routing problem, based on the solution of the scheduling problem, they apply a proximal policy optimization algorithm in order to find suitable trajectories.

Lopez-Ruiz et al. [125]: **Type:** JSSP, **Constraints:** A, D, P, **Objective:** Early and late delivery costs, switching costs, **Solver:** Iterative QAOA (IonQ Forte), **Size:** Up to 20 jobs and 3 machines (up to 97 qubits).

The switching costs are associated with changing production groups on the same machine. In their QAOA variant, they only use the measurement outcomes from the previous run in order to update the initial state. The number of QAOA layers and the parameter schedule are kept fixed. As for the experiments, they conclude that instances with 20 jobs and 3 machines would require 1300 binary variables. Therefore, they freeze many variables from a previously obtained optimal solution and only solve the sub-instance with their approach. Their method outperforms the LR-QAOA, which requires deeper circuits, however, it is outperformed by Gurobi.

Paul and Chakraborty [157]: **Type:** Blockchain transaction scheduling, **Constraints:** D, **Objective:** Transaction load, **Solver:** Hybrid QAOA, **Size:** No experiments.

They propose a hybrid QAOA where the scheduling problem is first decomposed into local scheduling problems. Each of these problems is tackled by one QAOA algorithm, whose numbers of layers are adjusted. Then, the problem is again decomposed into local scheduling problems. This procedure runs until convergence. The individual QAOA solvers for the local problems are interpreted as nodes in a network, and interact in order to respect constraints which involve other local schedules. This is realized by letting the individual algorithms optimize the global cost function and by delivering the cost gradient to the neighboring nodes.

7 Quantum hardware requirements

In this paper, we have reviewed only quantum and quantum-inspired algorithms, that can already be applied on noisy intermediate-scale quantum (NISQ) hardware. As the name suggests, these quantum computers operate with noisy, often called physical, qubits and low gate fidelities. According to Preskill [167], as a rule of thumb which "might be too pessimistic", a circuit of depth d requires a gate fidelity of at least $1 - 1/d$. In order to

have the potential to outperform classical algorithms, Ezratty [66] claims that NISQ decives require gate fidelities of around 99.99% with hundreds of qubits, whereas for fault-tolerant quantum computing, gate fidelities around 99.9% would suffice, for the price that the computational overhead would require millions of qubits. The reason for this is that fault-tolerant quantum computers are equipped with error correction techniques, where several physical qubits encode a single logical qubit. This enables us to apply different algorithms and in particular algorithms such as Grover's Search algorithm Grover [84], which outperform their classical counterpart. However, these limiting factors of current quantum hardware prevent the application of quantum algorithms to industry-relevant problems such as large-scale scheduling problems. Depending on the type of quantum hardware, these factors may play a different role. For instance, with ion-trap qubits, it is possible to achieve high fidelities but it is difficult to realize sufficiently many qubits, and they are around 1000 times slower than superconducting qubits (e.g., Ezratty [66], Preskill [167]).

Another limiting factor is the qubit connectivity, as a low connectivity necessitates additional SWAP gates that increase the circuit depth. The influence of the circuit depth on its fidelity can be observed for example in Kotil et al. [112]. Here, the authors apply QAOA on a real superconducting quantum computer to solve a multi-objective MaxCut problem. For this, they run the algorithm with up to 6 layers and measure the fidelity of the whole circuit, whose depth is up to 97 gates. They derive that the circuit fidelity is 13.63% for 3 layers with a circuit depth of 49, and drops to 3.71% for 6 layers.

Due to the current limitations, Leymann and Barzen [118] argue that hybrid algorithms with classical parts and shallow quantum parts are suitable for current NISQ devices. According to Lau et al. [122], in the NISQ era, quantum algorithms may only be part of some larger algorithm, which may indicate hybrid algorithms with quantum and classical subroutines.

In some works, quantum supremacy has been claimed, e.g., Arute et al. [2] using superconducting qubits and Zhong et al. [243] using photons. However, they have considered "custom-built" problems which are not of practical interest according to Lau et al. [122]. This problem is also reflected in the argumentation of Preskill [167], where the author emphasizes the commercial perspective that, even if quantum supremacy could be reached, it should hold on a "useful" task. In general, one has to be very cautious when claiming quantum supremacy as it may only be a temporary observation. See for example Tindall et al. [204], where a classical tensor network approach has been proposed which outperformed a recent quantum approach from Kim et al. [107] who use superconducting qubits.

As for scheduling problems, only a few of the reviewed papers have explicitly addressed hardware limitation problems. In QAOA applications, Vikstål et al. [211] show that the success probability of QAOA decreases with an increasing number of layers when the gates have imperfect fidelity. In order to optimize the circuit depth, Paul and Chakraborty [157], for instance, use an adaptive layer QAOA, that dynamically adjusts the number of layers, while Pires et al. [160] reduce the number of ancilla qubits to reduce the circuit depth of their QAOA implementation. Another approach by Sawaya et al. [193] is based on DQIR, which enables the optimization of the encoding in the sense that it trade-offs the qubit number with the circuit depth, which is applicable to discrete optimization problems, including JSSP. In addition to that, they consider QAOA circuits, where they assume all-to-all connectivity, and find that it is unclear how to optimize the mixers, since

the ones corresponding to deeper circuits may themselves be more efficient. Prüfer et al. [170] simulate a quantum gate error which should mimick that from an IBM hardware device, resulting in a decrease of the success probability of Grover's search so that the final state is no longer distinsguishable from a uniform distribution on the solution space.

According to Dalal et al. [55], the required circuit depth for industry-relevant problems exceeds the coherence time of a NISQ processor, which is even engraved by imperfect quantum gate operations. Therefore, they propose circuit-compression techniques using digitized counterdiabatic protocols, using digitized counterdiabatic quantum optimization (DCQO) techniques. They further implement a user-defined threshold on the angles associated with the one- or two-qubit rotation gates. If a rotaton angle of the respective gate application in the circuit is smaller than the threshold, the gate is not applied as the change is the state is negligable. Furthermore, they use hardware specific transpilation techniques to rewrite R_{YZ} and R_{ZY} rotation gates in terms of native gates applicable for superconducting circuits as well as trapped-ion hardware. With this, they further reduce the circuit depth and apply their method on IonQ's trapped ions and on IBMQ's superconducting circuits. Their results confirm a higher success probability and approximation ratio than DQA and QAOA with an equal circuit depth. As for the impact of noise, they investigate the performance of DCQO on a 16-qubit subproblem of the JSSP, revealing that in a noisy simulation, the exact solution probability is reduced to around a quarter of the success probability in an ideal simulator. However, the exact solution probability on the Aria-2 hardware of IonQ is comparable to that in the ideal simultion, thanks to an all-to-all qubit connectivity and a high two-qubit gate fidelity. As for the hDCQO, it achieves a higher exact solution probability on Aria-2 than QAOA on an ideal simulator. It is concluded that, due to the exponential decrease of the success probability with respect to the number of qubits, the standard DCQO becomes infeasible for Ising problems with more than 20 qubits. As further improvements, they mention a better bit-to-qubit encoding.

From an industry perspective, Erdmann et al. [64] mention that assignment and scheduling problems have up to around 150,000 variables. One the one hand, they state that the D-Wave hybrid solver can handle such problems, and, in some cases, even provide solutions quicker than a classical solver. On the other hand, classical solvers outperform quantum solvers in terms of solution quality for some problem types. In addition to that, they point out that due to the hybridity of the solver, it remains unclear how much performance gain has been achieved from the quantum subroutine. For a train rescheduling and rerouting problem, they apply a Benders decomposition of the MILP and achieve sufficient scalability. However, problems of this kind require a solution within a few seconds, which has not been achieved due to the overhead resulting from embedding, pre- and postprocessing steps. A gate-based solver for the binary problem was much slower than a classical solver. However, although QA was fast, the solution quality was sub-optimal. For a job shop scheduling problem, they report that a classical solver, applied to a time-indexed MILP formulation, outperformed the hybrid solver, which was applied to a corresponding QUBO problem. In addition, the solution quality of the hybrid solver was considerably lower than that of the classical solver, and the hybrid solver also required a longer runtime. Challenges arising from other industry-relevant problems considered are continuous decision variables and slack variables arising from the QUBO problem formulation of the actual MILP, in particular, inequality constraints. When incorporating inequality constraints, Müller et al. [135] point out that even their proposed representation method

called unbalanced penalization does not allow for scalability. Therefore, they suggest to focus more on hybrid approaches.

