Scheduling with Multiple Dispatch Rules : A Quantum Computing Approach

Poojith U Rao and Balwinder Sodhi

Indian Institute of Technology, Ropar {poojith.19csz,sodhi}@iitrpr.ac.in

Abstract. Updating the set of Multiple Dispatch Rules (MDRs) for scheduling of machines in a Flexible Manufacturing System (FMS) is computationally intensive. It becomes a major bottleneck when these rules have to be updated in real-time in response to changes in the manufacturing environment. Machine Learning (ML) based solutions for this problem are considered to be state-of-the-art. However, their accuracy and correctness depend on the availability of high-quality training data. To address the shortcomings of the ML-based approaches, we propose a novel Quadratic Unconstrained Binary Optimization (QUBO) formulation for the MDR scheduling problem. A novel aspect of our formulation is that it can be efficiently solved on a quantum annealer. We solve the proposed formulation on a production quantum annealer from D-Wave and compare the results with single dispatch rule based baseline model.

Keywords: Quantum Annealing · Multiple Dispatch Rules · Flexible Manufacturing System.

1 Introduction

Scheduling in manufacturing systems refers to the process of allocating a common set of production resources such as machines to different jobs simultaneously. The aim of a scheduling problem is to allocate the resources such that an overall objective is optimized. Some of the common objectives are optimizing makespan, total tardiness, mean lateness, mean flow-time and so on. The Job Shop Scheduling Problem (JSSP) aims to calculate the start time of each operation on machines in order to optimize the overall objective. This problem is constrained such that each operation of a job is scheduled only once and when all preceding operations in the job are completed. Here each machine can process only one job at a time [11]. Flexible JSSP (F-JSSP) is an extension of JSSP where each operation can be scheduled on any machine from a given set of machines[2]. JSSP is proven to be an NP-hard problem [3] and F-JSSP, which is its extension, is also NP-hard making it a very difficult combinatorial optimization problem.

Occurrences of unforeseen events in the job shop environment may void the applicability of a scheduling decision. Mitigating the effects of such events necessitates real-time corrective action. Real-time Scheduling (RTS) of operations is a prominent approach to address this issue. The RTS in manufacturing systems has to take decisions in real-time based on the changes in the job shop environment. One category of such decisions is about allotting different Dispatch Rules (DRs) to machines so that the overall scheduling objective is satisfied. In a typical manufacturing shop scenario, the time taken for an operation on a machine is in the order of minutes. Any RTS approach which can decide the DRs within those few minutes is generally desirable.

1.1 Motivation

The main challenge with the approaches that aim to find a (global) optimal job schedule is that they are inefficient and often impractical for real-time scheduling. This is because the job schedule needs to be recomputed quickly in response to changes in the job shop environment. That is, a machine in a shop will not be able to deploy the next job until the optimal schedule is recomputed. Since a practical shop has to handle a large number of jobs and machines, such approaches do not scale well for real-time scheduling in such an environment.

A DR based approach eases the requirement for immediate re-computation of the job schedules for the machines, whenever there is a change in the environment. In the case of DR-based scheduling, each machine on the shop is assigned a DR which the machine will use to select the next job in real-time. The machine can continue to use a given DR for job scheduling, while a "higher-layer" computes an updated DR that reflects the new reality of the shop's environment. Since the DRs immediately react to dynamic events in the environment, they achieve the best time efficiency. Though the dynamic DRs may fail to guarantee a global minimum, they are good for RTS because of zero delay in job scheduling in the machine. This has motivated us, like many other researchers, to apply different techniques to find the optimal set of DRs to drive real-time job scheduling in machine shops.

There are various approaches available today that can perform the RTS decisions of dispatch rules. Predominant among them is based on machine learning methods[11] because they provide approximate optimal answers within a quick response time. However, the downside of such methods is the significant amount of time and computing resources they consume to build the predictive models that they rely on. Secondly, they also require a sufficiently large volume of relevant training data, which may not always be available.

