A HYBRID COLUMN GENERATION-BASED HEURISTIC FOR SOLVING THE PARALLEL MACHINE SCHEDULING PROBLEM WITH SEQUENCE-DEPENDENT SETUP TIMES

Luis Fernando Pérez Armas, Samuel Deleplanque, Riad Aggoune, Stefan Creemers

REPRINT | 3335

UCLouvain

CORE | LIDAM
Louvain Institute of Data Analysis and Modeling in economics and statistics

PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Research

Article submitted to journal

Subject Areas:
Quantum Optimization, Quantum Decision Making

Keywords:
Quantum Annealing, Hybrid Algorithms, Column Generation, Scheduling

Author for correspondence:
Luis Fernando Pérez Armas
e-mail: l.perezarmas@ieseg.fr

THE ROYAL SOCIETY PUBLISHING

A Hybrid Column Generation-Based Heuristic for Solving the Parallel Machine Scheduling Problem with Sequence-Dependent Setup Times

Luis Fernando Pérez Armas[1], Samuel Deleplanque[2], Riad Aggoune [3] and Stefan Creemers[4]

[1]Operations Management, IESEG School of Management, Univ. Lille, CNRS, UMR 9221 - LEM - Lille Economie Management, 3 rue de la digue, Lille, F-59800, Nord, France.
[2]Univ. Lille, CNRS, Centrale Lille, Junia, Univ. Polytechnique Hauts-de-France, UMR 8520 - IEMN, 41 Bd Vauban, Lille, 59000, France.
[3]Luxembourg Institute of Science and Technology, 5, avenue des Hauts-Fourneaux, L-4362, Esch-sur-Alzette, Luxembourg.
[4]Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), Voie du Roman Pays 34, B-1348 Louvain-la-Neuve, Belgium.

This study explores the application of a hybrid quantum-classical algorithm for solving the parallel machine scheduling problem with sequence-dependent setup times, a pivotal scheduling problem that has applications in multiple industries. Using a column generation-based approach, we propose a heuristic that combines a classical linear relaxation for the master problem with quantum annealing for solving the pricing sub-problem. Whereas the pricing sub-problem generates columns (i.e., a sequence of jobs that are assigned to a machine), the master problem selects which columns to use in order to minimize the makespan of the schedule. To generate columns, the pricing sub-problem solves a traveling salesman problem that is formulated as a quadratic unconstrained binary optimization problem. The big advantage thereof is that subtours can be eliminated by use of quadratic terms in the objective function. In addition, our approach also leverages the quantum annealer's capability to generate many high-quality solutions (i.e., columns) in a very short time. To assess the performance of our hybrid column generation-based heuristic, we perform a computational experiment. The results of this experiment demonstrate the synergy of hybrid methods for tackling complex decision making problems, achieving competitive high-quality solutions and computational advantages when compared to classical solution methods.

1. Introduction

In this paper, we consider the Parallel Machine Scheduling Problem (PMSP) with sequence-dependent setup times. Among scheduling problems, the PMSP is of particular importance due to its widespread applications in manufacturing, logistics, and service industries [3,28]. Most variants of the PMSP are NP-hard [39] and are typically addressed using heuristic or metaheuristic approaches [59].

Our primary objective is to develop and evaluate a hybrid quantum-classical framework for solving the PMSP more efficiently. Specifically, we propose a column generation-based heuristic that integrates Quantum Annealing (QA) for solving a key subproblem. While quantum approaches are gaining momentum in the field of combinatorial optimization, their integration within decomposition-based methods for complex scheduling problems such as the PMSP remains relatively underexplored. This study contributes to the growing research area of hybrid-classical quantum algorithms by proposing a novel hybrid design that takes advantage of both classical and quantum technologies.

QA has been explored in numerous studies as a promising method for solving optimization and scheduling problems on NISQ (Noisy Intermediate-Scale Quantum) machines. For instance, see [14,16,34,46] for recent developments demonstrating the potential of QA in addressing hard combinatorial problems. These works highlight the opportunities to further explore quantum-enhanced methods for industrial optimization problems and motivate the investigation of hybrid approaches that combine classical efficiency with quantum capabilities.

Rather than aiming to demonstrate the superiority of quantum machines over classical ones, our approach seeks to harness the strengths of both. Classical methods for solving linear programs have advanced significantly, enabling the efficient solution of LP problems with millions of variables at relatively low computational cost [15]. In this work, we combine the efficiency of classical LP solvers with the potential advantages of currently available quantum hardware to tackle the PMSP.

The proposed hybrid column generation methodology is not only theoretically motivated but also practically relevant to a number of real-world applications. For instance, with minimal adaptations, our approach could be applied to solve crew scheduling problems in transportation systems, such as those studied in [23,33], where decomposition-based strategies are central to handling large-scale assignment and sequencing decisions. Furthermore, the methodology is closely aligned with well-established column generation approaches used in airline crew scheduling [56]. Perhaps most importantly, this framework lends itself naturally to healthcare applications, particularly the nurse and operating room rostering problem [24], where schedules must account for staff availability, sequence-dependent setup or transition constraints, and institutional policies. As highlighted by recent reports on nurse shortages and healthcare system strain [25], improving the efficiency of these scheduling systems has substantial societal impact. These examples demonstrate the broad applicability and potential impact of our hybrid optimization approach in complex scheduling environments.

To address the PMSP and its potential real-world analogues, we adopt a decomposition strategy based on column generation, which divides the PMSP into a master problem and a pricing subproblem. The master problem is solved using linear relaxation techniques on a classical machine, while the pricing subproblem, modeled as a Traveling Salesman Problem (TSP) variant, handles the sequencing of jobs on each machine. Due to the inherent difficulty of the TSP, we further decompose the pricing problem into two tasks: selection and sequencing. This decomposition enables us to formulate the subproblem as a Quadratic Unconstrained Binary Optimization (QUBO) model, which can be efficiently solved using QA. The QUBO formulation offers structural advantages such as implicitly avoiding subtours without introducing additional constraints or variables. Additionally, quantum methods allow for the generation of a large set of high-quality solutions (and thus columns) in a single iteration, making them especially appealing for column generation procedures.

To assess the practical viability of the proposed hybrid framework, we conduct computational experiments using a D-Wave quantum annealer. Our results demonstrate the strengths of QA in generating effective columns for the PMSP, particularly in dense and complex instances. To the best of our knowledge, this is the first study to apply a quantum-classical hybrid column generation method based on QA to a scheduling problem as complex as the PMSP.

The main contributions of this paper are:

- We propose a decomposition of the PMSP that enables a transformation of its pricing subproblem into a QUBO formulation.
- We develop a novel hybrid heuristic that integrates QA within a classical column generation framework.
- We provide a detailed analysis of the impact of quantum hardware constraints, particularly qubit connectivity, on the performance of the hybrid approach.

The paper is organized as follows. Section 2 reviews the relevant literature. Section 3 presents the compact MILP formulation for the PMSP. Section 4 introduces the decomposition strategy, including the restricted master problem (RMP) and the pricing subproblem (PSP). Section 5 describes the column generation procedure, the proposed heuristic, and the QUBO formulation. Section 6 presents the computational results and experimental methodology. Finally, Section 7 discusses limitations, open questions, and future research directions.

2. Related Literature

To position our work, we first discuss the literature on the PMSP. Next, we provide a brief overview of the literature on QA and its implementation on a D-Wave machine. Lastly, we also discuss existing works that adopt QA to tackle the PMSP.

(a) The Parallel Machine Scheduling Problem with Sequence-Dependent Setup Times

In the PMSP, a set of $\mathcal{N}$ jobs has to be processed on a set of $\mathcal{M}$ machines. Two cases are generally considered in the literature: machines are either identical, or they are different. In the first case, the processing time p_j of a job j does not depend on the machine to which job j is allocated. In the second case, the processing time p_{mj} depends on the machine m that is selected for processing job j. In what follows, we assume that machines are identical.

The PMSP with sequence-dependent setup times is a classical problem in operations research and combinatorial optimization. In the PMSP, jobs must be assigned to one of a set of alternative machines and jobs that are assigned to the same machine have setup times that are defined as the time that must elapse between the end of one job and the start of the next job. The setup time is sequence dependent in the sense that the elapsed time between jobs will differ depending on the order in which jobs are scheduled. Those sequence-dependent setup times may correspond to maintenance or cleaning operations to be performed on the machines between the execution of two different jobs. Our goal is to minimize the makespan $C_{\max}$; that is, the maximum completion time of a schedule. The problem can be denoted by $P|s_{ij}|C_{\max}$, where P represents identical parallel machines, s_{ij} is the setup time in case job i precedes job j, and the objective is to produce the schedule that minimizes the makespan. The makespan minimization problem is NP-hard since the PMSP with two machines can be reduced to the partitioning problem [39]. In the literature, $P||C_{\max}$ is solved using exact methods such as branch and bound algorithms [42] or cutting plane techniques [44], as well as approximation schemes [30] and metaheuristics solution methods such as simulated annealing [38].

Taking into account setup times makes the PMSP even harder to solve. Most of the research in this field assumes identical parallel machines and focuses on heuristic solution methods [58].

In [31], a MILP model is proposed for $P|s_{ij}|C_{\max}$, as well as a heuristics approach based on the solution of a vehicle-routing problem. The authors in [29] propose lower bounds and a divide-and-merge heuristics that outperforms the tabu-search approach presented in [28]. [43] compares the performance of a heuristics that combines a genetic algorithm and a local-search procedure with a tabu search that is slightly adapted from [28]. Heuristic procedures are also prevalent in the literature that assumes the use of unrelated (i.e., non-identical) parallel machines [59].

Few studies have tackled the problem with decomposition methods. For example [18] was one of the first to use column generation for a problem with setup times similar to the one studied in this paper. [54] used a benders decomposition method for solving this problem.

(b) Quantum Annealing and its Implementation in D-Wave Machines

QA [16,49] and Simulated Annealing (SA) [37] share many similarities. SA is a classical optimization algorithm that constructs a discrete-time Markov chain to explore the solution space. The goal is to define the transition probabilities of this chain such that its limiting distribution converges to the optimal solution of the optimization problem. QA, on the other hand, is a quantum algorithm that operates through a continuous-time stochastic process over the solution space (or its relaxed version of it). Following the work of [36] and [2], QA can be viewed as a stochastic approach to minimizing a quadratic form $\boldsymbol{x}^\top Q\boldsymbol{x}$, also known as a QUBO problem, where the decision variables take binary values, $\boldsymbol{x} \in \{0,1\}^n$, with n representing the number of decision variables.