Dalal et al. [55] mention that for their JSSP setting, a non-trivial problem would require at least 360 qubits. However, Long et al. [130], who use up to 21 qubits in their experiments, state that quantum supremacy may be expected when using at least 1000 qubits and Bożejko et al. [34] argue that the practical use of quantum computing is currently not given due to the limited number of available qubits.

In summary, one can conclude that the current quantum hardware may already match some problem-specific requirements that could lead to industry-relevant applications. However, for many problems, the combination of gate fidelities, coherence times and available number of qubits are often not sufficient. In order to minimize this problem, various techniques to optimize the circuit depth and number of used qubits have been proposed. Considering the current state of the hardware, it seems reasonable to consider hybrid approaches with quantum and classical subroutines for industry-relevant problems, such as large-scale scheduling problems, and to assess whether they are applicable to such problems or even superior to pure classical algorithms.

8 Discussion and conclusion

In this work, we provided an overview of the current state of the literature concerning the application of quantum algorithms for solving scheduling problems. It turned out that the quantum-inspired evolutionary algorithms have been by far the first approaches for scheduling problems, going back to the mid 2000's. They also form the largest fraction of works in this review. The mathematically rigorous formulation of a scheduling problem as a quadratic unconstrained binary optimization problem can be translated into an Ising Hamiltonian and allows for the application of quantum annealing or other quantum optimization techniques. Those works emerged in the mid 2010's, but most of them were proposed in the past 6 years.

Although many works can be understood as proof of concept, some also consider large-scale problems that are encountered in the industry or other real-world applications. Showing that a quantum algorithm in principle works and provides good solutions for a scheduling problem is clearly a large step forward. However, a major drawback of many works is that the proposed quantum algorithms are either not compared with classical algorithms, or that the comparison is only made in terms of the solution quality or, for evolutionary algorithms, the number of generations.

Showing that, even on large problems, quantum algorithms achieve comparable performance within a comparable time, is clearly interesting for research. However, in industry, one would not deviate from classical solvers unless quantum solvers can beat their classical counterparts in terms of solution quality or computation time. So, given the often very good solution quality of classical solvers, the greater goal is to show that quantum algorithms achieve to output solutions of comparable quality in shorter time than classical solvers. This aspect has not been respected satisfactorily in many works, as one either does only compare the solution quality or the time, apart from the fact that some works even only report pure QPU times, not the total time. It has been pointed out in Shimada et al. [194] that for a fair comparison, it is necessary to measure the total time at least from the point where the input data files are read, as the data preparation time also needs to be accounted for.

In addition to that, not all works have used the commercially available quantum solvers but simulated ones on classical hardware, such as a Digital Annealer, or even own implementations, in particular for QIEAs. For such works, one has to put into question whether they would achieve a comparable performance on quantum hardware.

Some works, especially those proposing a QIEA, compared the performance of their algorithm with that of classical EAs. For a fair comparison, they fix the same upper bound on the computational time for all algorithms. This can be put into question when it comes to using quantum algorithms in industry, as achieving slightly better solutions within the same time may be less interesting than achieving solutions of comparable quality in (much) shorter time. From a research perspective, one should ask why an improvement in solution quality is possible in the first place, as classical algorithms often already lead to excellent solutions. This may be partly explained by the fact that when the competitors are EAs such algorithms tend to be hard to tune. Therefore, future work should consider additional comparisons between QIEAs and classical MILP or QUBO solvers, as well as quantum solvers for QUBO problems with classical EAs.

Another crucial aspect is the question how to set the hyperparameters. In the literature, one can observe that in some works, different hyperparameter settings for the proposed quantum algorithm are compared beforehand, and the best are then used for a comparison with a classical algorithm. In other works, hyperparameters are just fixed or default configurations are used. It is desirable that either the costs for finding good hyperparameters are respected when comparing quantum and classical algorithms, or some kind of hyperparameter robustness in the sense that the quantum algorithm achieves similar behavior for different hyperparameter configurations has to be established.

For future work, the following aspects should be accounted for:

- honest reporting of the total computation time, including qubit preparation, hyperparameter optimization and actual QPU time,
- usage of real quantum hardware for the quantum algorithm in order to include realistic noise into the experiments,
- comparison of the solution quality in terms of the achieved objective function,
- comparison of different quantum algorithms (e.g., a quantum QUBO problem solver and a QIEA) with a classical EA and a classical MILP or QUBO problem solver,
- comparison with scheduling policies.

The last aspect refers to a very serious competitor of quantum algorithms, which is also applicable to scheduling problems and which has not been considered in nearly all the reviewed works: Artificial Intelligence (AI). In particular, reinforcement learning has been applied in order to learn a policy for scheduling problems, see Gu et al. [82], Kayhan and Yildiz [115], Zhang and Zhu [248] for an overview. Of course, for algorithms for static optimization problems (where a fixed time horizon is used), the problem must be formulated for the given time horizon, solved, and formulated anew for the next time horizon. In contrast, an AI algorithm learns a policy that is operational after training. Therefore, those approaches cannot be directly compared. One should however compare the performance of such a policy on a given scheduling problem in terms of the quality of the selected schedule with the solution of an optimization algorithm on the same problem. If the policy can compete, it most likely beats the static variant in terms of the required computation time for training and schedule selection over a long horizon. In this case, a fair comparison would require quantum reinforcement learning algorithms Dong et al.

[49] for scheduling problems. See Kim et al. [106] for a recent work on quantum reinforcement learning for scheduling aerial vehicles. An interesting line of research has recently emerged, where hybrid algorithms with a quantum part and an AI part have been proposed, see for example Cuong et al. [42]. Here, QAOA parameters are learned via a deep RL policy. Another example is Mahjoub et al. [137], where an RL part decides whether to explore or to exploit and where a quantum part enhances diversity.

Nevertheless, one should not expect that pure quantum algorithms will be used to tackle large industry problems on their own in the near future. Instead, hybrid algorithms where a nasty sub-problem such as a combinatorial problem is solved by a quantum algorithm, while other aspects of the problem are solved classically could be utilized. This outlook encourages to pursue decomposition strategies of large-scale optimization problems and further research on more qubit-efficient encoding schemes.

Abbreviations

ADAM, Adaptive moment estimation; BQM, Binary quadratic model; CP, Constraint programming; CQM, Constrained quadratic model; DCQO, Digitized counterdiabatic quantum optimization; DQM, Discrete quadratic model; EA, Evolutionary algorithm; FSSP, Flow shop scheduling problem; FJSSP, Flexible job shop scheduling problem; FOSSP, Flexible open shop scheduling problem; GA, Genetic algorithm; hDCQO, Hybrid digitized counterdiabatic quantum optimization; HOBO, Higher-order binary optimization; IA, Immune algorithm; IMSP, Identical machines scheduling problem; JSSP, Job shop scheduling problem; MILP, Mixed-integer linear program; MINLP, Mixed-integer non-linear program; MIP, Mixed-integer program; NISQ, Noisy intermediate-scale quantum; NW-FSSP, No-wait flow shop scheduling problem; OSSP, Open shop scheduling problem; PGA, Permutation genetic algorithm; PSO, Particle swarm optimization; QA, Quantum annealing; QAOA, Quantum approximate optimization algorithm; QIEA, Quantum-inspired evolutionary algorithm; QIGA, Quantum-inspired genetic algorithm; QIIA, Quantum-inspired immune algorithm; QUBO, Quadratic unconstrained binary optimization; RA, Reverse annealing; RL, Reinforcement learning; SA, Simulated annealing; SoC, State of charge; SMSP, Single-machine scheduling problem; UMSP, Unrelated machines scheduling problem; VNS, Variable neighborhood search; VQE, Variational quantum eigensolver; WSP, Workshop scheduling problem.

Acknowledgements

We would like to thank an anonymous referee for helpful comments that helped to improve the quality of the paper. Furthermore, we would like to thank Matthias Zimmermann for fruitful discussions and comments.

Author contributions

TW had the idea for this paper, conducted the literature research, wrote major parts of the text, and made the figures. FU provided background knowledge about quantum physics and quantum computing, and contributed mainly to Sect. 3 and the structure.

Funding information

Open Access funding enabled and organized by Projekt DEAL. This project was made possible by the DLR Quantum Computing Initiative and the Federal Ministry for Research, Technology and Space; qci.dlr.de/projects/qcmobility/.

Data availability

No datasets were generated or analysed during the current study.

Declarations

Competing interests

The authors declare no competing interests.

Author details

[1] Institute of Systems Engineering for Future Mobility, German Aerospace Center (DLR), Escherweg 2, Oldenburg, 26129, Lower Saxony, Germany. [2] Institute of Quantum Technologies, German Aerospace Center (DLR), Wilhelm-Runge-Strasse 10, Ulm, 89081, Baden-Württemberg, Germany.

