LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

EXPANDING JOB SHOP SCHEDULING PROBLEM TO THE CONTEXT OF SERVICE PRIORITIZATION

Erico Souza Teixeira
CESAR
Rua Bione, Cais do Apolo, 220, Recife - PE, 50030-390
est@cesar.org.br

Pâmela Thays Lins Bezerra
CESAR School
Avenida, Cais do Apolo, 77, Recife - PE, 50030-220
ptlb@cesar.school

Arthur Carlos da Rocha e Silva
CESAR School
Avenida, Cais do Apolo, 77, Recife - PE, 50030-220
acrs@cesar.school

Marcelo Lima Arcoverde
CESAR School
Avenida, Cais do Apolo, 77, Recife - PE, 50030-220
mla@cesar.school

Gabriel Pires Aquino de Carvalho
CESAR School
Avenida, Cais do Apolo, 77, Recife - PE, 50030-220
gpac@cesar.school

ABSTRACT

In the Brazilian Energy Sector, effective management of maintenance services is critical due to the high complexity and urgency variations of tasks. This challenge is known as Service Prioritization, and here, we extend the classical Job Shop Scheduling Problem (JSSP) to address it. Utilizing Adiabatic Quantum Computing and Exhaustive Classical approach, the extended JSSP formulation is evaluated, demonstrating its correctness and the potential of Quantum Computing to achieve efficient solutions. Results indicate that the Binary Quadratic Model solver performs comparably to classical exact solvers. Ultimately, this study marks a significant step towards integrating quantum technologies into the Brazilian Energy Sector, underscoring the potential benefits and areas for future exploration as quantum computing evolves.

KEYWORDS. Job Shop Scheduling. Service Prioritization. Quantum Computing.

Paper topics
SS-QPO - Quantum Methods and their Applications in Operational Research
OC - Combinatorial Optimization
L&T - Logistics and Transport

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

1. Introduction

In the Energy Sector, failures in the transmission and distribution of energy can lead to harmful inconveniences for society and the companies responsible for these services. Given this, maintenance services are essential for the correct operation of energy companies. However, the number of services is usually much higher than the available operation teams and equipment required to attend them. Additionally, different services have different levels of urgency and complexity. Managing the correct order of execution of these services and allocating the proper teams and materials are daily challenges faced by this sector. This challenge is known as **Service Prioritization (SP)** and is commonly studied in other industries, such as transportation and logistics (Chong et al. [2019]).

One way of modeling and solving the SP problem is using the Job Shop Scheduling Problem (JSSP) (Jain and Meeran [1999]). This resource allocation problem consists of a finite set of jobs that a finite set of machines must process. Additionally, each job consists of a series of operations, which must be scheduled in a predefined order, with each machine processing only one operation at a time (Xiong et al. [2022a]). The correlation between JSSP and SP becomes clear when we map the jobs as **services**, the operations as **tasks** that comprise each service, and the machines as **operation teams**.

JSSP, however, is proven to be an NP-hard combinatorial problem (Xiong et al. [2022b]), requiring an exponential execution time to find an optimal solution. Currently, it is mainly solved through approximation methods and heuristics (Zhang et al. [2023]). Additionally, the original JSSP model does not capture the complexity of some real-world scenarios. Given this, many variations of JSSP were proposed over the years (Xie et al. [2019])(Wang and Yu [2010])(Chan et al. [2006]), each one adding extra complexity to the original formulation.

Recent works have explored the use of Quantum Computing (QC) for solving JSSP (Venturelli et al. [2016]) (Carugno et al. [2022])(Kurowski et al. [2020]). QC is a new way of computation that harnesses quantum mechanics effects and properties to provide potentially faster and more accurate results, particularly by solving certain types of problems more effectively than classical computers. The main difference between QC and classical computing is its basic processing unit, the qubit. Contrary to the usual bit, which has a deterministic behavior storing the values 0 and 1 per time, qubits are capable of representing multiple combinations of the base states 0 and 1, enabling quantum computers to process more data with fewer processing units and, consequently, solve complex problems, such as JSSP, in less time.