To address these problems, we propose a quantum algorithmic approach. Quantum algorithms are known to offer exponential speedup over their classical counterparts for specific problems. With the availability of systems such as D-Wave's Quantum annealer, the quantum annealing approach has become a promising option for solving complex combinatorial optimization problems. In order to take advantage of such quantum platforms, we have developed a novel quantum formulation for the F-JSSP scheduling *with dispatch rules.*

2 Related work

JSSP has been attempted on a quantum annealer by [8]. They show how the problem can be split into small optimization problems such that it can be solved on a limited quantum hardware capacity. Similarly, [2] explore how a parallel flexible JSSP can be solved on a quantum annealer. They formulate the problem as a QUBO problem along with various variable pruning techniques to solve bigger scheduling problems on quantum annealers. These works indicate the quantum annealer approach is indeed promising for solving the scheduling problems.

Apart from quantum algorithms, a sizeable body of classical work is available in the literature where different algorithms and heuristics are applied for finding the optimal dispatch rules. The new job insertion problem in dynamic flexible job shop scheduling problem has also been addressed in the literature [9] [14] [7]. [9] develop a deep Q-learning method with double DQN and soft weight update to tackle this problem and to learn the most suitable action (e.g., dispatch rule) at each rescheduling point. [7] propose a new Karmarkar-Karp heuristic and combine it with a genetic algorithm for solving the dynamic JSSP. They show that their proposed methodologies generate excellent results. [14] propose a reinforcement learning-based MDRs selection mechanism to tackle this problem. They determine the system state by a two-level self-organizing map and use the Q-learning algorithm as a reinforcement learning agent. [6] take a random-forest based approach for learning the dispatch rules to minimize the total tardiness. They compare their method with other decision-tree-based algorithms and show their approach is effective in terms of extracting scheduling insights.

3 Solution architecture

The overall architecture for the proposed quantum formulation is shown in Figure 1. The input to the FMS is job requests and the DRs for each machine. The FMS outputs the finished products by scheduling the machines using the assigned dispatch rules. When determining the scheduling decisions, our approach considers the jobs received during fixed time intervals T (e.g., every 60 minutes). T is treated as a hyper-parameter in our approach, which can be estimated using suitable heuristics (a naive one would be just to fix the value of T to the average job completion time).

For the jobs arriving during every T interval, the DR estimator calculates the best possible DRs. The mathematical models for estimating DRs are developed in the following section.

4 Problem Formulation

A job shop has a set of n jobs $J = \{j_1, j_2, ..., j_n\}$ which need to be executed on k machines $M = \{m_1, m_2, ..., m_k\}$. Each job j_i has a sequence of operations $O_i = \{o_{i,1}, o_{i,2}, ...\}$ which need to be executed in a specific order. The objective is to assign a dispatch rule $r \in R$ for each machine such that overall tardiness is minimized. The DR assignment problem is framed as a QUBO problem such that we may use quantum annealing to solve it.

4.1 Example scenario used for testing and describing our approach

We use a modification of the example FMS used by a well-known work by Montazeri to describe and test our approach [10]. It consists of 3 machine families (F1, F2, F3), three load/unload stations (L1, L2, L3) and a sufficient Work in Process (WiP) buffer. Machine family F1 and F2 consist of 2 machines each (F11, F12 and F21, F22), and we assume zero loading time for material carriers (e.g., conveyor belt). Each machine in the manufacturing system is connected by a conveyor belt. The belt can carry a maximum of 3 products at any given time. The time taken by the conveyor to transport intermediate products between the machines is shown in the Figure 1. The conveyor belt is modelled as a machine family C with three machines C1,C2,C3. As it has 3 machines, a maximum of 3 products can be carried at anytime. The processing time for these machines is equal to the time needed for the product to be transported on the conveyor belt. There are five different products that need to be manufactured by the FMS. The routing and timing for each of them are given in Table 1. The DRs considered in this work are SIO,SPT,SRPT,SDT,SMT,SIO,LPT,LRPT,LDT,LMT. The description of these rules can be found in Table 2. We would like to note that our approach works in general scenarios similar to the above.

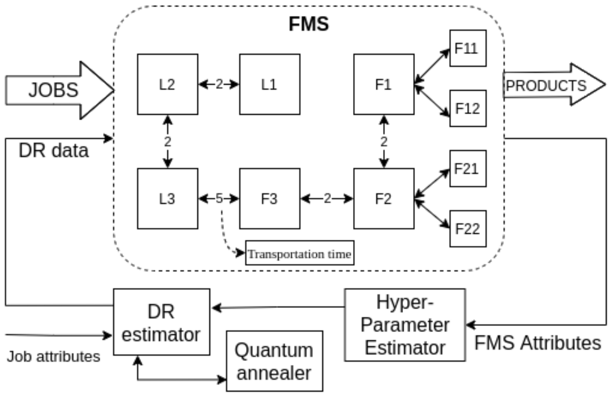

Fig. 1: Proposed solution architecture

.