Received: 1 July 2025 Accepted: 5 March 2026 Published online: 19 March 2026

References

1. Abdalkareem ZA, Amir A, Al-Betar MA, Ekhan P, Hammouri AI. Healthcare scheduling in optimization context: a review. Health Technol. 2021;11:445–69.
2. Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, Biswas R, Boixo S, Brandao FG, Buell DA, et al. Quantum supremacy using a programmable superconducting processor. Nature. 2019;574(7779):505–10.
3. Adams J, Balas E, Zawack D. The shifting bottleneck procedure for job shop scheduling. Manag Sci. 1988;34(3):391–401.

4. Aggoune R, Deleplanque S. Solving the job shop scheduling problem: QUBO model and quantum annealing. In: Emerging optimization methods: from metaheuristics to quantum approaches. 2023.
5. Aggoune R, Deleplanque S. Addressing machine unavailability in job shop scheduling: a quantum computing approach. In: 15th metaheuristics international conference MIC 2024. 2024.
6. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH. The arithmetic optimization algorithm. Comput Methods Appl Mech Eng. 2021;376:113609.
7. Allahverdi A. The third comprehensive survey on scheduling problems with setup times/costs. Eur J Oper Res. 2015;246(2):345–78.
8. Allahverdi A. A survey of scheduling problems with no-wait in process. Eur J Oper Res. 2016;255(3):665–86.
9. Anand E, Panneerselvam R, et al. Literature review of open shop scheduling problems. Intell Inf Manag. 2015;7(01):33.
10. Acharya B, Panda S, Das S, Majhi SK, Gerogiannis VC, Kanavos A. Optimizing task scheduling in cloud environments: a hybrid golden search whale optimization algorithm approach. Neural Comput Appl. 2025;1–23.
11. Acharya B, Panda S, Ray NK. Multiprocessor task scheduling optimization for cyber-physical system using an improved salp swarm optimization algorithm. SN Comput Sci. 2024;5(1):184.
12. Alam T, Raza Z. Quantum genetic algorithm based scheduler for batch of precedence constrained jobs on heterogeneous computing systems. J Syst Softw. 2018;135:126–42.
13. Amaro D, Rosenkranz M, Fitzpatrick N, Hirano K, Fiorentini M. A case study of variational quantum algorithms for a job shop scheduling problem. EPJ Quantum Technol. 2022;9(1):5.
14. Bean JC. Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput. 1994;6(2):154–60.
15. Benders J. Partitioning procedures for solving mixed-variables programming problems. Numer Math. 1962;4(1):238–52.
16. Bourreau E, Fleury G, Lacomme P. Indirect job-shop coding using rank: application to QAOA (IQAOA). 2024. arXiv preprint. arXiv:2402.18280.
17. Baioletti M, Fagiolo F, Oddi A, Rasconi R. A variational quantum algorithm for the permutation flow shop scheduling problem. In: Proceedings of the genetic and evolutionary computation conference companion. 2025. p. 2389–95.
18. Bittner T, Groppe S. Avoiding blocking by scheduling transactions using quantum annealing. In: Proceedings of the 24th symposium on international database engineering & applications. 2020. p. 1–10.
19. Bittner T, Groppe S. Hardware accelerating the optimization of transaction schedules via quantum annealing by avoiding blocking. Open J Cloud Comput (OJCC). 2020;7(1):1–21.
20. Bierwirth C. A generalized permutation approach to job shop scheduling with genetic algorithms. OR Spektrum. 1995;17(2):87–92.
21. Bożejko W, Klempous R, Pempera J, Rozenblit JW, Smutnicki C, Uchroński M, Wodecki M. Optimal solving of a scheduling problem using quantum annealing metaheuristics on the D-Wave quantum solver. 2023.
22. Binkowski L, Koßmann G, Tutschku C, Schwonnek R. Symmetry-based quantum algorithms for open-shop scheduling with hard constraints. Acad Quantum. 2025;2(3).
23. Bożejko W, Klempous R, Uchroński M, Wodecki M. Solving two-machine flow shop scheduling problem with total weighted number of on-time tasks maximization criterion on D-Wave quantum computer. In: 2023 IEEE 23rd international symposium on computational intelligence and informatics (CINTI). New York: IEEE Press; 2023. p. 000037–000040.
24. Belmahdi R, Mechta D, Harous S, Bentaleb A. SQGA: quantum genetic algorithm-based workflow scheduling in fog-cloud computing. In: 2022 international wireless communications and mobile computing (IWCMC). New York: IEEE Press; 2022. p. 131–6.
25. Boutekkouk F, Oubadi S. Real time tasks scheduling optimization using quantum inspired genetic algorithms. In: Artificial intelligence perspectives in intelligent systems: proceedings of the 5th computer science on-line conference 2016 (CSOC2016). vol. 1. Berlin: Springer; 2016. p. 69–80.
26. Baioletti M, Oddi A, Rasconi R. Solving scheduling problems with quantum computing: a study on flexible open shop. In: Proceedings of the companion conference on genetic and evolutionary computation. 2023. p. 2175–8.
27. Bożejko W, Pempera J, Uchroński M, Wodecki M. Distributed quantum annealing on d-wave for the single machine total weighted tardiness scheduling problem. In: International conference on computational science. Berlin: Springer; 2022. p. 171–8.
28. Bożejko W, Pempera J, Uchroński M, Wodecki M. Determination of the lower bounds of the goal function for a single-machine scheduling problem on D-Wave quantum annealer. In: International conference on computational science. Berlin: Springer; 2023. p. 201–8.
29. Bożejko W, Pempera J, Uchroński M, Wodecki M. Quantum annealing-driven branch and bound for the single machine total weighted number of tardy jobs scheduling problem. Future Gener Comput Syst. 2024.
30. Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res. 1993;41(3):157–83.
31. Bajracharya R, Shakya S. Task scheduling optimization in the cloud using quantum annealing. In: 2023 7th international conference on I-SMAC (IoT in social, mobile, analytics and cloud) (I-SMAC). New York: IEEE Press; 2023. p. 385–91.
32. Botelho L, Salehi Ö. Fixed interval scheduling problem with minimal idle time with an application to music arrangement problem. 2023. arXiv preprint. arXiv:2310.14825.
33. Bottarelli A, Schmitt S, Hauke P. Inequality constraints in variational quantum circuits with qudits. 2024. arXiv preprint. arXiv:2410.07674.
34. Bożejko W, Trotskyi S, Uchroński M, Wodecki M. Optimizing two-machine scheduling in flexible manufacturing systems using autonomous ai and quantum computing. Neurocomputing. 2025;132067.
35. Bożejko W, Uchroński M, Wodecki M. Non-crossing neighborhood searching on quantum computer for a single machine scheduling problem. IFAC-PapersOnLine. 2025;59(24):19–23.
36. Boyd S, Vandenberghe L. Convex optimization. Cambridge: Cambridge university press; 2004.
37. Carlier J. Ordonnancements a contraintes disjonctives. RAIRO Oper Res. 1978;12(4):333–50.
38. Chen C-A, Chien C-F, Kuo H-A. Hybrid quantum annealing genetic algorithm with auxiliary resource dispatching for TFT-LCD array photolithography scheduling and an empirical study. Comput Ind Eng. 2025;203:110989.