The objective of QA is to generate a probability distribution in $\{0,1\}^n$. This distribution is represented by the continuous-time stochastic process $\{|\Psi(t)|^2 : t \geq 0\}$, whose evolution is dictated by the Schrödinger equation:

$$i\hbar\frac{\partial\Psi(t)}{\partial t} = \mathcal{H}(t,\boldsymbol{x})\Psi(t),$$

where i is the imaginary unit, $\hbar$ is the reduced Planck constant, and $\mathcal{H}(t,\boldsymbol{x}) : \mathbb{R} \times \{0,1\}^{2^n} \to \mathbb{R}$ denotes the Hamiltonian.

The time-dependent Hamiltonian in QA evolves according to the principles of adiabatic evolution [10]. According to the adiabatic theorem, if a quantum system experiences a slow and continuous change in its Hamiltonian, it will remain in its instantaneous eigenstate during the transformation [2,36]. QA is physically implemented using analog devices that manipulate the states of qubits through a time-varying Hamiltonian defined as:

$$\mathcal{H}(t,x) = A(t)\mathcal{H}_0(x) + B(t)\mathcal{H}_1(x),$$

where $A(t)$ is an increasing function of time, and $B(t)$ decreases monotonically over time. The combination of these functions forms the annealing schedule, as shown in Figure 1.

The QA algorithm gradually transitions from the initial ground state of the starting Hamiltonian $\mathcal{H}_0$, known as the "mixing" or "tunneling" Hamiltonian, to the problem Hamiltonian $\mathcal{H}_1$. The problem Hamiltonian $\mathcal{H}_1$ corresponds to the energy function of the optimization problem, ensuring that the ground state of $\mathcal{H}_1$ represents the minimum-cost solution for the optimization problem. Typically, $\mathcal{H}_1$ is based on the Ising model.

The Ising model can be visualized by a graph $G = (V, E)$. In this context, the goal is to assign each variable s_i a value of either -1 or +1 (or binary values for QUBO) to minimize the following energy function $\mathcal{H}_1$:

$$\mathcal{H}_1 = \sum_{i\in V} h_i s_i + \sum_{(i,j)\in E} J_{ij} s_i s_j,$$

where h_i denotes the weight associated with variable s_i, and J_{ij} represents the coupling constant between variables s_i and s_j. The Ising and QUBO models are isomorph, with a transformation between them given by:

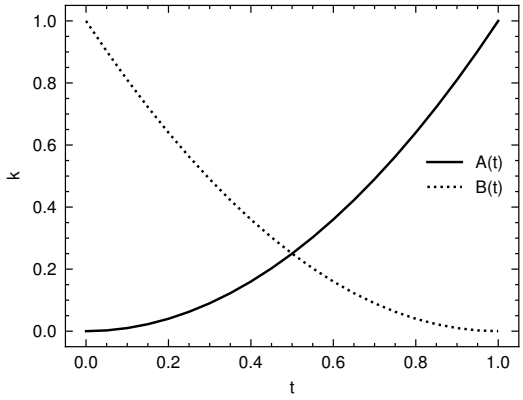


Figure 1: Typical annealing schedule. The y-axis represents the anneal fraction k, indicating the system's progress towards $\mathcal{H}_0$, while the x-axis shows the elapsed time.

$$x_i = \frac{1 + s_i}{2}.$$

The tunneling Hamiltonian $\mathcal{H}_0$ commonly used in QA is defined as:

$$\mathcal{H}_0 = \sum_{i \in V} \sigma_x s_i,$$

where σ_x, the Pauli-X operator applied to qubit i, is represented by:

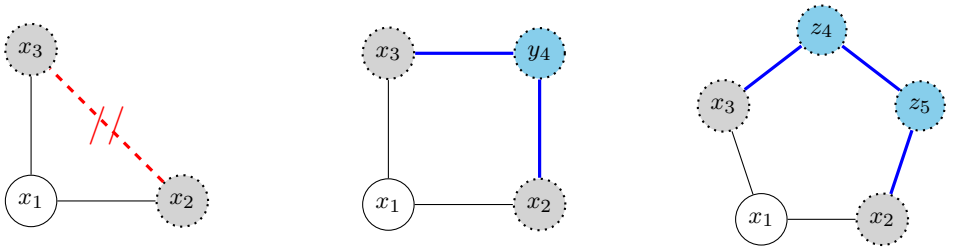
$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The choice of $\mathcal{H}_0$ is motivated by its well-known ground state and can be efficiently computed in polynomial time [26,27].

The advantage of QA for optimization lies in its ability to exploit quantum superposition. Unlike classical simulation approaches, QA allows quantum systems to evolve naturally over time according to the Schrödinger equation, eliminating the need for multiple simulations to approximate the limiting distribution, as required in SA. QA is a versatile optimization framework, capable of solving numerous hard problems since both the Ising and QUBO models are NP-hard [35].

The integration of quadratic models onto the Quantum Processing Unit (QPU) is referred to as embedding. If the user provides a QUBO model as input, it is transformed into an Ising model to map it onto the qubit graph of the QPU. This process varies across quantum machines. In D-Wave systems, the qubit graph is static and not fully connected. Therefore, the connectivity of the qubit graph plays a critical role in the success of the embedding process. To achieve higher connectivity and better align with the structure of the input graph, additional qubits, often referred to as *logical qubits*, are introduced and chained together using coupling constraints to enforce that they take the same value. Figure 2 illustrates this embedding process through the addition of one and two extra logical qubits, respectively.

To establish the required links between the original qubits and the additional logical qubits, the adjusted QUBO/Ising formulation incorporates penalty terms that enforces value consistency across the chain. For example, in the first embedding shown in Figure 2, the logical qubit y_4 is chained to the original qubit x_3 using a penalty term such as $\lambda_c(y_4 - x_3)^2$ or $\lambda_c(y_4 + x_3 - 2y_4x_3)$. The parameter λ_c, known as the *chain strength*, is critical for preserving the coherence of logical qubits throughout the annealing process.



Original complete graph used in the QUBO/Ising

Embedding with a chain length of 1

Embedding with a chain length of 2

Figure 2: Two examples of embedding a 3-vertex clique onto a qubit graph topology that lacks sufficient connectivity. The left panel depicts the original QUBO/Ising graph, with a dashed edge indicating that qubits are not physically connected in the QPU. The middle panel shows an embedding using one additional qubit (y_4). The right panel represents another embedding that uses two additional qubits ($z4$ and $z5$). The blue lines correspond to logical edges added to the graph and they are formally known as a "chain".

As a result of the embedding, the number of qubits required may exceed the number of decision variables. Consequently, the denser the input graph, the greater the number of qubits that may be needed, as illustrated in Figure 4 of Section 6. In addition, the coupling between qubits in a chain may lead to an issue known as a "chain break". This occurs when a chain of qubits fails to maintain the same value among all qubits in the chain.

(c) Quantum Annealing for Scheduling Problems and the PMSP

QA has been applied to a wide variety of decision problems across multiple domains. Refer, for instance, to [50] for an early application of QA in a manufacturing context, as well as [22] for a more recent application to the vehicle routing problem, an important problem in logistics and transportation. In the domain of scheduling, QA-based methods have been used to tackle well-known classical problems such as job-shop scheduling [1] and the resource-constrained project scheduling problem (RCPSP) [46], both of which have wide, ranging applications in manufacturing systems and project management.

More recently, hybrid quantum-classical algorithms have been explored for increasingly complex logistics and transportation problems that involve integrated scheduling components. For example, [53] proposes a hybrid QA approach to generate optimized pickup schedules for automated guided vehicles in warehouse operations, while [32] addresses the joint routing and scheduling of urban delivery fleets using quantum-inspired methods.

With regards to the PMSP, many papers make a direct use of the hybrid quantum-classical solver proposed by D-Wave. In [45], the authors addressed the unrelated PMSP with priorities for jobs and changeover times. They proposed QUBO formulations and tested their models on the D-Wave platform. Recently, [5] proposed a binary quadratic program for a PMSP with setup times, where the objective is to reduce the total setup time, to uniformly distribute the busy times of all machines, and to maximize the total value of the schedule. They compared the performance of the D-Wave Leap hybrid solver with that of a classical solver for solving a mixed-integer convex program.

QA is increasingly used in more complex solution procedures, such as the branch-and-bound method proposed in [11] for single machine scheduling problems, and the column-generation approach developed in what follows.

3. Compact Formulation of the PMSP

In this paper, we consider a PMSP where a set of jobs, $\mathcal{N}$, has to be scheduled on a set of identical machines, $\mathcal{M}$, with the objective of minimizing the makespan $C_{\max}$. Each job has a processing time p_j that corresponds to the time required to process job j. There are sequence-dependent setup times, s_{ij}: the time that must elapse between the completion of job i and the start of job j if job i precedes job j. The goal of the PMSP is to minimize the makespan by assigning jobs to machines and to sequence the execution of jobs on each machine.

To formally define the PMSP, we present a simplified version of the compact model developed by Avalos et al. [4] and Mokotoff et al. [44].

Sets:

$\mathcal{N}$	Set of all jobs;
$\mathcal{N}_0$	Set of all jobs with an additional dummy job (indexed by 0);
$\mathcal{M}$	Set of machines;

Exogenous parameters:

s_{ij}	Setup time if job i precedes job j;
p_j	Processing time of job j;
M	A large positive number.

Endogenous model decision variables:

x_{ijm}	Binary variable that equals 1 if job i precedes job j on machine m;
C_j	Completion time of job j;
O_m	Latest completion time of jobs on machine m;
$C_{\max}$	Maximum completion time (makespan).