4.2 Proposed QUBO formulation for the problem

In the following subsections, we describe the crucial elements of the proposed QUBO formulation for the F-JSSP problem that also takes into consideration the dispatch rules in an RTS scenario.

Job	Operations	Timing (minutes)
P1	L2, F2, L3, F1, L3, F2, L2	2, 11, 10, 20, 3, 14, 2
P2	L2, F2, L3, F1, L3, F2, L2	2, 10, 10, 24, 3, 10, 2
P3	L2, F2, L2, F1, L3, F2, L3	2, 15, 2, 30, 10, 21, 3
P4	L2, F2, L2, F1, L3, F2, L3	2, 12, 2, 26, 10, 13, 3
P5	L2, F2, L3, F1, L3, F1, L3	8, 16, 5, 25, 5, 22, 10

Table 1: Routing and timing information of products / jobs

Table 2: Description of different dispatch rules

DR	Description
SIO	Select the job with the shortest imminent operation time
SPT	Select the job with the shortest processing time
SRPT	Select the job with the shortest remaining processing time
SDT	Select the job with the smallest ratio obtained by dividing the processing time of the imminent operation by the total processing time
SMT	Select the job with the smallest value obtained by multiplying the processing time of the imminent operation by the total processing time
LIO	Select the job with the largest imminent operation time
LPT	Select the job with the largest processing time
LRPT	Select the job with the largest remaining processing time
LDT	Select the job with the largest ratio obtained by dividing the processing time of the imminent operation by the total processing time
LMT	Select the job with the largest value obtained by multiplying the processing time of the imminent operation by the total processing time

Decision variables Two sets of binary decision variables are used for representing different decisions needed for solving the problem. The first set of binary variables $x_{j,o,m,t}$ represent the decision whether the o^{th} operation of j^{th} job must be scheduled on a machine m at time t. The second set of binary variables $x_{m,r}$ represent if the dispatch rule r is assigned for machine m.

$$x_{j,o,m,t} = \begin{cases} 1 & \text{if the operation o of job j must be scheduled on machine m at time t} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$x_{m,r} = \begin{cases} 1 & \text{if machine m must be assigned rule r} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The constraints and objective functions that we formulate using these decisions variables will be expressed as Hamiltonians in the following subsections [13]. Further, to understand the background of choosing the specific terms in an objective function or constraint, please refer to the literature on QUBO formulation [4].

Operation scheduling A job comprises of a sequence of operations needed to be performed in order to manufacture a product (see example in Table 1). A product requires each operation to be performed (and thus scheduled) only once. This constraint is expressed as the Hamiltonian in Equation 3

$$H_1 = \sum_{j \in J} \sum_{o \in O} \left[\sum_{m,t} x_{j,o,m,t} - 1 \right]^2 \tag{3}$$

Preference order Every job has a predefined order in which the operations need to be carried out. In our example from Table 1, the product P1 needs the operations to be scheduled in this order: L2 → F2 → L3 → F1 → L3 → F2 → L2. If this order is violated, the schedule generated will not be a valid one. In order to force the scheduler to follow the defined order, a penalty is added for each violation.

Let $x_1 = x_{j,o_1,m_1,t_1}$ and $x_2 = x_{j,o_2,m_2,t_2}$ represent scheduling decisions for two consecutive operations of the same job j at time t_1 and t_2. Let p represent the processing time of operation o_1. In a valid schedule, the operation o_1 must be scheduled before o_2. o_1 and o_2 need to be scheduled (i.e.,for them $x_1 = 1$ and $x_2 = 1$) such that t_2 must be greater than $t_1 + p$ (i.e., o_2 can be scheduled only after o_1 is completed). All combinations of x_1 and x_2 that do not satisfy the condition are violations and must be penalized. The set A in Equation 4 contains all such combinations and the penalty term is shown in Equation 5.

$$A = [(x_{j,o,m,t_1}, x_{j,o+1,m,t_2}) \forall t_2 < t_1 + p\ ] \tag{4}$$

$$H_2 = \sum_{x_1,x_2 \in A} x_1 x_2 \tag{5}$$

Machine scheduling Each machine can handle only one operation at a time. Schedules resulting in overlapping operations on a machine must be avoided. To avoid all such conflicting (i.e., those having overlapped schedule) operations a penalty term is added.