39. Carugno C, Ferrari Dacrema M, Cremonesi P. Evaluating the job shop scheduling problem on a D-wave quantum annealer. Sci Rep. 2022;12(1):6539.
40. Ciacco A, Guerriero F, Macrina G. Review of quantum algorithms for medicine, finance and logistics. Soft Comput. 2025;29(4):2129–70.
41. Chaudhry IA, Khan AA. A research survey: review of flexible job shop scheduling techniques. Int Trans Oper Res. 2016;23(3):551–91.
42. Cuong TN, Kim H-S, You S-S, Tan ND, et al. Deep learning-enhanced quantum optimization for integrated job scheduling in container terminals. Eng Appl Artif Intell. 2025;148:110431.
43. Cadavid AG, Montalban I, Dalal A, Solano E, Hegade NN. Efficient digitized counterdiabatic quantum optimization algorithm within the impulse regime for portfolio optimization. Phys Rev Appl. 2024;22(5):054037.
44. Choi J, Oh S, Kim J. Quantum approximation for wireless scheduling. Appl Sci. 2020;10(20):7116.
45. Chen Z-S, Tan Y, Ma Z, Zhu Z, Skibniewski MJ. Unlocking the potential of quantum computing in prefabricated construction supply chains: Current trends, challenges, and future directions. Inf Fusion. 2025;103043.
46. Christeson T, Ullah MH, Arabnya A, Khodaei A, Fan R. Hybrid quantum-classical optimization of the resource scheduling problem. 2025. arXiv preprint. arXiv:2511.00733.
47. Chen Z, Zheng X, Zhou S, Liu C, Chen H. Quantum-inspired ant colony optimisation algorithm for a two-stage permutation flow shop with batch processing machines. Int J Prod Res. 2020;58(19):5945–63.
48. Davarzani Z, Akbarzadeh-T M-R. A novel quantum immune algorithm for multiobjective flexible job shop scheduling. Int J Artif Intell Tools. 2014;23(05):1450006.
49. Dong D, Chen C, Li H, Tarn T-J. Quantum reinforcement learning. IEEE Trans Syst Man Cybern, Part B, Cybern. 2008;38(5):1207–20.
50. Durr C, Hoyer P. A quantum algorithm for finding the minimum. 1996. arXiv preprint. arXiv:quant-ph/9607014.
51. Dalyac C, Henriet L, Jeandel E, Lechner W, Perdrix S, Porcheron M, Veshchezerova M. Qualifying quantum approaches for hard industrial optimization problems. A case study in the field of smart-charging of electric vehicles. EPJ Quantum Technol. 2021;8(1):12.
52. Durasevic M, Jakobovic D. Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey. Artif Intell Rev. 2023;56(4):3181–289.
53. Domino K, Koniorczyk M, Krawiec K, Jałowiecki K, Gardas B. Quantum computing approach to railway dispatching and conflict management optimization on single-track railway lines. 2020. arXiv preprint. arXiv:2010.08227.
54. Domino K, Kundu A, Salehi Ö, Krawiec K. Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing. Quantum Inf Process. 2022;21(9):337.
55. Dalal A, Montalban I, Hegade NN, Cadavid AG, Solano E, Awasthi A, Vodola D, Jones C, Weiss H, Füchsel G. Digitized counterdiabatic quantum algorithms for logistics scheduling. Phys Rev Appl. 2024;22(6):064068.
56. Dauzère-Pérès S, Paulli J. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann Oper Res. 1997;70:281–306.
57. Denkena B, Schinkel F, Pirnay J, Wilmsmeier S. Quantum algorithms for process parallel flexible job shop scheduling. CIRP J Manuf Sci Technol. 2021;33:100–14.
58. Dai S, Saurabh N, Wang Q, Nian J, Kan S, Mao Y, Cheng L. Quantum reinforcement learning for QoS-aware real-time job scheduling in cloud systems. IEEE Syst J. 2025.
59. Dominguez F, Unger J, Traube M, Mant B, Ertler C, Lechner W. Encoding-independent optimization problem formulation for quantum computing. Front Quantum Sci Technol. 2023;2:1229471.
60. Dantzig GB, Wolfe P. Decomposition principle for linear programs. Oper Res. 1960;8(1):101–11.
61. D-Wave. Hybrid solver for constrained quadratic models. D-Wave Syst. Inc., Burnaby, BC, Canada, Tech. Rep. 2021.
62. Deng G, Wei M, Su Q, Zhao M. An effective co-evolutionary quantum genetic algorithm for the no-wait flow shop scheduling problem. Adv Mech Eng. 2015;7(12):1687814015622900.
63. Das K, Zaman S, Sadhu A, Banerjee A, Khan S. Quantum annealing for solving a nurse-physician scheduling problem in Covid-19 clinics. 2020. viXra.
64. Erdmann M, Karch L, Awasthi A, Jones CI, Bhardwaj P, Krellner F, Stein J, Linnhoff-Popien C, Kraus N, Eder P, et al. Quantum computing–strategic recommendations for the industry. 2026. arXiv preprint. arXiv:2601.08578.
65. Erdős P, Rényi A. On the strength of connectedness of a random graph. Acta Math Hung. 1961;12(1):261–7.
66. Ezratty O. Where are we heading with NISQ? 2023. arXiv preprint. arXiv:2305.09518.
67. Farhani Y, Arbaoui T, Benatchba K. Manual vs. automated QUBO formulations for flow shop scheduling: a comparative study on D-wave and InfinityQ. In: Proceedings of the genetic and evolutionary computation conference companion. 2025. p. 2424–32.
68. Fang Z. A quantum immune algorithm for multiobjective parallel machine scheduling. In: Advances in swarm intelligence: first international conference, ICSI 2010. Beijing, China, June 12-15, 2010. Proceedings, part I 1. Berlin: Springer; 2010. p. 321–7.
69. Farhi E, Goldstone J, Gutmann S. A quantum approximate optimization algorithm. 2014. arXiv preprint. arXiv:1411.4028.
70. Fu K, Liu J, Chen M, Zhang H. Solving flexible job-shop scheduling problems based on quantum computing. Entropy. 2025;27(2):189.
71. Farhani Y, Saiem M, Arbaoui T, Hnaien F. Enhanced QUBO formulations for the flow shop scheduling problem. In: Proceedings of the genetic and evolutionary computation conference companion. 2024. p. 1942–9.
72. Gandhi T, Alam T, et al. Quantum genetic algorithm with rotation angle refinement for dependent task scheduling on distributed systems. In: 2017 tenth international conference on contemporary computing (IC3). New York: IEEE Press; 2017. p. 1–5.
73. Groppe S, Groppe J. Optimizing transaction schedules on universal quantum computers via code generation for Grover's search algorithm. In: Proceedings of the 25th international database engineering & applications symposium. 2021. p. 149–56.
74. Goh ST, Gopalakrishnan S, Bo J, Lau HC. A hybrid framework using a qubo solver for permutation-based combinatorial optimization. 2020. arXiv preprint. arXiv:2009.12767.
75. Gu J, Gu M, Cao C, Gu X. A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. Comput Oper Res. 2010;37(5):927–37.