If we assume that all inputs are non negative. Then a MILP formulation of the PSMP is given by:

$$
\begin{cases}
\min \quad C_{\max} & & (3.1a)\\
\text{subj. to:} \quad \displaystyle\sum_{i\in\mathcal{N}_0\setminus\{j\}}\sum_{m\in\mathcal{M}} x_{ijm} = 1 & \forall j\in\mathcal{N} & (3.1b)\\
\displaystyle\sum_{j\in\mathcal{N}\setminus\{i\}}\sum_{m\in\mathcal{M}} x_{ijm} = 1 & \forall i\in\mathcal{N}_0 & (3.1c)\\
\displaystyle\sum_{i\in\mathcal{N}_0\setminus\{h\}} x_{ihm} = \sum_{j\in\mathcal{N}\setminus\{h\}} x_{hjm} & \forall h\in\mathcal{N}, \forall m\in\mathcal{M} & (3.1d)\\
C_j - C_i + M(1-x_{ijm}) \geq s_{ij} + p_j & \forall i\in\mathcal{N}_0, j\in\mathcal{N}\setminus\{i\}, m\in\mathcal{M} & (3.1e)\\
\displaystyle\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}} x_{0jm} \leq 1 & & (3.1f)\\
C_0 = 0 & & (3.1g)\\
\displaystyle\sum_{i\in\mathcal{N}_0}\sum_{j\in\mathcal{N}\setminus\{i\}} (s_{ij}+p_j)x_{ijm} = O_m & \forall m\in\mathcal{M} & (3.1h)\\
O_m \leq C_{\max} & \forall m\in\mathcal{M} & (3.1i)
\end{cases}
$$

Objective function (3.1*a*) minimizes the makespan. Constraints (3.1*b*) and (3.1*c*) ensure that each job has exactly one predecessor and one successor on a single machine. Constraint (3.1*d*) specifies that a job can only have a successor on a machine if it also has a predecessor on the

same machine. Constraint (3.1*e*) defines the completion times of jobs such that, if job i precedes job j on machine m, the earliest time that job j can finish must exceed the completion time of job i plus the setup time between job i and j and the processing time of job j. Specifically, if job j is scheduled directly after job i on machine m, $(1 - x_{ijm}) = 0$, and the constraint simplifies to $C_j - C_i \geq s_{ij} + p_j$. Conversely, if job j is not scheduled directly after job i on machine m, $(1 - x_{ijm}) = 1$, and M renders the constraint non-binding. Constraint (3.1*f*) ensures that only one job is scheduled first on each machine. Constraint (3.1*g*) sets the completion time of job 0, a dummy job used to define the start of the schedule. Constraint (3.1*h*) calculates the makespan for each individual machine. Finally, constraint (3.1*i*) links the makespan of the individual machines to the overall schedule makespan.

The PMSP with sequence-dependent setup times is strongly NP-hard, as the assignment and sequencing of jobs on each machine can be modeled as a TSP (which is also NP-hard [6]). Due to the combinatorial complexity of the PMSP, even solving relatively small instances (fewer than 40 jobs) may require substantial computational effort [4]. The complexity of the problem can be mitigated by dividing the problem using Dantzig-Wolfe decomposition [21] methods such as column generation. In this decomposition, the main idea is to generate feasible, high-quality sequence assignments to machines (referred to as columns) and, once a sufficiently large pool of high-quality columns has been generated, decide which columns to select such that the overall makespan is minimized while ensuring that all jobs are processed.

4. Problem Decomposition

In this section, we provide a detailed breakdown of the decomposition process for the compact formulation described in (3.1), as well as a hybrid quantum decomposition heuristic for the PMSP. This decomposition procedure is a critical step in solving the associated combinatorial problem using a column generation-based algorithm. The necessity of such an approach arises from the inherent complexity of the PMSP. Enumerating all possible configurations for assigning jobs from a set of jobs $\mathcal{N}$ to a set of machines $\mathcal{M}$ and determining their sequencing becomes computationally infeasible, even for small-scale instances. To manage this complexity, the problem is divided into two components: the Restricted Master Problem (RMP) and the Pricing Sub-Problem (PSP). The RMP considers a reduced set of variables sufficient to compute a feasible allocation of jobs to machines, while the PSP identifies new sequencing assignments that satisfy the technical constraints imposed by the RMP. These new variables (columns) are then added to the RMP model, potentially enhancing the solution.

The column generation method is based on the principles of duality theory [8,21], which establishes a complementary relationship between the primal and dual optimization problems. This relationship can be summarized as follows: (i) each variable in the primal problem corresponds to a constraint in the dual and each constraint in the primal corresponds to a variable in the dual, and (ii) the optimization objective in the dual problem (e.g., maximization or minimization) is the opposite of that in the primal. For any sub-optimal solution that satisfies the primal constraints, there exists a direction in which the objective function can be improved. These improvement directions are represented by the dual variables, which help tighten the bounds of the primal problem. For a detailed discussion on the primal-dual connection, refer to [8]. After solving the primal problem as a linear model, dual variables, or improvement directions, can be retrieved using built-in solver functions.

To construct an effective column generation approach, the PSP leverages dual information from the current RMP solution, generating a new pricing instance whenever the sub-problem is invoked. By utilizing duality theory, the PSP identifies new variables that improve the RMP's objective function. The iterative process continues until no additional columns can be generated, signaling convergence.

(a) The Restricted Master Problem

Given that the PMSP may involve a vast number of potential job-sequence machine assignments (represented by the set $\Omega^\star$), the extended formulation (3.1) cannot be used to solve even small-scale instances. To avoid the immense challenge of exploring the entire set of assignments $\Omega^\star$, we propose constructing only a subset of high-quality assignments, $\Omega \subseteq \Omega^\star$. The assignments in Ω serve as columns that can be selected by the RMP such that the overall makespan is minimized.

As a simple initialization strategy, Ω can be created by naively dividing the jobs among machines in a balanced manner and sequencing each assignment using a greedy nearest-neighbor heuristic. This subset Ω is then iteratively updated by incorporating new variables generated by the PSP. This reduced model, referred to as the RMP, can be interpreted as a set partitioning problem and is formulated as follows:

Sets:

$\mathcal{N}$	Set of all jobs;
Ω	The set columns.

Exogenous parameters:

K_{jk}	Binary indicator that equals 1 if job j belongs to column k;
O_k	Completion time of last job in column k.

Endogenous model decision variables:

y_k	Binary variable that equals 1 if column k is used;
$C_{\max}$	Maximum completion time (makespan).

Formulation (RMP):

$$
\begin{cases}
\min & C_{\max} & & (4.1a)\\
\text{subj. to:} & \displaystyle\sum_{k\in\Omega} K_{jk} y_k \geq 1 & \forall\, j \in \mathcal{N} & (4.1b)\\
& O_k y_k \leq C_{\max} & \forall\, k \in \Omega & (4.1c)\\
& \displaystyle\sum_{k\in\Omega} y_k = |\mathcal{M}| & & (4.1d)
\end{cases}
$$

The objective function of the RMP (4.1*a*) minimizes the makespan. Constraint (4.1*b*) ensures that each job j is assigned to at least one column. Constraint (4.1*c*) ensures that the makespan is greater than or equal to the completion time of each selected column. Finally, constraint (4.1*d*) ensures that the number of selected columns does not exceed the total number of available machines, represented by the cardinality of set $\mathcal{M}$.

(b) The Pricing Sub-Problem

Given the RMP (4.1) and by associating a dual variable to each constraint (4.1*b*), multiple PSPs can be formulated for different numbers of jobs. These PSPs can be conceived as modified versions of the TSP, where a subset of W "cities" (i.e., jobs) must be selected and sequenced. The sequencing uses variables z_{ij} such that the selection minimizes the tour distance (i.e., the makespan of the sequence) while maximizing the sum of the dual values associated with the W selected cities.

The objective function of the PSP is referred to as the reduced price, which represents the marginal cost or improvement in the RMP's objective function when including a specific column in the solution. Thus, the pricing sub-problem for the PMSP can be defined as follows:

Sets:

$\mathcal{N}$	Set of all jobs (i.e., cities);
$\mathcal{N}_0$	Set of all jobs with an additional dummy job (that represents a virtual depot).

Exogenous parameters:

π_j	Dual values of constraint (4.1*b*);
π_0	Sum of dual values associated with constraint (4.1*c*) and (4.1*d*);
s_{ij}	Setup time if job i precedes job j;
p_j	Processing time of job j;
W	Maximum number of jobs to be selected;
M	A large positive number (can be set equal to $W+1$).

Endogenous model decision variables:

v_j	Binary variable that equals 1 if job j is selected for sequencing;
z_{ij}	Binary variable that equals 1 if job i precedes job j in the route;
u_j	Miller-Tucker-Zemlin sub-tour elimination variable for job j.

Formulation (PSP):

$$
\begin{cases}
\min \displaystyle\sum_{i\in\mathcal{N}_0}\sum_{j\in\mathcal{N}\setminus\{i\}} (s_{ij}+p_j)\, z_{ij} - \sum_{j\in\mathcal{N}_0} \pi_j v_j - \pi_0 & & (4.2a)\\
\text{subj. to: } \displaystyle\sum_{j\in\mathcal{N}_0} v_j = W & & (4.2b)\\
\displaystyle\sum_{j\in\mathcal{N}} z_{0j} = 1 & & (4.2c)\\
\displaystyle\sum_{i\in\mathcal{N}_0} z_{i0} = 1 & & (4.2d)\\
\displaystyle\sum_{j\in\mathcal{N}\setminus\{i\}} z_{ij} = v_i & \forall\, i\in\mathcal{N}_0 & (4.2e)\\
\displaystyle\sum_{i\in\mathcal{N}_0\setminus\{j\}} z_{ij} = v_i & \forall\, j\in\mathcal{N} & (4.2f)\\
u_i - u_j + M z_{ij} \le V - v_j & \forall\, i\in\mathcal{N}_0, j\in\mathcal{N}\setminus\{i\} & (4.2g)\\
u_j \ge v_j & \forall\, j\in\mathcal{N}_0 & (4.2h)\\
u_j \le W v_j & \forall\, j\in\mathcal{N}_0 & (4.2i)\\
u_j \le W & \forall\, j\in\mathcal{N}_0 & (4.2j)\\
u_j \ge 0 & \forall\, j\in\mathcal{N}_0 & (4.2k)
\end{cases}
$$

The objective function (4.2*a*) of the PSP minimizes the reduced cost of a sequence based on the dual values obtained from the RMP. Constraint (4.2*b*) ensures that, for each PSP, at most W jobs can be selected. Constraints (4.2*c*) and (4.2*d*) ensure departure from and return to the dummy depot (i.e., job 0). Constraints (4.2*e*) and (4.2*f*) enforce that each selected job is preceded and succeeded by exactly one job. Finally, (4.2*g*) to (4.2*k*) implement the Miller-Tucker-Zemlin (MTZ) constraints to eliminate subtours.

5. Quantum Hybrid Column Generation Heuristic Approach

Solving the pricing sub-problems is often the bottleneck in column generation-based procedures, as it requires to repeatedly solve multiple instances of a hard combinatorial problem (in our case

we need to solve multiple TSPs). To address this challenge, and inspired by the work of [20] and [22], we propose a quantum pricing heuristic algorithm that can find (near-) optimal solutions faster than its classical counterpart.

The quantum pricing heuristic algorithm needs to transform the PSP (4.2) into a QUBO model. The transformation is achieved using the formulation for the TSP proposed by [40], which reformulates the problem into a quadratic form by introducing positional variables and quadratic terms. The resulting model is expressed as the quadratic integer programming formulation shown below:

Sets:

$\mathcal{N}$	Set of all jobs;
$\mathcal{N}_0$	Set of all jobs with an additional dummy job;
$\mathcal{P}$	Set of positions in the sequence.

Exogenous parameters:

π_j	Dual values of constraint (4.1*b*);
π_0	Sum of dual values associated with constraint (4.1*c*) and (4.1*d*);
s_{ij}	Setup time if job i precedes job j;
p_j	Processing time of job j;
W	Maximum number of jobs to be selected.