Given this context, this paper aims **to present an extension of the original JSSP** formulation to better attend to the SP requirements in the Brazilian Energy Sector. Additionally, this new formulation is **evaluated using QC** to explore the challenges and opportunities of this promising technology. The results obtained are, therefore, an important step in adopting quantum technologies in the Brazilian Energy sector.

This document is organized as follows: **Section 2** details combinatorial optimization; **Section 3** describes JSSP and the proposed solution; **Section 4** explains a QC technology known as Adiabatic Computing; and finally **Sections 5** and **6** present the obtained results and the main conclusions, respectively.

2. Combinatorial Optimization

Combinatorial Optimization (CO) problems are a special class of optimization problems that deals with discrete variables (Du et al. [2022]). Their main goal is to find an optimal object from a finite set of objects. In recent years, two important scientific advances have shown promising alternatives for solving CO: Quantum Computing (QC) and Quadratic Unconstrained Binary Optimization (QUBO). Many studies in QC have shown that this technology enables a faster optimal

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

solution search for some optimization problems (Bhatia and Ramkumar [2020]). Additionally, the mathematical formulation called QUBO has emerged as a simple way to model a variety of CO problems (Quintero and Zuluaga [2022]).

2.1. QUBO

The Quadratic Unconstrained Binary Optimization (QUBO) problem represents a category of mathematical problems featuring binary variables, which can exhibit both quadratic and linear characteristics. Formally, this model can be described as (nez Barrera et al. [2024]):

$$f_{obj}(\bar{x}) = \sum_{i}^{n} a_i x_i + \sum_{i}^{n} \sum_{j>i}^{n} b_{ij} x_i x_j \quad \text{and} \quad \bar{x} = (x_1, x_2, ..., x_n) \tag{1}$$

where x is a column vector of binary decision variables, n is the number of variables, f_{obj} is the objective function to minimize, and a_i and b_{ij} are the linear and quadratic coefficients, respectively. The essence of QUBO models lies in finding the optimal combination of variables that yields the most favorable outcome while avoiding penalties.

3. JSSP and its variants

The JSSP is a popular CO problem within operations research and manufacturing. In this problem, a set of jobs $J = \{j_1, j_2, ..., j_n\}$, each comprising of a sequence of operations $O = \{On_1, On_2, ..., On_{Ln}\}$, must be scheduled across a list of machines $M = \{m_1, m_2, ..., m_y\}$. Three fundamental constraints underpin this problem: (1) **single operation**, meaning that each operation is executed only once to avoid redundant actions; (2) **shared machine**, which can only perform one operation simultaneously to prevent parallelism; and (3) **precedence restriction**, meaning that operations of the same job must be carried out in a specific order (Xiong et al. [2022b]).

Typically, the primary objective of JSSP is to minimize a specific scheduling metric, such as Makespan (the total time to complete all jobs) and Total Flow Time (the total time a job spends in the system, from start to completion).

Several extensions of JSSP have been proposed to address different real-world scenarios and challenges. Some notable examples are: **Flexible JSSP**, where each operation can be processed on one of several alternative machines, introducing flexibility into the scheduling process (Xie et al. [2019]); **JSSP with Maintenance**, where machines may require periodic maintenance, leading to inoperative times (Wang and Yu [2010]); **JSSP with Resource Constraints**, which introduces constraints related to limited resources, such as labor, raw materials, or energy (Chan et al. [2006]).

Apart from being a non-exhaustive list, these variations highlight the diversity of challenges that can be addressed within JSSP, reflecting the complexity of real-world manufacturing and production environments.

3.1. QUBO Formulation

The first work to propose a QUBO formulation for JSSP was Venturelli et al. [2016], which is a proof of concept of applying QC to solve JSSP. Here, we adopt the same formulation, in which each variable can be defined as follows:

$$x_{i,t} = \begin{cases} 1 & \text{if operation } O_i \text{ starts at time } t, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where $x_{i,t}$ represents whether a specific operation of O_i of a given job starts at time period t. It is possible to observe that the number of variables is proportional to the number of operations and possible start times.