76. Gu J, Gu M, Gu X, et al. A mutualism quantum genetic algorithm to optimize the flow shop scheduling with pickup and delivery considerations. Math Probl Eng. 2015;2015.
77. Gu J, Gu X, Jiao B. A quantum genetic based scheduling algorithm for stochastic flow shop scheduling problem with random breakdown. IFAC Proc Vol. 2008;41(2):63–8.
78. Gu J, Gu X, Jiao B. Solving stochastic earliness and tardiness parallel machine scheduling using quantum genetic algorithm. In: 2008 7th world congress on intelligent control and automation. New York: IEEE Press; 2008. p. 4154–9.
79. Geitz M, Grozea C, Steigerwald W, Stöhr R, Wolf A. Solving the extended job shop scheduling problem with AGVs–classical and quantum approaches. In: International conference on integration of constraint programming, artificial intelligence, and operations research. Berlin: Springer; 2022. p. 120–37.
80. Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Math Oper Res. 1976;1(2):117–29.
81. Glos A, Kundu A, Salehi Ö. Optimizing the production of test vehicles using hybrid constrained quantum annealing. SN Comput Sci. 2023;4(5):609.
82. Gu Y, Liu Z, Dai S, Liu C, Wang Y, Wang S, Theodoropoulos G, Cheng L. Deep reinforcement learning for job scheduling and resource management in cloud computing: an algorithm-level review. 2025. arXiv preprint. arXiv:2501.01007.
83. Grange C, Poss M, Bourreau E, T'kindt V, Ploton O. Moderate exponential-time quantum dynamic programming across the subsets for scheduling problems. Eur J Oper Res. 2025;320(3):516–26.
84. Grover LK. A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing. 1996. p. 212–9.
85. Galavani S, Younesi A, Ansari M. QIGA: quantum-inspired genetic algorithm for dynamic scheduling in mobile edge computing. In: 2025 29th international computer conference, computer society of Iran (CSICC). New York: IEEE Press; 2025. p. 1–5.
86. Hegade NN, Chen X, Solano E. Digitized counterdiabatic quantum optimization. Phys Rev Res. 2022;4(4):042030.
87. Hsu C-H, Ding Z. Variational quantum algorithm for user scheduling in broadband MU-MIMO systems. IEEE Trans Wirel Commun. 2025;99:1–1.
88. Han K-H, Kim J-H. Genetic quantum algorithm and its application to combinatorial optimization problem. In: Proceedings of the 2000 congress on evolutionary computation. CEC00 (cat. no. 00TH8512). vol. 2. New York: IEEE Press; 2000. p. 1354–60.
89. Han K-H, Kim J-H. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans Evol Comput. 2002;6(6):580–93.
90. Hamada N, Saito K, Kawashima H. Practical effectiveness of quantum annealing for shift scheduling problem. In: 2022 IEEE international parallel and distributed processing symposium workshops (IPDPSW). New York: IEEE Press; 2022. p. 01–4.
91. Hamada N, Saito K, Kawashima H. Applying quantum annealing for shift scheduling problem for call centers. Int J Network Comput. 2023;13(1):2–17. https://doi.org/10.15803/ijnc.13.1_2.
92. Herman D, Shaydulin R, Sun Y, Chakrabarti S, Hu S, Minssen P, Rattew A, Yalovetzky R, Pistoia M. Constrained optimization via quantum zeno dynamics. Commun Phys. 2023;6(1):219.
93. Hussain M, Wei L-F, Abbas F, Rehman A, Ali M, Lakhan A. A multi-objective quantum-inspired genetic algorithm for workflow healthcare application scheduling with hard and soft deadline constraints in hybrid clouds. Appl Soft Comput. 2022;128:109440.
94. Hussain M, Wei L-F, Rehman A, Ali M, Waqas SM, Abbas F. Cost-aware quantum-inspired genetic algorithm for workflow scheduling in hybrid clouds. J Parallel Distrib Comput. 2024;191:104920.
95. Ikeda K, Nakamura Y, Humble TS. Application of quantum annealing to nurse scheduling problem. Sci Rep. 2019;9(1):12837.
96. Jain R, Jain P, Tyagi B. A model for SLA aware admission control and QoS aware task scheduling in cloud environment. Peer-to-Peer Netw Appl. 2025;18(4):197.
97. Jing Y, Liu Y, Bi M. Quantum-inspired immune clonal algorithm for railway empty cars optimization based on revenue management and time efficiency. Clust Comput. 2019;22:545–54.
98. Jayaraman S, Premkumar R, Raguraman H, Visuwasam LMM, et al. A quantum-assisted framework for energy-efficient cloud task scheduling using the quantum approximate optimization algorithm and variational quantum eigensolver. In: 2025 international conference on computing technologies & data communication (ICCTDC). New York: IEEE Press; 2025. p. 1–6.
99. Jain R, Sharma N. A quantum inspired hybrid SSA–GWO algorithm for SLA based task scheduling to improve QoS parameter in cloud computing. Clust Comput. 2023;26(6):3587–610.
100. Joseph I, Shi Y, Porter M, Castelli A, Geyko V, Graziani F, Libby S, DuBois J. Quantum computing for fusion energy science applications. Phys Plasmas. 2023;30(1).
101. Jiang Y, Wan S. Parallel flow shop scheduling problem using quantum algorithm. In: Applied informatics and communication: international conference, ICAIC 2011. Xi'an, China, August 20-21, 2011 Proceedings, part V. Berlin: Springer; 2011. p. 269–74.
102. Ji B, Yuan X, Yuan Y. A hybrid intelligent approach for co-scheduling of cascaded locks with multiple chambers. IEEE Trans Cybern. 2018;49(4):1236–48.
103. Ji B, Yuan X, Yuan Y, Lei X, Fernando T, Iu HH. Exact and heuristic methods for optimizing lock-quay system in inland waterway. Eur J Oper Res. 2019;277(2):740–55.
104. Krellner F, Awasthi A, Kraus N, Braun S, Poppel M, Porawski D. Solving a real-world modular logistic scheduling problem with a quantum-classical metaheuristics. 2025. arXiv preprint. arXiv:2507.21701.
105. Konar D, Bhattacharyya S, Sharma K, Sharma S, Pradhan SR. An improved hybrid quantum-inspired genetic algorithm (HQIGA) for scheduling of real-time task in multiprocessor system. Appl Soft Comput. 2017;53:296–307.
106. Kim GS, Cho Y, Chung J, Park S, Jung S, Han Z, Kim J. Quantum multi-agent reinforcement learning for cooperative mobile access in space-air-ground integrated networks. 2024. arXiv preprint. arXiv:2406.16994.
107. Kim Y, Eddins A, Anand S, Wei KX, Van Den Berg E, Rosenblatt S, Nayfeh H, Wu Y, Zaletel M, Temme K, et al. Evidence for the utility of quantum computing before fault tolerance. Nature. 2023;618(7965):500–5.

108. Kedziera E, Gamon W, Koniorczyk M, Mzaouali Z, Galadíková A, Domino K. Quantum and classical algorithms for daily railway rolling stock circulation plans. 2025. arXiv preprint. arXiv:2512.19340.
109. Kacem I, Hammadi S, Borne P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Trans Syst Man Cybern, Part C, Appl Rev. 2002;32(1):1–13.
110. Koniorczyk M, Krawiec K, Botelho L, Bešinović N, Domino K. Solving rescheduling problems in heterogeneous urban railway networks using hybrid quantum–classical approach. J Rail Transp Plan Manag. 2025;34:100521.
111. Klug F. Quantum optimization algorithms in operations research: Methods, applications, and implications. 2023. arXiv preprint. arXiv:2312.13636.
112. Kotil A, Pelofske E, Riedmüller S, Egger DJ, Eidenbenz S, Koch T, Woerner S. Quantum approximate multi-objective optimization. Nat Comput Sci. 2025;1–10.
113. Kurowski K, Pecyna T, Slysz M, Różycki R, Waligóra G, Węglarz J. Application of quantum approximate optimization algorithm to job shop scheduling problem. Eur J Oper Res. 2023;310(2):518–28.
114. Kaipu Y, Tao N, et al. Cloud-based adaptive quantum genetic algorithm for solving flexible job shop scheduling problem. In: 2018 IEEE 3rd international conference on cloud computing and Internet of things (CCIOT). New York: IEEE Press; 2018. p. 30–4.
115. Kayhan BM, Yildiz G. Reinforcement learning applications to machine scheduling problems: a comprehensive literature review. J Intell Manuf. 2023;34(3):905–29.
116. Kroczek M, Zawalska J, Rycerz K. Workflow decomposition algorithm for scheduling with quantum annealer-based hybrid solver. 2025. arXiv preprint. arXiv:2506.01567.
117. Lilhore UK, Alex S, Paul V, Puthan Purayil R, Aldossary SMA, Simaiya S, Mohamed HG, Khan M, et al. QHRMOF: a quantum-inspired hybrid multi-objective framework for energy-efficient task scheduling and load balancing in cloud computing. J Cloud Comput. 2025;14(1):1–38.
118. Leymann F, Barzen J. The bitter truth about gate-based quantum algorithms in the NISQ era. Quantum Sci Technol. 2020;5(4):044007.
119. Lahoz-Beltra R. The conquest of quantum genetic algorithms: the adventure to cross the valley of death. 2023. arXiv preprint. arXiv:2401.08631.
120. Lehmer DH. Teaching combinatorial tricks to a computer. In: Proceedings of symposia in applied mathematics. Providence: Am. Math. Soc.; 1960. p. 179–93.
121. Larsson N, Kanerot S. Quantum optimization of physician scheduling for maximal healthcare capacity. 2025.
122. Lau JWZ, Lim KH, Shrotriya H, Kwek LC. NISQ computing: where are we and where do we go? AAPPS Bull. 2022;32(1):27.
123. Lei W, Manier H, Manier M-A, Wang X, et al. A hybrid quantum evolutionary algorithm with improved decoding scheme for a robotic flow shop scheduling problem. Math Probl Eng. 2017;2017.
124. Li Y, Ma J, Xie Z, Hu Z, Shen X, Zhang K. A scheduling method for heterogeneous signal processing platforms based on quantum genetic algorithm. Appl Sci. 2023;13(7):4428.
125. Lopez-Ruiz MA, Tucker EL, Arnold EM, Epifanovsky E, Kaushik A, Roetteler M. A non-variational quantum approach to the job shop scheduling problem. 2025. arXiv preprint. arXiv:2510.26859.
126. Li M, Song X. An improved quantum genetic algorithm for the resource constrained project scheduling. In: 2016 4th international conference on electrical & electronics engineering and computer science (ICEEECS 2016). Atlantis Press; 2016. p. 612–7.
127. Leib D, Seidel T, Jäger S, Heese R, Jones C, Awasthi A, Niederle A, Bortz M. An optimization case study for solving a transport robot scheduling problem on quantum-hybrid and quantum-inspired hardware. Sci Rep. 2023;13(1):18743.
128. Lucas A. Ising formulations of many NP problems. Front Phys. 2014;2:5.
129. Li B-B, Wang L. A hybrid quantum-inspired genetic algorithm for multi-objective scheduling. In: International conference on intelligent computing. Berlin: Springer; 2006. p. 511–22.
130. Long LNB, You S-S, Kim H-S, Cuong TN, Nguyen DA, Tan ND, et al. Multiple equipment scheduling and routing for container terminals using quantum-inspired deep learning. Appl Soft Comput. 2025;114530.
131. Liu M, Yi S, Wen P. Quantum-inspired hybrid algorithm for integrated process planning and scheduling. Proc Inst Mech Eng, B J Eng Manuf. 2018;232(6):1105–22.
132. Latif MS, Zhou H, Amir M. A hybrid quantum estimation of distribution algorithm (Q-EDA) for flow-shop scheduling. In: 2013 ninth international conference on natural computation (ICNC). New York: IEEE Press; 2013. p. 654–8.
133. Latif MS, Zhou H, Ali A. A novel variant of QGA with VNS for flowshop scheduling problem. J Comput. 2014;9(9):2191–7.
134. Mohammadbagherpoor H, Dreher P, Ibrahim M, Oh Y-H, Hall J, Stone RE, Stojkovic M. Exploring airline gate-scheduling optimization using quantum computers. 2021. arXiv preprint. arXiv:2111.09472.
135. Müller S, Dukalski M, Phillipson F. Quantum annealing for optimizing unit scheduling in renewable energy systems: Formulation and evaluation. IEEE Trans Power Syst. 2025.
136. Misra SK, Kuila P. Energy-efficient task scheduling using quantum-inspired genetic algorithm for cloud data center. In: Advanced computational paradigms and hybrid intelligent computing: proceedings of ICACCP 2021. Berlin: Springer; 2022. p. 467–77.
137. Mahjoub A, Khalilian M, Mohammadzadeh J. QQLAOA: task scheduling with multi-objectives quantum mutation and Q-learning based arithmetic optimizer algorithm in cloud data centers. Computing. 2025;107(4):109.
138. Morita S, Nishimori H. Mathematical foundation of quantum annealing. J Math Phys. 2008;49(12).
139. Mori AM. Replanning flight schedules using quantum computing. Master's thesis. Portugal: Universidade do Porto; 2022.
140. Mishra K, Pradhan R, Majhi SK. Quantum-inspired binary chaotic salp swarm algorithm (QBCSSA)-based dynamic task scheduling for multiprocessor cloud computing systems. J Supercomput. 2021;77:10377–423.
141. Motta M, Rice JE. Emerging quantum computing algorithms for quantum chemistry. Wiley Interdiscip Rev Comput Mol Sci. 2022;12(3):1580.
142. Murata T. Petri nets: properties, analysis and applications. Proc IEEE. 1989;77(4):541–80.