Let $x_1 = x_{j_1,o_1,m,t_1}$ and $x_2 = x_{j_2,o_2,m,t_2}$ represent scheduling decisions for two operations on the same machine m at time t_1 and t_2 respectively. Let p be the processing time of operation o_1. In a valid schedule, the operations o_1 and o_2 must be scheduled such that they do not overlap each other. If both o_1 and o_2 need to be scheduled i.e. $x_1 = 1$ and $x_2 = 1$, then t_2 must not lie in the range t_1 to $t_1 + p$. All combinations of x_1 and x_2 that do not satisfy the condition are violations and must be penalized. The set B in Equation 6 contains all such combinations and the penalty term is shown in Equation 7.

$$B = [(x_{j_1,o_1,m,t_1}, x_{j_2,o_2,m,t_2}) \forall t1 \leq t_2 \leq t_1 + p\ ] \tag{6}$$

$$H_3 = \sum_{x_1,x_2 \in B} x_1 x_2 \tag{7}$$

Squeezed scheduling In the case of DR based scheduling, the DRs select the operation to be scheduled for each machine. When the current operation is completed, the scheduler picks the next operation from the queue based on the DR and schedules it immediately. That is, when an operation O is scheduled at time t on machine M, then the next operation is scheduled at $t+p+1$ if at all available where p is the processing time of the operation O.

To encourage the solutions that have operations scheduled one after the other whenever possible, a reward of -1 is given when the time gap between two operations scheduled on the same machine is zero. Let $C_{m,t'}$ represent the set of all operations which can be scheduled on machine m at time t', then the Equation 9 assigns the above-mentioned reward.

$$C_{m,t'} = [x_{j,o,m,t'} \text{ Where, } t' = t + p + 1] \tag{8}$$

$$H_4 = \sum_{j,o,m,t} \left[\sum_{y \in C_{m,t'}} -x_{j,o,m,t} \times y \right] \tag{9}$$

Tardiness minimization The objective we have considered in our problem formulation is to minimize the total tardiness. In order to accomplish this, the penalty function we have considered is given in Equation 10. It is formulated as a sum of the product of decision variables denoting the last operation of each job and the delay incurred for that particular job. The delay is calculated as the difference between the job completion time (i.e., scheduled time + processing time p) and the deadline d.

$$H_5 = \sum_{j,t} x_{j,o,m,t} \times \max(0, (t + p - d)) \tag{10}$$

Rule assignment The objective is to assign each machine a DR such that the overall cost function is optimized. This implies that each machine must be assigned a single DR. This constraint is expressed as :

$$H_6 = \sum_{m \in M} \left[\sum_{r \in R} x_{m,r} - 1 \right]^2 \tag{11}$$

Rule selection Each rule selects a unique next operation from a queue of operations waiting to be scheduled on the machine based on its definition. For a sequence of operations that are scheduled on a machine immediately one after the other, the dispatch rule which imitates the same must be assigned to the machine. At each instance of time when an operation is scheduled on the machine, we assign a weight to all the rules based on the operation it chooses to be scheduled. The rule which would have picked the same operation that was scheduled is given assigned a reward of -1.

If an operation with processing time p is scheduled on machine m at time t, then let $C_{m,t+p+1}$ represent all operations that can be scheduled once the current operation is completed. In other words, $C_{m,t+p+1}$ contains all the operations waiting to be scheduled on the machine from which the DRs have to select the next operation from. We define function $d_{m,t+p+1}(r,x,y)$ to assign the penalty to decision variables. For each operation in $C_{m,t+p+1}$ if a rule r can select the operation o once the current operation is completed, then a reward of -1 is assigned is assigned to the operation. The function $d_{m,t}(r,x,y)$ is defined as in Equation 12. The Hamiltonian that assigns the weights based on the above conditions is given in Equation 13. It can be noted that 13 is not in the QUBO formulation and cannot be solved directly.

$$d_{m,t}(r,x,y) = \begin{cases} -1 & \text{if } y \text{ is the next operation selected from the list of operations in } C_{m,t} \text{ by rule r after operation } x \text{ is completed} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$H_7 = \sum_{j,o,m,t} x_{j,o,m,t} \times \left[\sum_{r\in R} x_{m,r}\left[\sum_{y\in C_{m,t+p+1}} y d_{m,t+p+1}(r, x_{j,o,m,t}, y)\right]\right] \tag{13}$$