Endogenous model decision variables:

q_{jp}	Binary variable that equals 1 if job j is assigned to position p;
a_j	Binary ancillary slack variable.

Quadratic integer programming formulation PSP:

$$\begin{cases} \min \displaystyle\sum_{i\in\mathcal{N}_0}\sum_{j\in\mathcal{N}\setminus\{i\}}\sum_{p}^{|\mathcal{P}|-1}(s_{ij}+p_j)q_{ip}q_{j(p+1)} - \sum_{j\in\mathcal{N}_0}\sum_{p}^{|\mathcal{P}|}q_{jp}\pi_j - \pi_0 & (5.1a) \\ \text{subj. to:} \displaystyle\sum_{j\in\mathcal{N}_0} q_{jp} = 1 \qquad \forall\, p\in\mathcal{P} & (5.1b) \\ \displaystyle\sum_{p}^{|\mathcal{P}|} q_{jp} \le 1 \qquad \forall\, j\in\mathcal{N}_0 & (5.1c) \end{cases}$$

The quadratic objective (5.1*a*) in the above formulation shares many similarities with objective function (4.2*a*) of the standard PSP formulation, as both aim to minimize the reduced cost of a column. However, a significant advantage of using a quadratic formulation for the PSP is that the objective function inherently incorporates subtour elimination constraints (4.2*g*) to (4.2*k*). Since the variables q_{jp} directly represent positions in the sequence, this eliminates the need for selection variables v_j and the MTZ subtour elimination variables u_j, as subtours are naturally handled by the structure of the quadratic terms in the objective function.

While this formulation requires quadratic terms, which classical solvers must linearize—often leading to computational overhead—this limitation does not apply to quantum annealers. The Ising model, utilized by quantum annealers, naturally handles quadratic terms without requiring linearization, making this approach well-suited for quantum computing.

In (5.1), constraint (5.1*b*) ensures that only one job j can be assigned to each position p, while constraint (5.1*c*) ensures that if a job j is selected, it cannot occupy more than one position.

QUBO model PSP: The quadratic PSP formulation (5.1) is subsequently transformed into a QUBO by incorporating constraints (5.1*b*) and (5.1*c*) as penalty terms within objective function (5.1*a*). This transformation yields the following QUBO expression:

$$f(\hat{X}) = \sum_{i\in\mathcal{N}_0}\sum_{j\in\mathcal{N}\setminus\{i\}}\sum_{p}^{|\mathcal{P}|-1}(s_{ij}+p_j)\,q_{ip}\,q_{j(p+1)} - \sum_{j\in\mathcal{N}_0}\sum_{p}^{|\mathcal{P}|}q_{jp}\pi_j - \pi_0 \tag{5.2a}$$

$$+\lambda_1\sum_{p}^{|\mathcal{P}|}\left(2\sum_{i\in\mathcal{N}_0}\sum_{j\in\mathcal{N}\setminus\{i\}}q_{ip}q_{jp} - \sum_{j\in\mathcal{N}_0}q_{jp}\right) \tag{5.2b}$$

$$+\lambda_2\sum_{j\in\mathcal{N}_0}\left(1-a_j-2\sum_{p}^{|\mathcal{P}|}q_{jp}+a_j\sum_{p}^{|\mathcal{P}|}q_{jp}+\sum_{p}^{|\mathcal{P}|}\sum_{g>p}^{|\mathcal{P}|}q_{jp}q_{jg}\right) \tag{5.2c}$$

The expression $f(\hat{X})$ represents the quadratic unconstrained binary representation of PSP (5.1), where $\hat{X}$ corresponds to a vector containing the variables q_{jp} and the additional ancillary slack variables a_j. Term (5.2*b*) is the penalty associated with constraint (5.1*b*) and, similarly, term (5.2*c*) represents the penalty associated with constraint (5.1*c*).

As demonstrated by the results in Figures 4 and 5 from Section 6 (below), and consistent with findings from [14] and [46], the QUBO formulations obtained for scheduling problems tend to be highly dense and connected. This creates significant challenges for the embedding process. Such embeddings require multiple qubits arranged in chains to achieve the connectivity required by the problem. Even for relatively modest problem sizes (e.g., sequencing 9 jobs), the resulting chains exceed the critical limit of 7, as defined by the current quantum annealer architecture of D-Wave.

Moreover even when using classical methods, PSP formulation (5.1) would still involve combining selection and sequencing decisions into a single problem, making it computationally expensive. Consequently, there is a clear need to further reduce the complexity of the PSP. To address this issue, we propose to divide the pricing problem into two distinct sub-problems that are solved sequentially: a selection problem followed by a sequencing problem.

It is important to note that this division introduces a trade-off between optimality and computational efficiency, potentially leading to suboptimal solutions for (5.1) and, in turn, dual gaps that may prematurely terminate the column generation process and guide it to sub-optimal columns. Similar trade-offs have been reported in related methods for other problems, such as the vehicle routing problem [55]. However, for the specific case of this study, the need for this trade-off arises from the current limitations of NISQ machines. As the architecture and connectivity of quantum machines improve, the reliance on such trade-offs is expected to decrease.

In the remainder of this section, we will present the proposed two-step heuristic method for solving the PSP (5.1).

Selection Sub-Problem: The selection process for the PMSP involves assigning jobs in $\mathcal{N}$ to the different machines in $\mathcal{M}$. To initiate this process, we begin by assigning seed jobs to each machine, referred to as "core" jobs. The dual values π_j obtained from the RMP provide valuable insights into the most critical jobs to prioritize, as jobs with higher dual values are more likely to contribute to smaller reduced costs.

Core jobs are selected based on a balanced strategy that aims to maximize the dual value of the core jobs while minimizing the difference in durations among them. This approach ensures that the selected core jobs are not disproportionately short, helping to balance the workload across machines. Additionally, we incorporate information about the largest setup times between already-selected core jobs to avoid assigning highly interdependent jobs at this initial stage. The detailed core selection strategy is presented in Algorithm 1. Assuming the use of *quicksort*, the algorithm requires, in the worst case, a number of steps on the order of $O(N\cdot(N+K))$, where N and K denote the cardinalities of the sets $\mathcal{N}$ and $\mathcal{K}$, respectively.

After initializing each machine with a core job, the remaining jobs still have to be assigned. To do this, we adopt a straightforward strategy that assigns the remaining jobs to the machines where

Algorithm 1: Core Job Selection Algorithm

Input: Set of jobs $\mathcal{N}$ with dual values π_j, durations p_j, and setup times s_{ij}.
Output: Core jobs $\mathcal{K}$ assigned to each machine in $\mathcal{M}$.

Step 1: Rank Jobs by Dual Values
Sort all jobs $\mathcal{N}$ in descending order of their dual values π_j;

Step 2: Assign Core Jobs to Machines
begin
  Select job j with the highest π_j as the first core job and update $\mathcal{K}$;
  Set remaining jobs $\mathcal{R} = \mathcal{N} \setminus \mathcal{K}$;
  foreach *remaining machine* m **do**
    foreach *remaining job* j *in* $\mathcal{R}$ **do**
      Calculate Score$(j, m) = \pi_j + p_j - s_{jj\prime}$, where jobs $j\prime$ are the core jobs in $\mathcal{K}$ that have already been selected;
    end
  end
  Select job j that maximizes Score(j, m) over all remaining machines;
  Assign job j to machine m and update $\mathcal{K}$ and $\mathcal{R}$;
end

Step 3: Return core jobs $\mathcal{K}$

they contribute the most to minimizing the reduced cost. The assignment strategy is detailed further in Algorithm 2. Assuming the use of *quicksort* in Algorithm 1, and that the array of jobs sorted by dual values π_j produced in Algorithm 2 is maintained in memory, then Algorithm 2 requires, in the worst case, a number of steps on the order of $O(N \cdot K)$.

Algorithm 2: Job Assignment to Machines

Input: Set of jobs $\mathcal{N}$, core jobs $\mathcal{K}$, durations p_j, dual values π_j, and setup times s_{ij}.
Output: Set of jobs $\mathcal{Q}_m$ assigned to machine m.

Step 1: Initialize Sets of Assigned Jobs
Use Algorithm 1 to assign core job $j\prime \in \mathcal{K}$;
Set remaining jobs $\mathcal{R} = \mathcal{N} \setminus \mathcal{K}$.;
Sort all jobs $\mathcal{R}$ in descending order of their dual values π_j;

Step 2: Iterative Assignment of Remaining Jobs
while $\mathcal{R} \neq \emptyset$ **do**
  Initialize best cost $c^\star = \infty$;
  foreach *job* $j \in \mathcal{R}$ **do**
    foreach *machine* m **do**
      Let i be the last job processed on machine m;
      Compute cost: $c = s_{ij} + p_j - \pi_j$;
      if $c < c^\star$ **then**
        Update best cost $c^\star = c$, best job $j^\star = j$, and best machine $m^\star = m$;
  Assign $j^\star$ to $\mathcal{Q}_{m^\star}$;
  Remove $j^\star$ from $\mathcal{R}$;

Step 3: Return Sets of Assigned Jobs
Output set $\mathcal{Q}_m$ for each machine m;

The selection heuristic defined by Algorithms 1 and 2 produces a set of selected jobs $\mathcal{Q}_m$ aimed at generating columns with minimal reduced cost. It is important to note that the sequence of the jobs within the $\mathcal{Q}_m$ has not yet been optimized, a task that will be addressed in the next step–the sequencing problem.

Sequencing Sub-Problem: For each machine m, the jobs in $\mathcal{Q}_m$ still need to be scheduled in order to obtain the minimum-duration sequence. This requires us to solve a TSP for each machine m. To accomplish this, we leverage the advantages of the quadratic TSP formulation, which can be transformed into a QUBO that can be solved using a quantum annealer.

The QUBO expression for the sequencing sub-problem is similar to (5.2), but is restricted to the jobs in the set $\mathcal{Q}_m$ instead of $\mathcal{N}$. Additionally, the linear maximization terms corresponding to the dual values π_j are excluded, as this aspect of the objective function has already been addressed by the prior selection heuristic. Penalty term (5.2*b*) remains unchanged, while penalty term (5.2*c*) is modified to ensure that all jobs are assigned to exactly one position. Notably, this new penalty formulation eliminates the need for ancillary variables a_j.