143. Mo Z, Wu G, He Y, Liu H. Quantum genetic algorithm for scheduling jobs on computational grids. In: 2010 international conference on measuring technology and mechatronics automation. vol. 2. New York: IEEE Press; 2010. p. 964–7.
144. Nielsen MA, Chuang IL. Quantum computation and quantum information: 10th anniversary edition. Cambridge: Cambridge University Press; 2010. https://doi.org/10.1017/CBO9780511976667.
145. Naik BB, Priyanka B, Ansari SA. Energy-efficient task offloading and efficient resource allocation for edge computing: a quantum inspired particle swarm optimization approach. Clust Comput. 2025;28(3):155.
146. Nayak N, Prisacaru A, Çalıkyılmaz U, Groppe J, Groppe S. Quantum-enhanced transaction scheduling with reduced complexity via solving qubo iteratively using a locking mechanism. In: Proceedings of the 2nd workshop on quantum computing and quantum-inspired technology for data-intensive systems and applications. 2025. p. 26–35.
147. Narayanan A, Shmatikov V. Robust de-anonymization of large sparse datasets. In: 2008 IEEE symposium on security and privacy (sp 2008). New York: IEEE Press; 2008. p. 111–25.
148. Ning T, Wang Z, Duan X, Liu X. Research on flexible job shop scheduling with low-carbon technology based on quantum bacterial foraging optimization. Int J Low Carbon Technol. 2021;16(3):761–9.
149. Niu Q, Zhou T, Ma S. A quantum-inspired immune algorithm for hybrid flow shop with makespan criterion. J Univers Comput Sci. 2009;15(4):765–85.
150. Niu Q, Zhou F, Zhou T. Quantum genetic algorithm for hybrid flow shop scheduling problems to minimize total completion time. In: International conference on intelligent computing for sustainable energy and environment. Berlin: Springer; 2010. p. 21–9.
151. Ossorio-Castillo J, Pena-Brage F. Optimization of a refinery scheduling process with column generation and a quantum annealer. Optim Eng. 2022;23(3):1471–88.
152. Ohzeki M. Breaking limitation of quantum annealer in solving optimization problems under constraints. Sci Rep. 2020;10(1):3126.
153. Ou C-H, Lin Y-C, Chen W-Y, Wu K-C. Solving the cruise passenger itinerary scheduling problem using quantum-inspired computing. In: 2025 international conference on quantum communications, networking, and computing (QCNC). New York: IEEE Press; 2025. p. 205–9.
154. Orts F, Puertas A, Ortega G, Garzón E. Quantum annealing solution for the unrelated parallel machine scheduling with priorities and delay of task switching on machines. Future Gener Comput Syst. 2023;148:514–23.
155. Pérez Armas LF, Creemers S, Deleplanque S. Solving the resource-constrained project scheduling problem (rcpsp) with quantum annealing. 2024. Available at SSRN 4689017.
156. Pérez Armas LF, Deleplanque S, Aggoune R, Creemers S. A hybrid column generation-based heuristic for solving the parallel machine scheduling problem with sequence-dependent set-up times. Philos Trans R Soc A, Math Phys Eng Sci. 2025;383:2310
157. Paul S, Chakraborty S. Hybrid quantum approximate optimization algorithm (HQAOA) for efficient blockchain transaction scheduling. In: 2025 3rd international conference on intelligent systems, advanced computing and communication (ISACC). New York: IEEE Press; 2025. p. 864–9.
158. Pereira CS, Dias DM, Martí L, Vellasco M. A multi-objective decomposition optimization method for refinery crude oil scheduling through genetic programming. In: Proceedings of the companion conference on genetic and evolutionary computation. 2023. p. 1972–80.
159. Pereira CS, Dias DM, Pacheco MAC, Vellasco MMR, Cruz AVA, Hollmann EH. Quantum-inspired genetic programming algorithm for the crude oil scheduling of a real-world refinery. IEEE Syst J. 2020;14(3):3926–37.
160. Pires OM, Santiago R, Marchi J. Two stage quantum optimization for the school timetabling problem. In: 2021 IEEE congress on evolutionary computation (CEC). New York: IEEE Press; 2021. p. 2347–53.
161. Pereira CS, Dias DM, Vellasco MM, Viana FHF, Martí L. Crude oil refinery scheduling: addressing a real-world multiobjective problem through genetic programming and dominance-based approaches. In: Proceedings of the genetic and evolutionary computation conference companion. 2018. p. 1821–8.
162. Pfund M, Fowler JW, Gupta JN. A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. J Chin Inst Ind Eng. 2004;21(3):230–41.
163. Pinto JM, Grossmann IE. Assignment and sequencing models for thescheduling of process systems. Ann Oper Res. 1998;81:433–66.
164. Pokharel B, Izquierdo ZG, Lott PA, Strbac E, Osiewalski K, Papathanasiou E, Kondratyev A, Venturelli D, Rieffel E. Inter-generational comparison of quantum annealers in solving hard scheduling problems. Quantum Inf Process. 2023;22(10):364.
165. Pinedo ML. Scheduling. vol. 29. New York: Springer; 2012.
166. Perelshtein M, Pakhomchik A, Melnikov AA, Podobrii M, Termanova A, Kreidich I, Nuriev B, Iudin S, Mansell C, Vinokur V. NISQ-compatible approximate quantum algorithm for unconstrained and constrained discrete optimization. arXiv preprint. 2023. arXiv:2305.14197.
167. Preskill J. Quantum computing in the nisq era and beyond. Quantum. 2018;2:79.
168. Pooja SSK. Scientometric analysis of quantum-inspired metaheuristic algorithms. Artif Intell Rev. 2024;57(2):22.
169. Plewa J, Sieńko J, Rycerz K. Variational algorithms for workflow scheduling problem in gate-based quantum devices. Comput Inform. 2021;40(4):897–929. https://doi.org/10.31577/cai_2021_4_897.
170. Prüfer S, Scherer A, Spörl A, Guggemos T, Pomplun N, Lenzen C. Quantum shift scheduling – a comparison to classical approaches. In: 12th International Workshop on Planning and Scheduling for Space (IWPSS 2021), 2021.
171. Prakash S, Vidyarthi DP. A novel scheduling model for computational grid using quantum genetic algorithm. J Supercomput. 2013;65:742–70.
172. Potts CN, Van Wassenhove LN. A branch and bound algorithm for the total weighted tardiness problem. Oper Res. 1985;33(2):363–77.
173. Pakhomchik A, Yudin S, Perelshtein M, Alekseyenko A, Yarkoni S. Solving workflow scheduling problems with QUBO modeling. 2022. arXiv preprint. arXiv:2205.04844.
174. Reza Hejazi S, Saghafian S. Flowshop-scheduling problems with makespan criterion: a review. Int J Prod Res. 2005;43(14):2895–929.
175. Rao PU, Sodhi B. Scheduling with multiple dispatch rules: a quantum computing approach. In: International conference on computational science. Berlin: Springer; 2022. p. 233–46.