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

With this definition, the three constraints of the JSSP basic formulation can be rewritten as QUBO formulations. The first one, related to the imperative of a single operation, can be expressed by Constraint 3.

$$h_1 = \left(\sum_t x_{i,t} = 1 \text{ for each } i \right) \rightarrow \sum_i \left(\sum_t x_{i,t} - 1 \right)^2 \tag{3}$$

The second constraint, where a machine may only execute a single operation at a time (shared machines), can be described by Constraint 4.

$$h_2 = \left(\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} = 0 \quad \text{for each } m. \right) \rightarrow \sum_m \left(\sum_{(i,t,k,t') \in R_m} x_{i,t} x_{k,t'} \right) \tag{4}$$

$$\begin{aligned} R_m &= A_m \cup B_m, \\ A_m &= \{(i,t,k,t') : (i,k) \in I_m \times I_m, i \neq k, 0 \leq t, t' \leq T, 0 < t' - t < p_i\}, \\ B_m &= \{(i,t,k,t') : (i,k) \in I_m \times I_m, i < k, t' = t, p_i > 0, p_j > 0\}. \end{aligned}$$

where R_m is a set of operations executed on that specific machine m, and I_m is the set of indices of all of the operations that have to be executed on machine m. This constraint will penalize the solution if the operation i starts at time t and the operation k starts at time $t' \leq t + p_i$, where p_i is the execution time of i, at the same machine.

Finally, all operations of a job must be executed in a determined order (precedence restriction), which can be expressed by Constraint 5.

$$h_3 = \left(\sum_{\substack{k_{n-1} < i < k_n \\ t + p_i > t'}} x_{i,t} x_{i+1,t'} = 0 \quad \text{for each } n, \right) \rightarrow \sum_n \left(\sum_{\substack{k_{n-1} < i < k_n \\ t + p_i > t'}} x_{i,t} x_{i+1,t'} \right) \tag{5}$$

where $i \in \{On_1, ..., On_{Ln-1}\}$. Each term represents a possible precedence violation between operations i and $i+1$ in task n. The condition $t + p_i > t'$ checks whether the operation i ends after the operation $i+1$ begins, which is a violation. If this condition is satisfied, the product $x_{i,t} x_{i+1,t'}$ will be equal to 1, contributing to the sum and violating the restriction. Otherwise, if the condition is not met, the product will be equal to 0.

By doing the linear combination of the constraints 3 to 5 and multiplying each one by the empirically adjusted factors (α, β, and γ) which represent the importance of each restriction, we have the final formulation for JSSP described by Equation 6.

$$H_t = \alpha h_1 + \beta h_2 + \gamma h_3 \tag{6}$$

3.2. JSSP New formulation: Service-JSSP

This section provides the adaptation of JSSP traditional formulation into QUBO format, incorporating additional constraints for optimizing its applicability in Service Prioritization (SP) in the energy sector. Considering a set of operation teams and maintenance services, where each service is composed of a list of operations, some of the common scenarios are: (1) Different operation teams may perform the same operation. However, the specified operation should be executed **only**

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

once; (2) Some operation teams have **limited working** hours. For example, some may work morning and afternoon shifts, while others are available only at night; (3) Each operation may require a **set of equipment** for its completion. This set is limited and must be shared among the various operation teams. However, when a team uses some equipment to perform a specific operation, other teams cannot use it until the operation is completed.

Considering the above, the following variations are introduced to JSSP: (1) Operations can be performed not only by one machine but by one among **a set of selected machines**, providing greater flexibility in operations distribution; (2) Some machines may be **inactive during certain periods**; (3) Some operations require **a set of resources (i.e., equipment)** that a machine uses and cannot be shared with others during its execution.

The first variation requires adapting the three constraints specified for the traditional JSSP. The second variation can be implemented by removing from the problem variables related to a specific team during its periods of inactivity (variable reduction). Finally, the third condition requires defining a new constraint (equipment utilization constraint) that ensures at most one team uses the equipment at each moment, increasing the number of constraints to 4 in the new method. Additionally, we assume that operations cannot have zero duration, simplifying the definitions of the constraints related to the exclusivity of the operations and the use of equipment.

3.3. New QUBO Formulation

To adequately represent the problem functions in the QUBO format, some definitions are necessary: I is the set of indices of all operations in O; J is the set of all Jobs; M_i is the set of machines capable of performing operation of index $i \in I$; I_e is the set of indices of operations that require specific equipment e; E_i is the set of equipment required by operation of index i.