The QUBO for the sequencing sub-problem is presented below:

$$f(\hat{X}) = \sum_{i\in\mathcal{Q}_m}\sum_{j\in\mathcal{Q}_m\setminus\{i\}}\sum_{p}^{|\mathcal{P}|-1}(s_{ij}+p_j)q_{ip}q_{j(p+1)} \tag{5.3a}$$

$$+\lambda_1\sum_{p}^{|\mathcal{P}|}\left(2\sum_{i\in\mathcal{Q}_m}\sum_{j>i,j\in\mathcal{Q}_m}q_{ip}q_{jp}-\sum_{j\in\mathcal{Q}_m}q_{jp}\right) \tag{5.3b}$$

$$+\lambda_2\sum_{j\in\mathcal{Q}_m}\left(2\sum_{p}^{|\mathcal{P}|}\sum_{g>p}^{|\mathcal{P}|}q_{jp}q_{jg}-\sum_{p}^{|\mathcal{P}|}q_{jp}\right) \tag{5.3c}$$

(a) Column Generation Process and General Resolution Framework

The combination of the selection and sequencing heuristics produces valid, high-quality columns that can be incorporated into the RMP, enabling the implementation of a column generation scheme. Utilizing QA for the PSP provides a unique advantage, arising from what initially appears to be a limitation of the current NISQ annealing technology.

Due to the inability to fully satisfy the conditions of the adiabatic theorem, each run on the quantum annealer generates multiple shots (evolutions) to compensate for the non-ideal adiabatic conditions. Each shot has a runtime in the order of microseconds (μs) because the annealer employs superconducting technology, making the process computationally inexpensive. This ability to generate multiple solutions in a single annealing run offers a significant advantage for a hybrid column generation approach. Specifically, it allows to obtain multiple feasible and high-quality solutions for the PSP, each with the potential to be added to the RMP/to improve the RMP in one and the same iteration.

The hybrid column generation-based approach proposed for solving the PMSP is summarized in Algorithm 3 of Appendix (a), and in Figure 3 below.

As depicted in the UML diagram, the process begins with the initialization of the RMP using an initial set of columns, Ω^*, which is obtained on a classical computer. To initialize Ω^*, we use a naive but feasible set of columns that ensure constraint 4.1*b* of the RMP is satisfied, that is, each job i must be covered by at least one column. One way to construct such a feasible initial set is to generate N columns, each of which excludes a single job i, while sequencing the remaining jobs in increasing index order. Additionally, we include N singleton columns where each column consists of only a single job. The dual solutions, π, are then extracted from the RMP output and provided to the *Core Job Selection* algorithm to create an initial assignment of one job to each machine, denoted by $\mathcal{K}$. This result serves as input to the *Job Assignment to Machines* algorithm, which generates assignments, $\mathcal{Q}_m$, that are used to formulate a QUBO. The QUBO is solved using QA, thereby

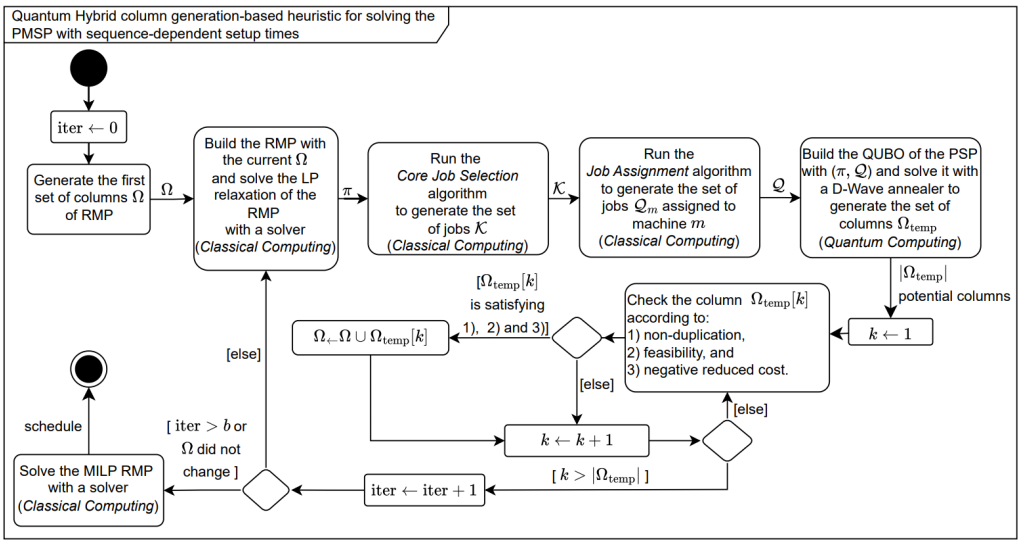


Figure 3: UML activity diagram: the general resolution framework operates in a closed-loop between the RMP and the PSP. The RMP provides dual solutions to the PSP, while the PSP supplies columns to the RMP. Additionally, the PSP is supported by algorithms 1 and 2 to select core jobs and to assign jobs to machines.

creating a temporary set of columns, Ω_{temp}. The temporary columns, Ω_{temp}, are then processed in three steps before being added to the general set of columns, Ω. First, duplicate columns are filtered out. Second, a feasibility test ensures that the generated columns satisfy all constraints. This step is essential because, even if a column has a negative reduced cost, it does not guarantee that constraints are not violated in the QUBO model (5.3) (e.g., violations of constraints (5.3*b*) or (5.3*c*)). Third, only columns with a negative reduced cost are selected for inclusion in Ω. The full process is iterated b times or stops if no new column is found. In the latter case, an alternative approach could, for instance, restart the process with additional iterations, as the probabilistic nature of the process allows for non-deterministic outcomes.

6. Computational Experiments

In this section, we first discuss the generation of the PMSP instances that are used in a computational experiment to benchmark our proposed algorithm. Next, we describe the machine configuration that was used to run the computational experiment. Finally, we analyze the numerical results, and compare the performance of the proposed method against that of classical approaches.

(a) Instances and Benchmark

Due to the current limitations in the number of qubits available in the annealer, as well as restricted access to quantum computational resources, the instances used in this study were generated rather than retrieved from the literature. The procedure used to generate the instances is similar to that of [4], with the exception that the instances in our study assume that machines are identical.

We generated a set of instances of varying sizes across multiple dimensions, including the number of jobs and the number of machines. Job durations were uniformly generated with values ranging from 1 to 20 ($p_j \in [1, 20]$), and setup times were generated with values ranging from

1 to 10 ($s_{ij} \in [1, 10]$). Table 1 summarizes the combinations of jobs and machines used in the computational experiments.

For each job-machine size combination, both a symmetric version (i.e., $s_{ij} = s_{ji}$) and an asymmetric version (i.e., $s_{ij} \neq s_{ji}$) of the problem was generated, resulting in a total of 14 distinct instances.

Table 1: Problem Instance sizes

Number of Jobs	Number of Machines
10	2
10	3
15	3
15	6
20	3
20	4
20	6

For each instance, we report the best makespan obtained, denoted as Optimal*, which corresponds to the best solution achieved by a classical solver running for a time limit of 600 seconds, solving extended formulation (3.1). Additionally, we present both the best solution and the runtime obtained by our quantum hybrid column generation heuristic, denoted as *hyb-QA-Col*, and compare it to the same heuristic implemented with classical methods, such as SA and a standard commercial solver (referred to as *hyb-SA-Col* and *Classic-Col*, respectively). The three methods were allowed to run for a maximum of 40 iterations, with the best result and corresponding runtime reported.

(b) Characteristics of Quantum and Classical Machines

Quantum Annealer: The quantum annealer used for our experiments is the D-Wave Advantage 6.4, which features 5,612 qubits and 40,088 couplers. These couplers are connected to an analog control system through a network of Josephson junctions. The machine follows the fixed Pegasus topology, offering a maximum qubit connectivity of 15. The embedding of the QUBO problem onto the hardware graph was carried out using D-Wave's default minor-embedding heuristic [13].

The penalty coefficients λ_1 and λ_2, associated with constraints (5.3*b*) and (5.3*c*), were selected following the approach proposed by [40] for the TSP. In particular, they were set to match the maximum expected value of the objective function in (5.3*a*), i.e., $\lambda = \sum_j p_j + \sum_i \sum_{j \neq i} s_{ij}$. This choice ensures that constraint violations are sufficiently penalized without dominating the energy landscape. The chain strength parameter was set to the same value as λ, in accordance with recommendations by [57], which suggest using a value on the same order of magnitude as the objective function to preserve logical qubit consistency while minimizing distortion of the problem structure.

All quantum annealing calls used an annealing time of 20 μs and 100 shots per call. This annealing time has been found to strike a good balance between solution quality and runtime in similar scheduling contexts [14,46,57]. While more refined annealing schedules (e.g., with pauses) could improve the likelihood of sampling ground states [57], we opted for a fixed annealing schedule to minimize quantum resource usage and maintain experimental consistency.

All quantum processing was conducted via the `dwave-ocean` software development kit using the `BinaryQuadraticModel` (BQM) class for QUBO formulation.

Classical Solver and Machine Configuration: The computational experiments were conducted on a cloud-based machine accessed via Google Colab. The system ran on a Linux 64-bit

platform using "Ubuntu 22.04.3 LTS" and was equipped with an Intel(R) Xeon(R) CPU @ 2.20GHz, featuring one physical core and two logical processors capable of utilizing up to two threads. The machine had access to 12 GB of working memory and 107 GB of disk storage. The classical optimization for all the computational experiments was performed using Gurobi Optimizer version 12.0.0, accessed via the web license service and invoked through Python library `gurobipy` version 12.0.0, running on `Python` 3.10.12. All the optimization problems were run using a `MIPGap` parameter of 0.0 and a `TimeLimit` equal to 120 seconds while running the iterations and 600 seconds while solving the non-restricted master problem.

Simulated Annealing: SA was executed using D-Wave's `dwave-neal` function in order to solve exactly the same QUBO formulation as the one that was used by QA. For each call to the SA solver, 100 shots were performed. The SA algorithm was run on the same machine with identical specifications as those used for the classical solver.

(c) Numerical Results

In this section, we first present the numerical results related to the challenge of embedding the PSP onto the QPU. Next, we present the numerical results obtained from running the three algorithms: *hyb-QA-Col*, *hyb-SA-Col*, and *Classic-Col*.

Embedding Analysis QUBO PSP: The results related to the embedding process of the QUBO formulation highlight the necessity of dividing the PSP QUBO problem (5.2) into a two-stage heuristic: selection (Algorithms 1 and 2) and sequencing (5.3). Table 2 reports the key statistics obtained when embedding the the QUBOs for formulation (5.3) for various problem sizes.

Table 2: PSP QUBO sizes and chain statistics

N	qubits	physical_qubits	mean_chain_L	max_chain_L
3	9	12	1.33	2
4	16	35	2.19	3
5	25	68	2.72	3
6	36	148	4.11	6
7	49	249	5.08	7
8	64	441	6.89	10
9	81	583	7.20	11
10	100	995	9.95	16
11	121	1152	9.52	13
12	144	2040	14.17	22
13	169	3352	19.83	34
14	196	3969	20.25	37

The results presented in Table 2 highlight the significant disparity between the number of qubits in the original QUBO formulation and the number of physical qubits required to embed the problem onto the quantum annealer. To further illustrate the challenge of embedding highly dense and connected problems, in Figure 4, we visualize the mapping of three example instances (sizes: 3, 9, and 14) onto the Pegasus architecture of the quantum annealer.