We solve the energy functions in two sequential phases. In the first phase, we calculate the best-squeezed schedule using the energy function H_{sch} described in Equation 14. This gives us the optimal schedule that can be simulated using dispatch rules.

$$H_{sch} = H_1 + H_2 + H_3 + H_4 + H_5 \tag{14}$$

The schedule calculated is used to find the best set of dispatch rules using the Hamiltonian described in Section 4.2. With the schedule known, it becomes possible to accurately find the waiting set of operations in each machine at any given time i.e. the $C_{m,t}$ can be accurately calculated. It represents the operations whose preceding operation was completed and are yet to be scheduled at time t

on machine m. The schedule returned from the first phase is used to replace the variables $x_{j,o,m,t}$ and y in the Equation 13. The term H_7, thus becomes:

$$\hat{H}_7 = \sum_{j,o,m,t} S[x_{j,o,m,t}] \times \left[\sum_{r \in R} x_{m,r} \left[\sum_{y \in C_{m,t+p+1}} S[y] d_{m,t+p+1}(r, x_{j,o,m,t}, y) \right] \right] \tag{15}$$

As the equation now is in the quadratic form, the best set of DRs that produce a schedule similar to S is evaluated by solving the energy function in Equation 16.

$$H_r = H_6 + \hat{H}_7 \tag{16}$$

4.3 Variable pruning

The complexity of solving the problem increases with the number of variables. Reducing the number of variables reduces the time needed to solve the problem and helps the model find a solution near the global minimum. The maximum number of variables needed are $|J| \times |O| \times |M| \times |T| + |M| \times |R|$. We apply three heuristics for reducing the number of variables, as described next:

Ignoring operation-incompatible machines An operation can be scheduled only on machines that can process them. For example, the operation $F1$ of job $J1$ can only be scheduled on machines that belong to the machine family $F1$: $F11$ *and* $F12$. Removing all incompatible machine-operation decision variables significantly reduces the number of redundant variables.

Suitable value for operation time-span Timespan T is a major factor that hugely varies the total number of decision variables. Choosing a large T increases the complexity of the problem as the model will have a large number of decision variables to choose from for each operation. On the other hand, reducing T's value can make the problem unsolvable because performing all scheduled operations requires a certain minimum duration. Thus, a balanced value of T is chosen heuristically.

Time-span bounds for an operation The time when an operation can be scheduled has certain lower and upper bounds depending on its position in the job's operation sequence. The decision variables that lead to those schedules that violate the operation's time span bounds can be safely removed. For example, the time span during which an operation can be scheduled depends on the processing time needed by the operations preceding and following it. Thus, the earliest time at which an operation can be scheduled is the sum of processing times of all its preceding operations. Similarly, the latest time before which the operation must be scheduled is the sum of its processing time and that of the operations following it.

5 Experiments and results

We performed several experiments to validate the correctness and efficacy of our approach. The important ones, along with their results, are described below.

Table 3: Makespan of different MDRs

Jobs	Products	DL	Tardiness										
	P1-P2-P3-P4-P5		QUANT	SIO	SPT	SRPT	SDT	SMT	LIO	LPT	LRPT	LDT	LMT
J1	2 - 2 - 2 - 2 - 2	200	**42**	61	75	90	94	90	124	256	170	137	199
J2	10 - 0 - 0 - 0 - 0	200	**40**	45	**40**	45	45	45	126	**40**	126	126	126
J3	0 - 10 - 0 - 0 - 0	200	**22**	37	23	37	37	37	41	23	38	41	41
J4	0 - 0 - 10 - 0 - 0	250	**9**	15	**9**	15	15	15	13	**9**	13	13	13
J5	0 - 0 - 0 - 10 - 0	220	**13**	33	**13**	33	33	33	**13**	**13**	19	**13**	**13**
J6	5 - 5 - 0 - 0 - 0	200	**35**	**35**	37	37	39	42	76	**35**	90	76	99
J7	5 - 0 - 0 - 5 - 0	200	**6**	11	26	26	45	26	15	130	98	19	183