3.3.1. Problem Variable

For this version of the problem, an operation can be performed by a determined set of machines, which must utilize a defined set of equipment. The QUBO variables for this context must now, therefore, include a new index to record which machine executed the operation, while the equipment usage can be implemented without adding another index. The problem variable is defined in Equation 7.

$$x_{i,m,t} = \begin{cases} 1 & \text{if operation } O_i \text{ starts at time } t \text{ by machine } m \in \text{M}_i\text{''}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

It is worth observing that the number of variables increases as more machines can perform each operation individually, in addition to being influenced by the number of operations and the timespan defined for the problem.

3.3.2. Single Start Constraint for Operations

Equation 8 represents a reformulation of Equation 3 to incorporate the constraint that each operation can start only once.

$$h_1(\bar{x}) = \sum_i \left(\sum_{m \in M_i} \left(\sum_t x_{i,m,t} - 1 \right)^2 \right) \tag{8}$$

This equation ensures that for each operation i, the sum of all the possible starting times t across all capable machines m from the set M_i is equal to one. This enforces that each operation starts exactly one time, penalizing it when it is executed more than once or not executed at all.

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

3.3.3. Operation Exclusivity Constraint

Equation 9 shows that each machine can perform only one operation at any given time.

$$h_2(\bar{x}) = \sum_{m} \left(\sum_{(i,t,i',t')} \in R_m \left(x_{i,m,t} \cdot x_{i',m,t'} \right) \right) \tag{9}$$

$$R_m = \{(i,t,i',t') : (i,i') \in I_m \times I_m; i \neq i'; t' \leq T; t < t' < t + p_i; t' \leq T\}$$

This constraint ensures that for each machine m, no two operations i and i' overlap in time. The set R_m consists of pairs of operations and their respective times (i,t) and (i',t') where: (1) Both operations are executable by the same machine m; (2) The operations are distinct ($i \neq i'$); (3) The start time t' of operation i' falls within the duration p_i of operation i after its start at t. This restriction ensures that a machine can only perform one operation at a time.

3.3.4. Operation Precedence Constraint

Equation 10 defines that operations within a job must follow a specified order:

$$h_3(\bar{x}) = \sum_{n} \left(\sum_{\substack{k_{n-1}<i<k_n, \\ t+p_i>t'}} \left(\sum_{m \in M_i} x_{i,m,t} \cdot \sum_{m' \in M_{i+1}} x_{i+1,m',t'} \right) \right) \tag{10}$$

where within each job (segmented by k_{n-1} and k_n indicating operation indices), every operation i must be completed before the beginning of the next operation $i+1$. It does so by ensuring that, for any given operation i, the end time ($t + p_i$, where p_i is the duration of the operation) must be less than or equal to the start time t' of the next operation $i+1$. The sums across machines m for operation i and m' for operation $i+1$ ensure this condition is met across all possible machines that could execute these operations. This constraint enforces, therefore, the strict sequence of operations that must be respected within each job.

3.3.5. Equipment Utilization Constraint

Equation 11 guarantees that each equipment can only be used by at most one machine at any given time.

$$h_4(\bar{x}) = \sum_{(i,m,t,i',m',t') \in R_e} \left(x_{i,m,t} \cdot x_{i',m',t'} \right) \tag{11}$$

$$R_e = \{(i,m,t,i',m',t') : (i,i') \in I \times I_e; i \neq i'; (m,m') \in M_i \times M_{i'}; \\ 0 \leq t \leq T; t \leq t' < t + p_i; t' < T\}$$

where each set within R_e includes a combination of all operations with those requiring equipment e, indicated by $(i,i') \in I \times I_e$. This constraint takes into account all the equipment used by each operation throughout the entire duration from the start to the end of that operation, as defined by $t < t' < t + p_i$, across all possible instances of time, given by $0 \leq t \leq T$, considering all machines that can perform operation i and i', indicated by $(m,m') \in M_i \times M_{i'}$. This effectively prevents the allocation of equipment to different operations at the same time interval, ensuring that there is no overlap in equipment usage and respecting the necessary duration for the completion of each operation.