Furthermore, Figure 5 illustrates the distribution of chain lengths for the various problem sizes evaluated in this study. For sequencing problem sizes of 8 or more jobs, the average chain lengths exceed the warning threshold established by D-Wave. Consequently, the samples corresponding to the lowest energy states consistently contained broken chains.

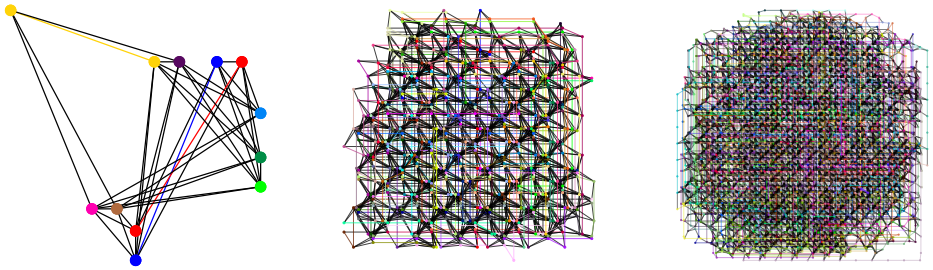


Figure 4: Illustration of the embedding of QUBO formulation (5.3) onto the Pegasus architecture of the quantum annealer for problem sizes 3, 9, and 14 (shown from left to right).

It is worth noting that methods exist to repair broken chains (e.g., using Monte Carlo sampling; see also [41]). However, in this study, infeasible solutions with broken chains were not included as part of the resolution process of the proposed algorithms.

Computational Results: We now present the computational results obtained for the benchmark instances evaluated in this study. Table 3 summarizes the performance of the three column generation methods tested, reporting the average percentage deviation from the best-known solution ($\Delta\, C_{\max}\%$) and the corresponding running times (in seconds). These metrics are aggregated across all instances to provide a comprehensive comparison of solution quality and computational efficiency.

Table 3: Summary of Results

	Classic-Col	*hyb-QA-Col*	*hyb-SA-Col*
$\Delta\, C_{\max}\%$	23.95 %	6.74 %	8.75 %
Running Time (s)	6.76 (s)	5.75 (s)	1.03 (s)

Table 4 provides a detailed report of the the makespan ($C_{\max}$) and runtime (in seconds) for the three algorithms, along with the best solution for each instance obtained by solving the extended problem (3.1) running for a maximum time-limit of 600 seconds.

Next, we compare the performance of *hyb-QA-Col*, *hyb-SA-Col*, and *Classic-Col*. The results are summarized in Table 4. Table 4 clearly demonstrates that, for instances with more than 10 jobs, all three heuristic methods provide a significant computational advantage when compared to solving extended formulation (3.1). This is particularly evident for *hyb-QA-Col* and *hyb-SA-Col*, which were able to obtain high-quality solutions that were close to Optimal* for the majority of the instances. Figure 6 illustrates an example solution of the schedule obtained by *hyb-QA-Col* for the asymmetric instance with 10 jobs and 2 machines.

Based on the runtime information for Optimal* in Table 4, we observe that symmetric instances tend to be more challenging to solve. This is expected, as integer programming methods, such as branch-and-bound, often struggle to differentiate between sequences that have the same objective value. To address this issue, classical methods typically incorporate symmetry-breaking cuts during the resolution process.

In contrast, for SA and particularly for QA, the symmetric structure of the instance; which in the quantum domain translates to the degeneracy of states sharing the same energy levels, appears to simplify the optimization procedure, enabling faster convergence to better solutions. Figure 7 illustrates the relative differences in solution quality among the three algorithms compared to Optimal*. On average, *hyb-QA-Col* achieves an average deviation of 6.74% across

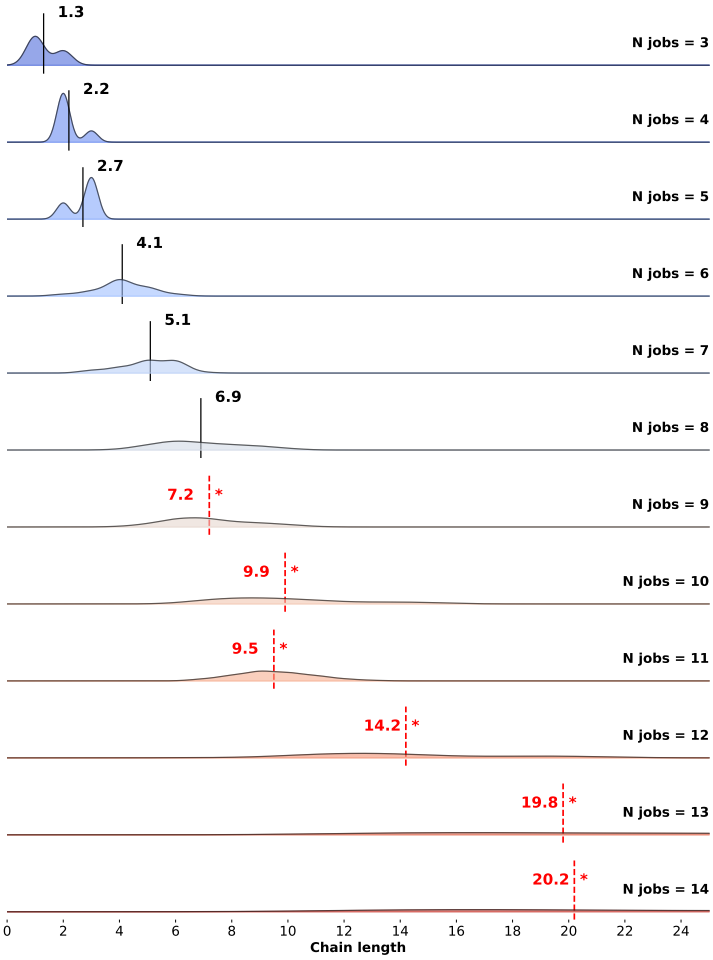


Figure 5: Distribution of chain lengths for each problem size of QUBO formulation (5.3). Vertical lines represent the mean chain length for each problem size. Lines are differentiated by color and style: red dashed lines indicate that the mean chain length exceeds the warning threshold of 7, while a $*$ next to the line indicates that the lowest-energy sample contained a broken chain.

all instances, while *hyb-SA-Col* and *Classic-Col* exhibit average deviations of 8.75% and 23.95%, respectively.

In general, there is evidence that both *hyb-SA-Col* and *hyb-QA-Col* benefit from the ability to add multiple columns in a single iteration, particularly for larger instance sizes, where *hyb-QA-Col* demonstrates notable gains. This observation aligns with the findings of [20]. However, it is important to note that adding multiple columns does not guarantee improved performance in all cases, as evidenced by specific instances such as (20-4 asymmetric) and (15-5 symmetric).

In terms of runtime, *hyb-SA-Col* shows a distinct advantage over both *hyb-QA-Col* and *Classic-Col*. This behavior seems to be driven by the size of the sequencing problems (TSPs) that these classical methods must solve, with smaller sequencing problems leading to faster runtimes. In contrast, *hyb-QA-Col*'s runtime is primarily influenced by the number of sequencing problems (equivalent to the number of machines and, therefore, the number of calls per iteration). This represents an inverse behavior compared to the classical methods, which benefit from solving a larger number of smaller problems, while QA performs better with fewer, larger problems. Nonetheless, *hyb-QA-Col* tends to outperform the other methods in terms of solution quality,

Table 4: Comparison of Methods

N	M	Symmetric	Optimal*	*Classic-Col*	*hyb-QA-Col*	*hyb-SA-Col*
10	2	T	(68 - 1.84 s)	(70 - 4.4 s)	(69 - 2.24 s)	(69 - 0.53 s)
10	2	F	(66 - 1.54 s)	(73 - 2.63 s)	(67 - 2.72 s)	(67 - 0.6 s)
10	3	T	(46 - 2.34 s)	(47 - 1.72 s)	(47 - 4.53 s)	(47 - 0.67 s)
10	3	F	(45 - 0.72 s)	(54 - 1.57 s)	(45 - 5.17 s)	(45 - 0.72 s)
15	3	T	(64 - 600.13 s)	(68 - 6.38 s)	(67 - 5.11 s)	(68 - 1.02 s)
15	3	F	(60 - 7.62 s)	(71 - 6.18 s)	(65 - 4.86 s)	(64 - 1.13 s)
15	5	T	(38 - 600.1 s)	(41 - 3.08 s)	(42 - 6.42 s)	(42 - 0.79 s)
15	5	F	(36 - 27.19 s)	(40 - 1.78 s)	(40 - 7.2 s)	(40 - 0.7 s)
20	3	T	(82 - 417.85 s)	(126 - 20.85 s)	(84 - 4.71 s)	(90 - 1.49 s)
20	3	F	(80 - 54.82 s)	(118 - 17.03 s)	(91 - 4.62 s)	(100 - 1.58 s)
20	4	T	(61 - 600.13 s)	(88 - 9.42 s)	(64 - 6.19 s)	(69 - 1.26 s)
20	4	F	(61 - 600.14 s)	(68 - 9.15 s)	(77 - 6.82 s)	(72 - 1.18 s)
20	6	T	(42 - 600.22 s)	(58 - 6.59 s)	(44 - 9.33 s)	(45 - 1.56 s)
20	6	F	(41 - 600.19 s)	(66 - 3.93 s)	(42 - 10.65 s)	(45 - 1.3 s)

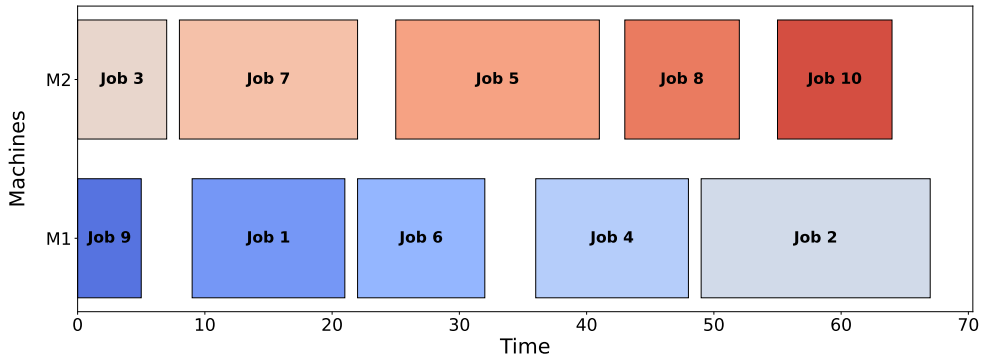


Figure 6: Example schedule obtained using *hyb-QA-Col* for an instance with 10 jobs and 2 machines, featuring asymmetric setup times. The schedule is presented in the form of a Gantt chart, where activities are organized in their order of execution across the machines. The x-axis represents the time dimension, while each unit of the y-axis corresponds to a different machine.

particularly for larger instance sizes. This is consistent with the findings reported in [7,19], which show that for discrete, non-convex problems where integrality constraints play a central role, quantum annealing tends to outperform simulated annealing in terms of solution quality. Consequently, we expect that as instance sizes grow and the sequencing subproblem becomes more complex, the quality of solutions obtained via simulated annealing will deteriorate more rapidly compared to those produced by quantum annealing.