176. Ripon KSN, Singh A. Quantum representation based job shop scheduling. In: 2023 IEEE symposium series on computational intelligence (SSCI). New York: IEEE Press; 2023. p. 1227–33.
177. Rylander B, Soule T, Foster J, Alves-Foss J. Quantum evolutionary programming. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2001). Citeseer; 2001. p. 1005–11.
178. Rieffel E, Venturelli D, Do M, Hen I, Frank J. Parametrized families of hard planning problems from phase transitions. In: Proceedings of the AAAI conference on artificial intelligence. vol. 28. 2014.
179. Rieffel EG, Venturelli D, O'Gorman B, Do MB, Prystay EM, Smelyanskiy VN. A case study in programming a quantum annealer for hard operational planning problems. Quantum Inf Process. 2015;14:1–36.
180. Svensson M, Andersson M, Grönkvist M, Vikstål P, Dubhashi D, Ferrini G, Johansson G. Hybrid quantum-classical heuristic to solve large-scale integer linear programs. Phys Rev Appl. 2023;20(3):034062.
181. Stollenwerk T, Basermann A. Experiences with scheduling problems on adiabatic quantum computers. In: Proceedings of the 1st international workshop on post-Moore era supercomputing (PMES). 2016. p. 45–6. Future Technologies Group Technical Report FTGTR-2016-11.
182. Schmid M, Braun S, Sollacher R, Hartmann MJ. Highly efficient encoding for job-shop scheduling problems and its application on quantum computers. 2024. arXiv preprint. arXiv:2401.16381.
183. Saad HM, Chakrabortty RK, Elsayed S. Quantum-inspired differential evolution for resource-constrained project-scheduling: preliminary study. In: 2021 IEEE congress on evolutionary computation (CEC). New York: IEEE Press; 2021. p. 1833–9.
184. Saad HM, Chakrabortty RK, Elsayed S, Ryan MJ. Quantum-inspired genetic algorithm for resource-constrained project-scheduling. IEEE Access. 2021;9:38488–502.
185. Slysz M, Grodzki Ł, Rydlichowski P, Siera D, Kurowski K, Waligóra G, Węglarz J. Solving combinatorial optimization and machine learning problems on hybrid near-term quantum photonic computers. Future Gener Comput Syst. 2026;174:107934.
186. Sun Y, Liu J, Ma Y, Tresp V. Differentiable quantum architecture search for job shop scheduling problem. In: ICASSP 2024-2024 IEEE international conference on acoustics, speech and signal processing (ICASSP). New York: IEEE Press; 2024. p. 236–40.
187. Singh MR, Mahapatra SS. A quantum behaved particle swarm optimization for flexible job shop scheduling. Comput Ind Eng. 2016;93:36–44.
188. Singh MR, Mahapatra S, Mishra R. Robust scheduling for flexible job shop problems with random machine breakdowns using a quantum behaved particle swarm optimisation. Int J Serv Oper Manag. 2015;20(1):1–20.
189. Shinjo T, Nakamura M, Itani N. Qubo model formulation for flow-shop scheduling problems with changeover based on timed colored Petri nets. In: 2022 tenth international symposium on computing and networking workshops (CANDARW). New York: IEEE Press; 2022. p. 212–7.
190. Sofge DA. Prospective algorithms for quantum evolutionary computation. 2008. arXiv preprint. arXiv:0804.1133.
191. Śmierzchalski T, Pawłowski J, Przybysz A, Pawela Ł, Puchała Z, Koniorczyk M, Gardas B, Deffner S, Domino K. Hybrid quantum-classical computation for automatic guided vehicles scheduling. Sci Rep. 2024;14(1):21809.
192. Singh KV, Raza Z. A quantum-inspired binary gravitational search algorithm–based job-scheduling model for mobile computational grid. Concurr Comput Pract Exp. 2017;29(12):4103.
193. Sawaya NP, Schmitz AT, Hadfield S. Encoding trade-offs and design toolkits in quantum algorithms for discrete optimization: coloring, routing, scheduling, and other problems. Quantum. 2023;7:1111.
194. Shimada D, Shibuya T, Shibasaki T. A decomposition method for makespan minimization in job-shop scheduling problem using Ising machine. In: 2021 IEEE 8th international conference on industrial engineering and applications (ICIEA). New York: IEEE Press; 2021. p. 307–14.
195. Su P-C, Tan S-Y, Liu Z, Yeh W-C. A mixed-heuristic quantum-inspired simplified swarm optimization algorithm for scheduling of real-time tasks in the multiprocessor system. Appl Soft Comput. 2022;131:109807.
196. Schworm P, Wu X, Glatt M, Aurich JC. Responsiveness to sudden disturbances in manufacturing through dynamic job shop scheduling using quantum annealing. Proc CIRP. 2023;120:511–6.
197. Schworm P, Wu X, Glatt M, Aurich JC. Solving flexible job shop scheduling problems in manufacturing with quantum annealing. SPE Prod Eng. 2023;17(1):105–15.
198. Schworm P, Wu X, Wagner M, Ehmsen S, Glatt M, Aurich JC. Energy supply scheduling in manufacturing systems using quantum annealing. Manuf Lett. 2023;38:47–51.
199. Sun J, Xu L. Cloud-based adaptive quantum genetic algorithm for solving flexible job shop scheduling problem. In: 2019 IEEE 7th international conference on computer science and network technology (ICCSNT). New York: IEEE Press; 2019. p. 1–5.
200. Taillard E. Benchmarks for basic scheduling problems. Eur J Oper Res. 1993;64(2):278–85.
201. Thakur AS, Biswas T, Kuila P. Binary quantum-inspired gravitational search algorithm-based multi-criteria scheduling for multi-processor computing systems. J Supercomput. 2021;77:796–817.
202. Tirado-Domínguez JA, Gutiérrez E, Plata O. Qtis: a qaoa-based quantum time interval scheduler. 2025. arXiv preprint. arXiv:2511.15590.
203. Tran T, Do M, Rieffel E, Frank J, Wang Z, O'Gorman B, Venturelli D, Beck J. A hybrid quantum-classical approach to solving scheduling problems. In: Proceedings of the international symposium on combinatorial search. vol. 7. 2016. p. 98–106.
204. Tindall J, Fishman M, Stoudenmire EM, Sels D. Efficient tensor network simulation of ibm's eagle kicked Ising experiment. PRX Quantum. 2024;5(1):010308.
205. Tan B, Lemonde M-A, Thanasilp S, Tangpanitanon J, Angelakis DG. Qubit-efficient encoding schemes for binary optimisation problems. Quantum. 2021;5:454.
206. Tayarani-Najaran M-H. Novel operators for quantum evolutionary algorithm in solving timetabling problem. Evol Intell. 2021;14(4):1869–93.
207. Tomasiewicz D, Pawlik M, Malawski M, Rycerz K. Foundations for workflow application scheduling on D-wave system. In: Proceedings, part VI 20. Amsterdam, The Netherlands, June 3–5, 2020 Computational Science–ICCS 2020: 20th International Conference. Berlin: Springer; 2020. p. 516–30.
208. Toma L, Zajac M, Störl U. Solving distributed flexible job shop scheduling problems in the wool textile industry with quantum annealing. 2024. arXiv preprint. arXiv:2403.06699.