Table 4: Energy vs Tardiness of various dispatch Rules

Rule	Min. Tardiness	Min. Energy
QUANT	**42**	**-658.16**
SIO	61	-656.56
SPT	75	-655.32
SRPT	90	-654.13
SDT	94	-652.7
SMT	90	-654.12
LIO	124	-650.61
LPT	256	-636.35
LRPT	170	-647
LDT	137	-649.87
LMT	199	-643.52

5.1 Tardiness comparison of the proposed approach with Single Dispatch Rules (SDR)

For validating the proposed approach, we compared the tardiness of different job groups on the FMS mentioned in the case study (see Table 1). Each job group consists of 10 random jobs and a deadline for completion. All jobs are submitted to the FMS at time $t = 0$, and the tardiness is calculated for the schedules generated by the proposed approach and those generated by SDR approach. The results are shown in the Table 3.

The solution to the proposed approach was solved on a real quantum annealer provided by D-Wave. Please note that the performance of these devices is affected by ambient noise. The answers calculated by these devices are only near-optimal solutions. Thus, the solution obtained for our problem formulation is not the best schedule but a sample near it. The numbers in Table 3 indicates that just within 2-3 minutes, the annealer is able to calculate the set of dispatch rules.

5.2 Validating the correctness of overall energy function

The solution to the scheduling problem is the combination of decision variables having minimum value to the energy function in Equation 14. We‘ calculate the energy value of the solution produced by different SDRs and compare it with the value obtained by using the proposed method. Our results in Table 4 show that the overall energy values are positively correlated with the tardiness. That is, the schedule with the least tardiness has the least energy value.

In order to verify the correctness of the Hamiltonian H_r (Equation 16) used for finding the best dispatch rules, we initialized all the machines with pre-determined dispatch rules. We first calculated the schedule generated by these rules by simulating the machines in the job shop environment. Later, we calculated back the best set of dispatch rules for this schedule using the Hamiltonian H_r. As expected, the pre-determined set of rules from which the schedule was generated had the minimum energy value. This validates the correctness of our hypothesis and approach.

6 Discussion

The number of binary variables in the formulation increase with the number of machines, jobs, operations and the time-span parameters. As an example for a 15x15x15 (Jobs $\times$ Operations $\times$ Machines) instance with time span of 250, the number of binary variables is around thirty five thousand. On the contrary the largest quantum annealer today has only 5000 qubits with 15 way connectivity. Ultimately, the linear coefficients in QUBO are mapped to qubit biases and quadratic coefficients to coupler strengths. With limited connectivity of qubits, QUBO formulations having large number of quadratic terms need much larger number of qubits to embed the problem on the annealer. Hence large problems such as ours cannot be directly solved on these devices without classical heuristics directly on annealers.

In this work we tested our QUBO formulation using the Leap's hybrid solvers for finding the best set of dispatch rules. These solvers implement state-of-the-art classical algorithms such as Tabu search together with intelligent allocation of the quantum computer to parts of the problem where it benefits most. While this enables them to accommodate very large problems, it also reduces their ability to find near optimal solutions.

We solved the above QUBO problem on public datasets [12] [15] available for 6x6x6, 8x8x8, 10x10x10 and 15x15x15 scenarios using Leap's hybrid solver and compared it with the baseline models as described above. The deadline for each job was set to 1.2 times the optimal makespan mentioned in the dataset for each job. The *time_ limit* parameter of the hybrid solver was kept at a large value such that no further improvement in the solution quality was observed on further increment. The results obtained showed that the Leap's hybrid solver was not able to sample solutions with minimum energy even with multiple attempts. The baseline models and the Leap hybrid solver found better solutions in equal number of cases. But in all cases the the QUBO formulation was able to determine the best solution accurately based on the energy value.

We observed that the energy value of the annealer solution (non optimal) was in some case around 5-10 percent higher than the energy value of the best answer known to us (through baseline models). This showed us two things 1) Leap's hybrid solver cannot sample solutions nearer to the true optimal for our problem formulation. This is also consistent with their claim on the website [1] 2) The QUBO formulation is able to detect the best answer in each case.

As the constraints in the problem are encoded as penalties, the coefficients of each term determine the energy characteristics of the total Hamiltonian. In order to ensure the constraints are always satisfied, the coefficients for the penalty terms are kept high. As the solutions returned by leap is sub optimal, the gap (ratio or difference) between the coefficient values of penalties and objective function plays a significant role. When it is small, the solutions has many constraint violations and in case the gap is high, the objective function value is not satisfactorily optimized.