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

3.4. Objective Function

In our approach, differently from Venturelli et al. [2016]), JSSP's objective function seeks to identify the solution that results in the **shortest total completion time for all jobs**. This objective can be achieved by considering the difference between the completion time of the last operation of each job and the minimum time required for its completion, which is the sum of the duration of all its operations.

Given the index of a job j, the index of the last operation of this job is denoted by k_j. The minimum time required to complete j is represented by P_j, described in Equation 12.

$$P_j = \sum_{k_{j-1}+1 \leq i \leq k_j} p_i \tag{12}$$

where p_i is the duration of operation i. To avoid solutions where jobs are completed after their minimum required time, we can introduce a penalty value given by a function, $f_{penalty}$, to all variables associated with the final operation of each job for all machines that can execute it, and for any time instance that exceeds each job's minimum completion time. This penalty, therefore, is a function of the instant t and the job j across all machines capable of executing it. The objective function can be described as in Equation 13.

$$f_{obj}(\bar{x}) = \sum_{j \in J} \sum_{m \in M_{k_j}} \sum_{P_j < t < T} f_{penalty}(j,t) \cdot x_{k_j,m,t} \tag{13}$$

To find the penalty function, we can establish that it must satisfy two essential conditions: (1) The further the job is from its minimum completion time, the higher the value of the function; (2) The time for all jobs to be completed should be minimum and take precedence over the total time the jobs were delayed, meaning that a single job being delayed by two-time units is considered worse than all jobs being delayed by one-time unit.

The penalty function can be written as a function of d, the difference between the completion time of the last operation of a job j and the minimum time required for the job's completion $d = t - P_j$. Considering that all restrictions are satisfied and all jobs finish with the same difference to its minimum completion time, d, the value of the objective function would be $N_j \cdot f_{penalty}(d)$. If only one job is delayed by one additional time unit beyond d while all other jobs are executed during its minimum time, the result would be $f_{penalty}(d+1)$. Thus, we seek a function $f_{penalty}$ as described by Equation 14.

$$f_{penalty}(d+1) > N_j \cdot f_{penalty}(d) \tag{14}$$

This condition can be proven for $f_{penalty}(d) = a^d$: If $f_{penalty}(d) = a^d$, then $f_{penalty}(d+1) = a^{d+1} = a \cdot a^d$. We need to prove that $a \cdot a^d > N_j \cdot a^d$ for $N_j \geq 1$ and $a > 1$. Simplifying, we find $a > N_j$. Therefore, by choosing an $a > N_j$, the condition $a^{d+1} > N_j \cdot a^d$ is satisfied. Hence, a is chosen as $N_j + 1$. The objective function results in Equation 15.

$$f_{obj}(\bar{x}) = \sum_{j \in J} \sum_{m \in M_{k_j}} \sum_{P_j < t < T} (N_j + 1)^{t - P_j} x_{k_j,m,t} \tag{15}$$

3.5. Problem Hamiltonian

The problem Hamiltonian, denoted by H_T, gives the energy of a solution and consists of the sum of the objective function with the four constraints multiplied by their respective Lagrange coefficients, L_1, L_2, L_3, and L_4, as defined at Equation 16.

$$\text{Minimize } H_T(\bar{x}) = L_1 h_1(\bar{x}) + L_2 h_2(\bar{x}) + L_3 h_3(\bar{x}) + L_4 h_4(\bar{x}) + f_{obj}(\bar{x}) \tag{16}$$

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

3.6. Lagrange Parameters

The Lagrange parameters must be set for all solutions that violate one restriction to have a greater value than any valid solution. For this to happen, those parameters should have a high value, higher than the maximum possible value of the objective function (Equation 17).

$$L_k > N_j N_m (N_j + 1)^T, \text{ for each } k \tag{17}$$

For this to work, the Lagrange parameters are set to be two times the maximum value of the objective function (as described in Equation 18).

$$L_1 = L_2 = L_3 = L_4 = \left(N_j N_m (N_j + 1)^T\right) * 2 \tag{18}$$

3.7. Variable Reduction

Many variables can be eliminated without impacting the overall solution to reduce the complexity of the QUBO problem. These variables can be discarded because their value of one would always lead to an invalid solution. Consequently, these variables, along with all their interactions, are removed from the problem.