209. Ullah MH, Eskandarpour R, Zheng H, Khodaei A. Quantum computing for smart grid applications. IET Gener Transm Distrib. 2022;16(21):4239–57.
210. Udrescu M, Prodan L, Vlăduţiu M. Implementing quantum genetic algorithms: a solution based on Grover's algorithm. In: Proceedings of the 3rd conference on computing frontiers. 2006. p. 71–82.
211. Vikstål P, Grönkvist M, Svensson M, Andersson M, Johansson G, Ferrini G. Applying the quantum approximate optimization algorithm to the tail-assignment problem. Phys Rev Appl. 2020;14(3):034009.
212. Venturelli D, Marchand D, Rojo G. Job shop scheduling solver based on quantum annealing. In: Proc. of ICAPS-16 workshop on constraint satisfaction techniques for planning and scheduling (COPLAS). 2016. p. 25–34.
213. Vyskočil T, Pakin S, Djidjev HN. Embedding inequality constraints for quantum annealing optimization. In: Quantum technology and optimization problems: first international workshop, QTOP 2019 Munich, Germany, March 18, 2019. Berlin: Springer; 2019. p. 11–22.
214. Watkins CJ, Dayan P. Q-Learn Mach Learn. 1992;8:279–92.
215. Wei X, Fan L, Guo Y, Gong Y, Han Z, Wang Y. Hybrid quantum-classical Benders' decomposition for federated learning scheduling in distributed networks. IEEE Trans Netw Sci Eng. 2024.
216. Windmann S. Quantum computer-aided job scheduling for storage and retrieval systems. Automatisierungstechnik. 2024;72(1):15–21.
217. Wu X, Li S. A quantum inspired algorithm for the job shop scheduling problem. In: 2011 IEEE 2nd international conference on computing, control and industrial engineering. vol. 2. New York: IEEE Press; 2011. p. 212–5.
218. Wu Q, Li H, Wang Z, Meng F, Luo B, Li W, Ngan KN. Blind image quality assessment based on rank-order regularized regression. IEEE Trans Multimed. 2017;19(11):2490–504.
219. Wu F, Wu Q, Tan Y. Workflow scheduling in cloud: a survey. J Supercomput. 2015;71:3373–418.
220. Wang L, Wu H, Tang F, Zheng D-Z. A hybrid quantum-inspired genetic algorithm for flow shop scheduling. In: Advances in intelligent computing: international conference on intelligent computing, ICIC 2005. Hefei, China, August 23-26, 2005. Proceedings, part II 1. Berlin: Springer; 2005. p. 636–44.
221. Wang L, Wu H, Zheng D-Z. A quantum-inspired genetic algorithm for scheduling problems. In: Advances in natural computation: first international conference, ICNC 2005. Changsha, China, AAugust 27-29, 2005. Proceedings, part III 1. Berlin: Springer; 2005. p. 417–23.
222. Xu H-Z, Chen J-H, Zhang X-C, Lu T-E, Gao T-Z, Wen K, Ma Y. High-speed train timetable optimization based on space–time network model and quantum simulator. Quantum Inf Process. 2023;22(11):418.
223. Xie J, Gao L, Peng K, Li X, Li H. Review on flexible job shop scheduling. IET Collab Intell Manuf. 2019;1(3):67–77.
224. Xiong H, Shi S, Ren D, Hu J. A survey of job shop scheduling problem: the types and models. Comput Oper Res. 2022;142:105731.
225. Xu Y, Wang D, Zhang M, Yang M, Liang C. Quantum particle swarm optimization with chaotic encoding schemes for flexible job-shop scheduling problem. Swarm Evol Comput. 2025;93:101836.
226. Xin F, Yun C, Hegen X. Research on optimal scheduling of workshop equipment maintenance based on quantum genetic algorithm. In: 2020 3rd world conference on mechanical engineering and intelligent manufacturing (WCMEIM). New York: IEEE Press; 2020. p. 318–23.
227. Xu Y, Zhang M, Wang D, Yang M, Liang C. Hybrid Gaussian quantum particle swarm optimization and adaptive genetic algorithm for flexible job-shop scheduling problem. Eng Appl Artif Intell. 2025;154:110882.
228. Xue H, Zhang P, Wei S, Yang L. An improved immune algorithm for multi-objective flexible job-shop scheduling. J Netw. 2014;9(10):2843.
229. Xu Y, Zhang M, Yang M, Wang D. Hybrid quantum particle swarm optimization and variable neighborhood search for flexible job-shop scheduling problem. J Manuf Syst. 2024;73:334–48.
230. Yonaga K, Miyama MJ, Ohzeki M. Solving inequality-constrained binary optimization problems on quantum annealer. 2020. arXiv preprint. arXiv:2012.06119.
231. Yu S, Nabil T. Applying the Hubbard-Stratonovich transformation to solve scheduling problems under inequality constraints with quantum annealing. Front Phys. 2021;9:730685.
232. Yao Y-J, Qian B, Hu R, Wang L, Xiang F-H. Carbon-efficient scheduling of blocking flow shop by hybrid quantum-inspired evolution algorithm. In: Intelligent computing theories and application: 14th international conference, ICIC 2018. Wuhan, China, August 15–18, 2018. Proceedings, part I. vol. 14. Berlin: Springer; 2018. p. 606–17.
233. Yarkoni S, Raponi E, Bäck T, Schmitt S. Quantum annealing for industry applications: Introduction and review. Rep Prog Phys. 2022.
234. Zielewski MR, Agung M, Egawa R, Takizawa H. Improving quantum annealing performance on embedded problems. Supercomput Front Innov. 2020;7(4):32–48.
235. Zhang J, Bianco GL, Beck JC. Solving job-shop scheduling problems with QUBO-based specialized hardware. In: Proceedings of the international conference on automated planning and scheduling. vol. 32. 2022. p. 404–12.
236. Zhang S, Du H, Borucki S, Jin S, Hou T, Li Z. Dual resource constrained flexible job shop scheduling based on improved quantum genetic algorithm. Machines. 2021;9(6):108.
237. Zhang Q, Hu S. An improved hybrid quantum particle swarm optimization algorithm for FJSP. In: Proceedings of the 2019 11th international conference on machine learning and computing. 2019. p. 246–52.
238. Zhang S-X, Hsieh C-Y, Zhang S, Yao H. Differentiable quantum architecture search. Quantum Sci Technol. 2022;7(4):045023. https://doi.org/10.1088/2058-9565/ac87cd.
239. Zhao X, Lin Q, Yu H. A co-scheduling problem of ship lift and ship lock at the three gorges dam. IEEE Access. 2020;8:132893–910.
240. Zhu H, Qi X, Chen F, He X, Chen L, Zhang Z. Quantum-inspired cuckoo co-search algorithm for no-wait flow shop scheduling. Appl Intell. 2019;49:791–803.
241. Zhao J-X, Qian B, Hu R, Zhang C-S, Li Z-H. An improved quantum-inspired evolution algorithm for no-wait flow shop scheduling problem to minimize makespan. In: Intelligent computing theories and application: 12th international conference, ICIC 2016. Lanzhou, China, August 2-5, 2016, Proceedings, part I. vol. 12. Berlin: Springer; 2016. p. 536–47.
242. Zhou Y, Tang Z, Nikmehr N, Babahajiani P, Feng F, Wei T-C, Zheng H, Zhang P. Quantum computing in power systems. IEnergy. 2022;1(2):170–87.

243. Zhong H-S, Wang H, Deng Y-H, Chen M-C, Peng L-C, Luo Y-H, Qin J, Wu D, Ding X, Hu Y, et al. Quantum computational advantage using photons. Science. 2020;370(6523):1460–3.
244. Zheng T, Yamashiro M. A novel hybrid quantum-inspired evolutionary algorithm for permutation flow-shop scheduling. J Stat Manag Syst. 2009;12(6):1165–82.
245. Zheng T, Yamashiro M. Minimizing total flowtime in flow shop scheduling by a quantum-inspired swarm evolutionary algorithm. In: 2010 international conference on electronics and information engineering. vol. 1. New York: IEEE Press; 2010. p. 1–351.
246. Zheng T, Yamashiro M. Solving flow shop scheduling problems by quantum differential evolutionary algorithm. Int J Adv Manuf Technol. 2010;49:643–62.
247. Zheng T, Yamashiro M. Solving no-wait flow shop scheduling problems by a hybrid quantum-inspired evolutionary algorithm. In: Advances in soft computing: 9th Mexican international conference on artificial intelligence, MICAI 2010. Pachuca, Mexico, November 8-13, 2010, Proceedings, part II. vol. 9. Berlin: Springer; 2010. p. 315–24.
248. Zhang X, Zhu G-Y. A literature review of reinforcement learning methods applied to job-shop scheduling problems. Comput Oper Res. 2025;175:106929.

Publisher's note