Instead of formulating such complex Hamiltonians with many constraints another approach is to use more sophisticated heuristic algorithms such as Quantum Alternating Operator Ansatz (QAOA). QAOA is a quantum classical hybrid meta-heuristic framework proposed by [5] for performing approximate optimization on gate-based quantum computers. It is well suited for scenarios such as JSSP where the feasible solution space is much smaller, such as optimization where solutions have to satisfy multiple hard constraints. But a state-of-art gate-based quantum computer currently available can support less than 100 qubits.

7 Conclusion and future work

Finding the best set of Multiple Dispatch Rules (MDRs) needed for real-time scheduling in a flexible manufacturing system scenario is an important scheduling problem. We have formulated it as a Quantum Unconstrained Binary Optimization (QUBO) problem and solved it on a quantum annealer. The results obtained from quantum annealer have the least tardiness compared with the schedules produced by Single Dispatch Rules (SDRs). The strong correlation between energy values of different schedules with the tardiness validates that the proposed method can find MDRs that produce schedules with minimum tardiness.

The annealers provided by D-Wave currently cannot obtain the global minimum to the energy functions. This puts a limit on the quality of solutions obtained for our formulations. The quality and the time needed to calculate solutions will improve with advancements in quantum technology.

Future research should focus on how the FMS attributes can be used to find good hyperparameter settings such as length of time span. To further check the robustness of the approach, the following types of experiments can be conducted: 1) Finding the overall tardiness value when large job groups are split into small subgroups and solved sequentially. 2) Checking performance of the approach when the length of time span is varied and 3) Comparison of the quantum method with other classical methods. 4) Performance when run directly on quantum hardware.

References

1. D-Wave: https://docs.dwavesys.com/docs/latest/handbook_hybrid.html
2. Denkena, B., Schinkel, F., Pirnay, J., Wilmsmeier, S.: Quantum algorithms for process parallel flexible job shop scheduling. CIRP Journal of Manufacturing Science and Technology **33**, 100–114 (2021)
3. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of operations research **1**(2), 117–129 (1976)
4. Glover, F., Kochenberger, G., Du, Y.: A tutorial on formulating and using qubo models (2019)
5. Hadfield, S., Wang, Z., O'Gorman, B., Rieffel, E.G., Venturelli, D., Biswas, R.: From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. Algorithms **12**(2), 34 (2019)
6. Jun, S., Lee, S., Chun, H.: Learning dispatching rules using random forest in flexible job shop scheduling problems. International Journal of Production Research **57**(10), 3290–3310 (2019)
7. Kundakcı, N., Kulak, O.: Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. Computers & Industrial Engineering **96**, 31–51 (2016)
8. Kurowski, K., Weglarz, J., Subocz, M., Różycki, R., Waligóra, G.: Hybrid quantum annealing heuristic method for solving job shop scheduling problem. In: International Conference on Computational Science. pp. 502–515. Springer (2020)
9. Luo, S.: Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Applied Soft Computing **91**, 106208 (2020)
10. Montazeri, M., Van Wassenhove, L.: Analysis of scheduling rules for an fms. The International Journal of Production Research **28**(4), 785–802 (1990)
11. Priore, P., Gomez, A., Pino, R., Rosillo, R.: Dynamic scheduling of manufacturing systems using machine learning: An updated review. Ai Edam **28**(1), 83–97 (2014)
12. Samsonov, V., Kemmerling, M., Paegert, M., Lütticke, D., Sauermann, F., Gützlaff, A., Schuh, G., Meisen, T.: Manufacturing control in job shop environments with reinforcement learning. In: ICAART (2). pp. 589–597 (2021)
13. Santoro, G.E., Tosatti, E.: Optimization using quantum mechanics: quantum annealing through adiabatic evolution. Journal of Physics A: Mathematical and General **39**(36), R393 (2006)

14. Shiue, Y.R., Lee, K.C., Su, C.T.: Real-time scheduling for a smart factory using a reinforcement learning approach. Computers & Industrial Engineering **125**, 604–614 (2018)
15. Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P.S., Chi, X.: Learning to dispatch for job shop scheduling via deep reinforcement learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1621–1632. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/11958dfee29b6709f48a9ba0387a2431-Paper.pdf