An example of this variable type is those related to the last operation of the jobs starting at a time that results in their completion beyond the timespan of the problem. Similarly, variables representing job operations that would start after the maximum possible start time of their preceding operations can also be eliminated. Additionally, we can specify periods during which a machine is inoperable. All variables associated with that machine, as well as any variables that would result in a start time that would lead to operations being executed by that machine during its inoperable moments, can also be removed.

4. Adiabatic Computing

According to Albash and Lidar [2018], Adiabatic Quantum Computing (AQC) stands as a fundamental paradigm for QC, drawing upon the adiabatic theorem (Born and Fock [1928]) in quantum mechanics. In broad terms, AQC commences with a quantum system initially in its ground state and is associated with a simple Hamiltonian ($\hat{H}_0$), a function that represents the total energy of a system. Over time, this Hamiltonian smoothly changes to represent the problem Hamiltonian ($\hat{H}_f$), wherein the ground state encapsulates the solution to a computational problem.

4.1. Quantum Annealing

Morita and Nishimori [2008] describez Quantum Annealing (QA) as a specific implementation of AQC, with the time evolution of the quantum system drawing inspiration from the classical annealing process in metallurgy, where a material is heated and slowly cooled to remove defects and optimize its structure. QA is a heuristic quantum approximation method since the switch from $\hat{H}_0$ to $\hat{H}_f$ is determined heuristically, and the adiabatic conditions are not guaranteed. As a result, QA is particularly well-suited for encoding binary CO problems expressed in QUBO form.

4.2. D-Wave Computers

One of the most popular QA Systems is D-Wave (DWAVE [2024]), and currently, its more powerful device, the Advantage, has 5000 qubits and an optimized topology, capable of solving complex commercial problems with more than 1 million variables. The company is a pioneer as the first commercial supplier of quantum computers and software and services, developing its own Python-based open-source SDK, D-Wave Ocean. Additionally, it provides a cloud service, the D-Wave Leap, for real-time remote access to its devices. The experiments reported in this article were performed directly through the Leap interface.

The essential elements of a D-Wave solution are the **samplers** and **solvers**. A sampler is a process that analyzes objective functions formulated as QUBO to find likely solutions by sampling

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

low-energy states. These samplers run on a physical device called a solver, which can be classical (a regular computer), hybrid (combining classical and quantum resources), or purely quantum. In our experimental analyses, we have used a classical solver, **Exact Solver**, which implements the exhaustive search, and **DWaveSampler**, a pure quantum-based solver provided by D-Wave Systems.

5. Implementation Validation

Experiments were conducted to validate the implementation of the proposed method, Service-JSSP, and to analyze the performance of both classical and quantum methods. This section details the test scenarios used and the obtained results.

5.1. Test Scenarios

To verify the formulation's implementation, 18 test scenarios were created covering possible combinations of specific conditions for each constraint (details on Table 2). Additionally, three scenarios were included to evaluate the moments when the machines are not allowed to operate (known as downtime). These conditions were selected to have a maximum or minimum impact for each constraint so the behavior of opposite extreme situations could be analyzed. For example, for the equipment usage constraint, "0" indicates that operations do not use the equipment, meaning it has no effect in the scheduling process. Meanwhile, "1" indicates that all operations use the same equipment, where the constraint would have the greatest impact. Finally, "2" represents a scenario where each operation requires different equipment, resulting in no effect. More details for the different Service-JSSP constraints are explained in Table 1.

Table 1: Restriction Scenarios Description

Restriction	Condition	Description
Exclusivity	0	All operations are performed by different machines
	1	All operations are performed by the same machine
	2	All operations can be performed by all machines
Precedence	0	All operations are present in different jobs
	1	All operations are present in the same job
Equipment	0	All operations do not use equipment
	1	All operations require the same equipment
	2	All operations use different equipment
Machine Downtimes	0	No downtimes
	1	Only one available time*

*This means that each machine has only one available instance of time that is not marked as downtime

Tests were conducted using the **Exact Solver (classic method)** and **DWaveSampler (quantum method)**, as explained in Section 4.2. While the Exact Solver (ES) will try every possible solution, the D-Wave solver (DS) will run the test for a maximum of 1000 iterations. More iteration generally increases the chances of finding a better solution because it allows the D-Wave sample to explore different regions of the solution space. It is worth noticing that the ES instances were performed in a local machine with an AMD Ryzen 5 5600X CPU with 6 cores 3.9 GHz, and 32 GB of RAM memory.