7. Summary and Discussion

In this work, we have conducted a comprehensive exploration of the practical implementation of a hybrid quantum heuristic based on a Dantzig-Wolfe decomposition (column generation) to address one of the most significant production problems, the parallel machine scheduling problem with sequence-dependent setup times. This problem has extensive applications across various industries, including manufacturing, agriculture, healthcare, communications, and task allocation in multiprocessor computing systems.

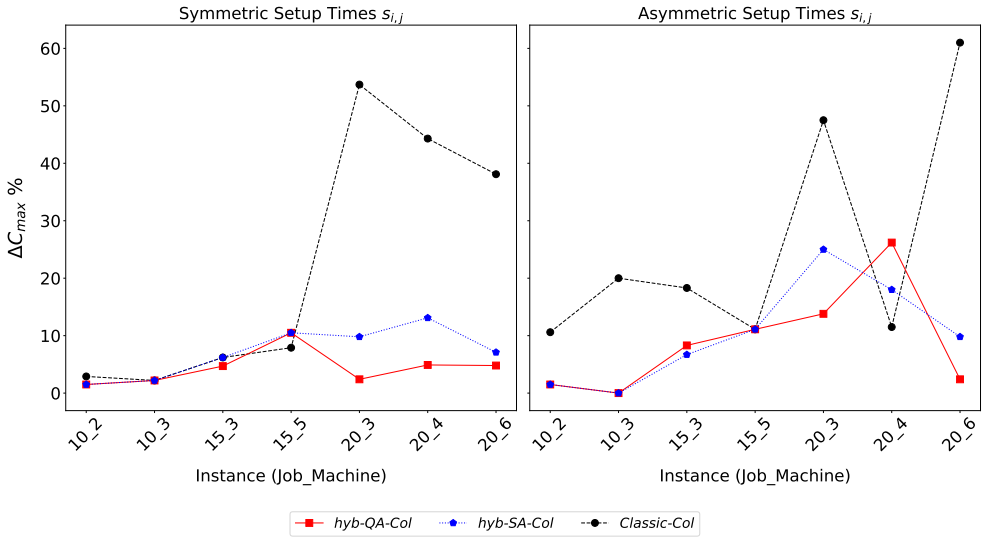


Figure 7: The Y-axis represents the relative difference in makespan ($C_{\max}$), calculated as the percentage difference between the makespan values of the different algorithms and the Optimal* value. The X-axis corresponds to the various instance sizes, arranged in ascending order of jobs followed by machines. The left plot shows the relative $\Delta C_{\max}$ values for the symmetric instances, while the right plot presents the results for the asymmetric instances.

Our study provides an in-depth examination of what it entails to solve this problem using currently available NISQ quantum annealers. We propose a novel hybrid heuristic approach that leverages dual values obtained from the solution of a relaxed restricted master problem to guide the optimization process. This approach demonstrates the potential advantages of hybrid classical-quantum algorithms in addressing some of the most challenging decision-making problems in the near term. To the best of our knowledge, this is the first study to tackle the parallel machine scheduling problem using a hybrid quantum approach grounded in duality theory and column generation techniques. Notably, our approach is flexible and can be adapted to optimize other objectives within the parallel machine problem framework, such as minimizing tardiness or total tardiness.

Limitations and Practical Challenges: While our research offers valuable insights, particularly regarding the short- and medium-term evolution of quantum optimization techniques applied to scheduling problems, it also has limitations. First, our proposed approach is not an exact method but a heuristic. Nonetheless, the results obtained are promising, demonstrating the potential of our approach in practice. Second, the algorithms that were used in our approach could be further refined by incorporating, for instance, weight coefficients α, β, and γ, allowing for adjustable relevance of job durations, setup times, and dual values. Exploring this refinement could potentially improve the performance of our approach.

In addition, we used problem instances of limited size, primarily due to limitations imposed by the quantum hardware. This restricted our study to sequencing problems involving no more than 8 jobs per machine. As quantum hardware advances, we anticipate these limitations will diminish, enabling researchers to tackle larger and more complex optimization problems. It is worth emphasizing that embedding remains a critical bottleneck; future research should investigate the impact of better/optimal embedding strategies such as the ones proposed by [9], as well as chain-repair and solution-improvement methodologies [41]. Emerging quantum technologies, such as neutral atom machines [20,52] and photonic wave-based annealers [47],

are also showing promise in addressing connectivity challenges, potentially paving the way for solving larger-scale and more complex problems.

Although quantum annealing remains one of the most promising NISQ optimization methods, other quantum algorithms, such as variational quantum algorithms, notably the Quantum Approximate Optimization Algorithm (QAOA) [26], merit further exploration. These methods could potentially be integrated into the hybrid approach proposed in this study as sub-routines of the pricing-sub-problem, to perhaps further enhance its effectiveness.

Future perspective: Despite the acknowledged limitations, this work serves as a step in advancing the development of quantum hybrid algorithms that combine the strengths of classical and quantum optimization. Beyond the results presented here, several promising directions for future research remain open. From an algorithmic perspective, the solutions generated by our hybrid column generation framework could be further refined using quantum amplitude amplification techniques [12] to enhance the quality of the selected columns and further reduce the makespan $C_{\max}$. Additionally, the current heuristic framework could be extended into an exact branch-and-price algorithm [17] by incorporating classical exact methods for solving the pricing subproblem in the final iterations of column generation. This would provide optimality guarantees while still benefiting from the exploratory capabilities of quantum annealing in the earlier stages.

From an application standpoint, our proposed methodology is broadly relevant across multiple high-impact domains. With minimal adaptations, it can be applied to crew scheduling problems in transportation systems, such as those addressed in [23,33], where decomposition strategies are key to managing large-scale assignments. It also aligns with the structure of airline crew scheduling problems [56], as well as healthcare scheduling tasks like nurse and operating room rostering [24], which are increasingly critical given ongoing staffing shortages [25]. Moreover, this hybrid framework could inspire advancements in energy systems optimization, such as electricity production planning [51] and emergency base station power distribution [48], where column generation has proven effective and scalable.

In this sense, our methodology not only contributes to the field of hybrid quantum-classical optimization but also opens the door to a broader class of combinatorial problems where quantum and classical methods can work in synergy.

We hope that our findings and methodologies will inspire future benchmarks and contribute to the ongoing progress toward practical applications for solving some of the most challenging decision-making problems.

Acknowledgements. We acknowledge the support of Elizabeth and Zack Kane, whose keen and curious questions sparked some of the ideas explored in this work. We are also grateful to Professor Jeroen Belien from KU Leuven for his valuable feedback given to previous work of L.F.P.A. we are as well thankful for his sharp and insightful questions, which provided inspiration for this work.

Data Accessibility. The instanced used in this study are provided in electronic supplementary material github

Authors' Contributions. L.F.P.A: conceptualization, investigation, writing—original draft; S.D. & R.A. & S.C : conceptualization, writing—review and editing. The four authors gave their final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interests. We declare that we have no competing interests.

Funding. This research benefited from the support of the FMJH Program PGMO.

A. Appendix

(a) Quantum Hybrid Column Generation Algorithm

The following algorithm outlines the complete procedure of the quantum hybrid column generation-based heuristic proposed for solving the Parallel Machine Scheduling Problem with sequence-dependent setup times, as shown in Figure 3 and explained in detail in Section 5 of the current manuscript.

Algorithm 3: Quantum Hybrid Column Generation-based Heuristic for PMSP

Input: Set of jobs $\mathcal{N}$, number of machines $\mathcal{M}$, and a maximum number of iterations b.
Output: Final schedule covering all jobs.

Initialization:
Set iteration counter $\texttt{iter} \leftarrow 0$;
Generate initial set of columns Ω for the RMP (e.g., singleton and exclusion-based feasible columns);

while $\texttt{iter} \leq b$ ***and*** Ω *changed in previous iteration* **do**
  Build and solve the LP relaxation of the RMP with the current Ω using a classical solver (Problem 4.1);
  Extract dual values π from the RMP;
  Run Core Job Selection algorithm to generate set $\mathcal{K}$ (Algorithm 1);
  Run Job Assignment algorithm to generate the set of jobs $\mathcal{Q}_m$ assigned to machine m (Algorithm 2);
  Build the QUBO of the PSP using $(\pi, \mathcal{Q})$ and solve it with a D-Wave annealer to generate temporary columns Ω_{temp} (Problem 5.3);
  Initialize counter $k \leftarrow 1$;
  while $k \leq |\Omega_{temp}|$ **do**
    Check $\Omega_{\text{temp}}[k]$ for:
      (i) Non-duplication,
      (ii) Feasibility (e.g., constraint satisfaction),
      (iii) Negative reduced cost;

    if $\Omega_{temp}[k]$ *satisfies all conditions* **then**
      Update $\Omega \leftarrow \Omega \cup \Omega_{\text{temp}}[k]$;
    else
      $k \leftarrow k + 1$;
    end
  end
  $\texttt{iter} \leftarrow \texttt{iter} + 1$;
end

Final Step:
Solve the MILP RMP with the final Ω to obtain the full schedule;