Each test was then executed using **two** and **four operations**, with the number of machines, equipment, and jobs varying according to each test condition (Table 1). For example, if all operations are executed by the same machine, the test uses only one machine. Alternatively, if each operation is executed by a different machine, the test uses a number of machines equal to the number of operations. The minimum energy across all solutions is used as the evaluation metric.

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

Table 2: Combination of different constraints

ID	Exclusivity	Precedence	Equipment	Machine Downtimes
TC_001	0	0	0	0
TC_002	1	0	0	0
TC_003	2	0	0	0
TC_004	0	1	0	0
TC_005	1	1	0	0
TC_006	2	1	0	0
TC_007	0	0	1	0
TC_008	1	0	1	0
TC_009	2	0	1	0
TC_010	0	1	1	0
TC_011	1	1	1	0
TC_012	2	1	1	0
TC_013	0	0	2	0
TC_014	1	0	2	0
TC_015	2	0	2	0
TC_016	0	1	2	0
TC_017	1	1	2	0
TC_018	2	1	2	0
TC_019	0	0	0	1
TC_020	2	0	0	1
TC_021	0	0	2	1

5.2. Result and Discussion

Table 3 summarizes all the significant results obtained. It is worth noticing that while some instances have a minimum energy above 0, none of the cases violated the constraints. These values result from the objective function (Equation 15), where certain constraints prevent the last operation of a job from being executed at the minimum execution time. For example, in TC_02, the same machine executes all operations, each one with only one job. Consequently, the operations are executed sequentially, and since the jobs have only one operation each, the execution time of the last operation runs beyond its minimum execution time, increasing the energy.

Furthermore, for the tests conducted with two operations and most cases with four operations, the DS results align perfectly with those of the ES, showing that the quantum approach successfully finds optimal solutions. However, some four operations scenarios could not be executed with ES (represented as $-$ in table 3) due to hardware requirements necessary to perform an exhaustive search. This search requires a high amount of memory, highlighting the need for a more efficient method to navigate through the solution space, a need that the D-Wave Solver effectively addresses.

These results are promising and demonstrate the potential of QC in identifying efficient solutions. Furthermore, as the results obtained by Exact Solver are the same, this demonstrates BQM's ability to get results comparable to exhaustive solutions.

6. Conclusion

In this study, we have extended the JSSP to address the specific Service Prioritization (SP) requirements in the Brazilian Energy Sector. By incorporating the unique challenges of this sector,

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

Table 3: Lowest Energies obtained using 2 and 4 operations (2 Op and 4 Op, respectively). DS means D-Wave solver (quantum) and ES means the Exact Solver (classic)

TC	DS 2 Op	ES 2 Op	DS 4 Op	ES 4 Op
TC_01	0	0	0	0
TC_02	3	3	155	155
TC_03	0	0	10	-
TC_04	0	0	0	0
TC_05	0	0	0	0
TC_06	0	0	0	0
TC_07	3	3	155	155
TC_08	3	3	155	155
TC_09	3	3	155	-
TC_10	0	0	0	0
TC_11	0	0	0	0
TC_12	0	0	0	0
TC_13	0	0	0	0
TC_14	3	3	155	155
TC_15	0	0	35	-
TC_16	0	0	0	0
TC_17	0	0	0	0
TC_18	0	0	0	0
TC_19	3	3	155	155
TC_20	3	3	155	155

we have developed a formulation that can better manage the scheduling of maintenance services, considering the varying levels of urgency and task complexity.

In addition, our findings show that Quantum Computing (QC), specifically using Adiabatic Quantum Computing (AQC), has the potential to offer efficient solutions for this expanded JSSP formulation. The performance of the Binary Quadratic Model (BQM) solver, as demonstrated by the test cases, indicates that QC can achieve results comparable to those of classical exact solvers. In certain scenarios, it can even reach unattainable solutions through exhaustive methods.

Overall, this work marks a significant step towards integrating quantum technologies in the Brazilian Energy Sector, highlighting potential benefits and areas for further exploration. As QC continues to evolve, its application to CO problems like JSSP can provide more effective and efficient solutions, driving advancements in various industries.

In future work, we plan to enhance our comparative analysis by exploring other D-Wave Quantum Solvers, classic heuristics, such as Simulated Annealing, and better exact algorithms for solving QUBO problems. Additionally, we want to extend our current analysis beyond QA systems, testing on, for example, IBM gate-based systems.

References

Albash, T. and Lidar, D. A. (2018). Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002. ISSN 0034-6861. URL https://link.aps.org/doi/10.1103/RevModPhys.90.015002.

Bhatia, V. and Ramkumar, K. (2020). An efficient quantum computing technique for cracking rsa

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

using shorâs algorithm. In *2020 IEEE 5th international conference on computing communication and automation (ICCCA)*, p. 89–94. IEEE.

Born, M. and Fock, V. (1928). Beweis des adiabatensatzes. *Zeitschrift fÃ¼r Physik*, 51:165–180. URL https://doi.org/10.1007/BF01343193.

Carugno, C., Ferrari Dacrema, M., and Cremonesi, P. (2022). Evaluating the job shop scheduling problem on a d-wave quantum annealer. *Scientific Reports*, 12(1):6539.

Chan, F., Wong, T., and Chan, L. (2006). Flexible job-shop scheduling problem under resource constraints. *International journal of production research*, 44(11):2071–2089.

Chong, A. K. W., Mohammed, A. H., Abdullah, M. N., and Rahman, M. S. A. (2019). Maintenance prioritization â a review on factors and methods. *Journal of Facilities Management*, 17(1):18–39. ISSN 1472-5967. URL https://doi.org/10.1108/JFM-11-2017-0058.

Du, D., Pardalos, P. M., Hu, X., and Wu, W. (2022). *Introduction to Combinatorial Optimization*. Springer.

DWAVE (2024). About d-wave. URL https://www.dwavesys.com/company/about-d-wave/. Access: 2024-08-08.

Jain, A. S. and Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2):390–434. URL https://EconPapers.repec.org/RePEc:eee:ejores:v:113:y:1999:i:2:p:390-434.

Kurowski, K., WÈ©glarz, J., Subocz, M., Różycki, R., and Waligóra, G. (2020). Hybrid quantum annealing heuristic method for solving job shop scheduling problem. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part VI 20*, p. 502–515. Springer.

Morita, S. and Nishimori, H. (2008). Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12):125210. ISSN 0022-2488. URL http://aip.scitation.org/doi/10.1063/1.2995837.

nez Barrera, J. A. M., Willsch, D., Maldonado-Romo, A., and Michielsen, K. (2024). Unbalanced penalization: a new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms. *Quantum Science and Technology*, 9(2):025022. URL https://dx.doi.org/10.1088/2058-9565/ad35e4.

Quintero, R. A. and Zuluaga, L. F. (2022). Qubo formulations of combinatorial optimization problems for quantum computing devices. In *Encyclopedia of Optimization*, p. 1–13. Springer.

Venturelli, D., Marchand, D., and Rojo, G. (2016). Job shop scheduling solver based on quantum annealing. In *Proc. of ICAPS-16 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS)*, p. 25–34.

Wang, S. and Yu, J. (2010). An effective heuristic for flexible job-shop scheduling problem with maintenance activities. *Computers & Industrial Engineering*, 59(3):436–447.

Xie, J., Gao, L., Peng, K., Li, X., and Li, H. (2019). Review on flexible job shop scheduling. *IET collaborative intelligent manufacturing*, 1(3):67–77.

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br

Xiong, H., Shi, S., Ren, D., and Hu, J. (2022a). A survey of job shop scheduling problem: The types and models. *Comput. Oper. Res.*, 142(C). ISSN 0305-0548. URL https://doi.org/10.1016/j.cor.2022.105731.

Xiong, H., Shi, S., Ren, D., and Hu, J. (2022b). A survey of job shop scheduling problem: The types and models. *Computers & Operations Research*, 142:105731.

Zhang, F., Mei, Y., Nguyen, S., and Zhang, M. (2023). Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. *IEEE Transactions on Evolutionary Computation*.

DOI: 10.59254/sbpo-2024-193625
https://proceedings.science/p/193625?lang=pt-br