References

1. Riad Aggoune and Samuel Deleplanque.
   Addressing Machine Unavailability in Job Shop Scheduling: A Quantum Computing Approach.
   In *Metaheuristics*, volume 14753 of *Lecture Notes in Computer Science*, pages 234–245. Springer Nature Switzerland, June 2024.
2. Tameem Albash and Daniel A Lidar.
   Adiabatic quantum computation.
   Reviews of Modern Physics, 90(1):015002, 2018.
3. Ali Allahverdi.
   The third comprehensive survey on scheduling problems with setup times/costs.
   European journal of operational research, 246(2):345–378, 2015.
4. Oliver Avalos-Rosales, Ada Alvarez, and Francisco Angel-Bello.
   A reformulation for the problem of scheduling unrelated parallel machines with sequence and machine dependent setup times.
   In *Proceedings of the international conference on automated planning and scheduling*, volume 23, pages 278–282, 2013.
5. Abhishek Awasthi, Nico Kraus, Florian Krellner, and David Zambrano.
   Real world application of quantum-classical optimization for production scheduling.
   arXiv preprint arXiv:2408.01641, 2024.
6. Kenneth R Baker.
   Introduction to sequencing and scheduling.
   John Wiley & Sons, 1974.
7. Carlo Baldassi and Riccardo Zecchina.
   Efficiency of quantum vs. classical annealing in nonconvex learning problems.
   Proceedings of the National Academy of Sciences, 115(7):1457–1462, 2018.
8. Michel L Balinski and Albert W Tucker.
   Duality theory of linear programs: A constructive approach with applications.
   Siam Review, 11(3):347–377, 1969.
9. David E Bernal, Kyle EC Booth, Raouf Dridi, Hedayat Alghassi, Sridhar Tayur, and Davide Venturelli.
   Integer programming techniques for minor-embedding in quantum annealers.
   In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings 17*, pages 112–129. Springer, 2020.
10. Max Born and Vladimir Fock.
    Beweis des adiabatensatzes.
    Zeitschrift für Physik, 51(3-4):165–180, 1928.
11. Wojciech Bożejko, Jarosław Pempera, Mariusz Uchroński, and Mieczysław Wodecki.
    Quantum annealing-driven branch and bound for the single machine total weighted number of tardy jobs scheduling problem.
    Future generation computer systems, 155:245–255, 2024.
12. Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp.
    Quantum amplitude amplification and estimation.
    arXiv preprint quant-ph/0005055, 2000.
13. Jun Cai, William G Macready, and Aidan Roy.
    A practical heuristic for finding graph minors.
    arXiv preprint arXiv:1406.2741, 2014.
14. Costantino Carugno, Maurizio Ferrari Dacrema, and Paolo Cremonesi.
    Evaluating the job shop scheduling problem on a d-wave quantum annealer.
    Scientific Reports, 12(1):6539, 2022.
15. Jordi Castro, Stefano Nasini, and Francisco Saldanha-da Gama.
    A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method.
    Mathematical programming, 163(1-2):411–444, 2017.
16. Bikas K Chakrabarti, Hajo Leschke, Purusattam Ray, Tatsuhiko Shirai, and Shu Tanaka.
    Quantum annealing and computation: challenges and perspectives.
    Philosophical Transactions of the Royal Society A, 381(2241):20210419, 2023.

17. Jianfu Chen, Chengbin Chu, Abderrahim Sahli, and Kai Li.
A branch-and-price algorithm for unrelated parallel machine scheduling with machine usage costs.
European Journal of Operational Research, 316(3):856–872, 2024.
18. Zhi-Long Chen and Warren B Powell.
Solving parallel machine scheduling problems by column generation.
INFORMS Journal on Computing, 11(1):78–94, 1999.
19. D-Wave D-Wave Systems.
D-wave hybrid documentation.
https://docs.ocean.dwavesys.com/en/stable/docs_hybrid/intro/index.html.
11-2023.
20. Wesley da Silva Coelho, Loïc Henriet, and Louis-Paul Henry.
Quantum pricing-based column-generation framework for hard combinatorial problems.
Physical Review A, 107(3):032426, 2023.
21. George B Dantzig and Philip Wolfe.
Decomposition principle for linear programs.
Operations research, 8(1):101–111, 1960.
22. Samuel Deleplanque, Amina El Yaagoubi, Téo Gras, Florent Descamps, Julia Mourrier, and Isabelle Lefebvre.
Applying analog quantum computing to solve optimisation problems: case studies on max-cut and cvrp.
International Journal of Systems Science: Operations & Logistics, 12(1):2463530, 2025.
23. Martin Desrochers and François Soumis.
A column generation approach to the urban transit crew scheduling problem.
Transportation science, 23(1):1–13, 1989.
24. Christine Di Martinelly and Nadine Meskens.
A bi-objective integrated approach to building surgical teams and nurse schedule rosters to maximise surgical team affinities and minimise nurses' idle time.
International Journal of Production Economics, 191:323–334, 2017.
25. European Commission.
Launch of the first eu action to address nurse shortages shows positive impact of european health union, January 2025.
Accessed: 2025-05-10.
26. Edward Farhi, Jeffrey Goldstone, and Sam Gutmann.
A quantum approximate optimization algorithm.
arXiv preprint arXiv:1411.4028, 2014.
27. Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda.
A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem.
Science, 292(5516):472–475, 2001.
28. Paulo M França, Michel Gendreau, Gilbert Laporte, and Felipe M Müller.
A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times.
International Journal of Production Economics, 43(2-3):79–89, 1996.
29. Michel Gendreau, Gilbert Laporte, and Eduardo Morais Guimarães.
A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times.
European Journal of Operational Research, 133(1):183–189, 2001.
30. Laleh Ghalami and Daniel Grosu.
Scheduling parallel identical machines to minimize makespan: A parallel approximation algorithm.
Journal of Parallel and Distributed Computing, 133:221–231, 2019.
31. Alain Guinet.
Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria.
International journal of production research, 31(7):1579–1594, 1993.

32. Renichiro Haba, Takuya Mano, Ryosuke Ueda, Genichiro Ebe, Kohei Takeda, Masayoshi Terabe, and Masayuki Ohzeki.
Routing and scheduling optimization for urban air mobility fleet management using quantum annealing.
Scientific Reports, 15(1):4326, 2025.
33. Dennis Huisman.
A column generation approach for the rail crew re-scheduling problem.
European Journal of Operational Research, 180(1):163–173, 2007.
34. Kazuki Ikeda, Yuma Nakamura, and Travis S Humble.
Application of quantum annealing to nurse scheduling problem.
Scientific reports, 9(1):12837, 2019.
35. Sorin Istrail.
Statistical mechanics, three-dimensionality and np-completeness: I. universality of intracatability for the partition function of the ising model across non-planar surfaces.
In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 87–96, 2000.
36. Tadashi Kadowaki and Hidetoshi Nishimori.
Quantum annealing in the transverse ising model.
Physical Review E, 58(5):5355, 1998.
37. Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi.
Optimization by simulated annealing.
science, 220(4598):671–680, 1983.
38. Wen Chiung Lee, Chin Chia Wu, and Peter Chen.
A simulated annealing approach to makespan minimization on identical parallel machines.
International journal of advanced manufacturing technology, 31(3-4):328–334, 2006.
39. J.K. Lenstra and A.H.G. Rinnooy Kan.
Computational complexity of discrete optimization problems.
In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization I*, volume 4 of *Annals of Discrete Mathematics*, pages 121–140. Elsevier, 1979.
40. Andrew Lucas.
Ising formulations of many np problems.
Frontiers in physics, 2:5, 2014.
41. Jeffrey Marshall, Andrea Di Gioacchino, and Eleanor G Rieffel.
Perils of embedding for sampling problems.
Physical Review Research, 2(2):023020, 2020.
42. Dell Amico Mauro and Silvano Martello.
Optimal scheduling of tasks on identical parallel processors.
ORSA Journal on Computing, 7(2):191, 1995.
43. Alexandre S. Mendes, Felipe M. Müller, Paulo M. França, and Pablo Moscato.
Comparing meta-heuristic approaches for parallel machine scheduling problems.
Production planning & control, 13(2):143–154, 2002.
44. Ethel Mokotoff.
An exact algorithm for the identical parallel machine scheduling problem.
European Journal of Operational Research, 152(3):758–769, 2004.
45. F. Orts, A.M. Puertas, G. Ortega, and E.M. Garzón.
Quantum annealing solution for the unrelated parallel machine scheduling with priorities and delay of task switching on machines.
Future Generation Computer Systems, 148:514–523, 2023.
46. Luis Fernando Pérez Armas, Stefan Creemers, and Samuel Deleplanque.
Solving the resource constrained project scheduling problem with quantum annealing.
Scientific Reports, 14(1):16784, 2024.
47. A Prabhakar, P Shah, U Gautham, V Natarajan, V Ramesh, N Chandrachoodan, and S Tayur.
Optimization with photonic wave-based annealers.
Philosophical Transactions of the Royal Society A, 381(2241):20210409, 2023.
48. Hu Qin, Anton Moriakin, Gangyan Xu, and Jiliu Li.
The generator distribution problem for base stations during emergency power outage: A branch-and-price-and-cut approach.
European Journal of Operational Research, 318(3):752–767, 2024.

49. Atanu Rajak, Sei Suzuki, Amit Dutta, and Bikas K Chakrabarti.
Quantum annealing: An overview.
Philosophical Transactions of the Royal Society A, 381(2241):20210417, 2023.
50. Eleanor G. Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy.
A case study in programming a quantum annealer for hard operational planning problems.
Quantum information processing, 14(1):1–36, 2015.
51. Antoine Rozenknop, Roberto Wolfler Calvo, Laurent Alfandari, Daniel Chemla, and Lucas Létocart.
Solving the electricity production planning problem by a column generation based heuristic.
Journal of Scheduling, 16:585–604, 2013.
52. H Sadeghi.
Quantum computing with neutral atoms with applications in energy.
In *EAGE Seventh High Performance Computing Workshop*, volume 2023, pages 1–3. European Association of Geoscientists & Engineers, 2023.
53. Tomasz Śmierzchalski, Jakub Pawłowski, Artur Przybysz, Łukasz Pawela, Zbigniew Puchała, Mátyás Koniorczyk, Bartłomiej Gardas, Sebastian Deffner, and Krzysztof Domino.
Hybrid quantum-classical computation for automatic guided vehicles scheduling.
Scientific Reports, 14(1):21809, 2024.
54. Tony T Tran, Arthur Araujo, and J Christopher Beck.
Decomposition methods for the parallel machine scheduling problem with setups.
INFORMS Journal on Computing, 28(1):83–95, 2016.
55. Cristiano Arbex Valle, Leonardo Conegundes Martinez, Alexandre Salles Da Cunha, and Geraldo R Mateus.
Heuristic and exact algorithms for a min–max selective vehicle routing problem.
Computers & Operations Research, 38(7):1054–1065, 2011.
56. Pamela H Vance, Cynthia Barnhart, Ellis L Johnson, and George L Nemhauser.
Airline crew scheduling: A new formulation and decomposition algorithm.
Operations Research, 45(2):188–200, 1997.
57. Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy.
Quantum optimization of fully connected spin glasses.
Physical Review X, 5(3):031040, 2015.
58. Xiaoyan Zhu and Wilbert E. Wilhelm.
Scheduling and lot sizing with sequence-dependent setup: A literature review.
IIE transactions, 38(11):987–1007, 2006.
59. Marko Æurasevic and Domagoj Jakobovic.
Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey.
The Artificial intelligence review, 56(4):3181–3289, 2023.