REVISTA INVESTIGACION OPERACIONAL VOL. xx , NO.x, xxx-xxx, 201x
FORTHCOMING PAPER ·#68-20-18-02

OPTIMIZATION OF HEAVILY CONSTRAINED HYBRID FLEXIBLE FLOWSHOP PROBLEMS USING A MULTI-AGENT REINFORCEMENT LEARNING APPROACH

Yunior César Fonseca-Reyna[*1], Yailen Martínez-Jiménez[**], Alberto Verdecia Cabrera[*], EdelAngel Rodríguez Sánchez[***]
[*]Universidad de Granma, Bayamo, Granma, Cuba
[**]Universidad Central de las Villas, Santa Clara, Villa Clara, Cuba
[***]Universidad Técnica de Cotopaxi, Cotopaxi, Ecuador

ABSTRACT
This paper presentsan adaptation of the Reinforcement Learningapproach known as Q-Learning to solve a scheduling problem that comes from industry. The problem studied in this paper is knownas the Hybrid-FlexibleFlow Shop Schedulingwhere a variety of constraints are taken into account. These include precedence constraints, sequence dependent setup times(which can be anticipatory and non-anticipatory) along with machine lags, machine eligibility and release times. This problem mixes the features of regular FlowShop and parallel machine problems by considering stages with several unrelated parallel machines, where stage skipping might occur, i.e., not all stages must be visited by all the jobs. This version has been proved strongly NP-hard and the objective is to determine a schedule that minimizes the maximum completion time (makespanor C_{max}). The effectiveness of the proposed algorithm is empirically evaluated throughseveral standard benchmarks problems and the solutions arecompared against other high performing existing algorithm. The results shown that the proposed algorithm is very competitive for the studied problem.

KEYWORDS: Hybrid-flexible flow-shop, makespan; optimization; scheduling, q-learning.

MSC: 68T20, 68T05, 90C59

RESUMEN
El artículo presenta la adaptación de un enfoque del Aprendizaje Reforzado nombrado Q-Learning para la solución de un problema de la práctica industrial. El problema tratado se conoce como Variante de Flujo Regular Híbrido-Flexible en el cual se tiene en consideración un conjunto de restricciones presentes en entornos reales de la producción. Este incluye restricciones de precedencia entre trabajos, tiempos de configuración dependientes de la secuencia los cuales pueden ser anticipativos o no, máquinas elegibles y tiempos de liberación. Este problema es una generalización de la Variante de Flujo Regular donde se consideran varias etapas con máquinas no relacionadas en paralelo en el que pudiera ocurrir que un trabajo salte una o varias de ellas. Este problema es *NP-Hard* y el objetivo es determinar una secuencia de trabajos que minimice el tiempo total de procesamiento. En la presente investigación se evalúa la efectividad del algoritmo propuesto a través de un conjunto de instancias problemas de este tipo y las soluciones son comparadas conlas obtenidas por otro algoritmo de alto rendimientopropuesto por la literatura especializada. Los resultados obtenidos muestran que el algoritmo propuesto alcanza soluciones de buena calidad mostrando seraltamente competitivo.

1. INTRODUCTION

Scheduling is an attractive and one of the most important decision making process in the operation of manufacturing systems. It is not only a theoretical field of study but also an interesting field of application in industry and in many other real-world situations[13]. Basically, manufacturing scheduling decides the sequence of jobs andthe allocation of resources. As manufacturingsystems become more complex, conventional methodsfor scheduling become ineffective both in terms of computational time and resources. Many authors have recognized a gap between the literature and the industrial problems[30, 23, 36, 3].Most of the researches concentrate on optimization problems that are actually a very simplified version of reality. Thisallows the use of sophisticated approaches which guarantee, in many cases,optimal solutions. What the industry needs are systems for optimized production scheduling that adjust exactly to the conditions in the production plant and that generates good solutions in very little time.

[1]fonseca@udg.co.cu

In this paper, we focus on manufacturing scheduling where all jobs share the same route, specifically the Hybrid-Flexible Flow Shop Scheduling (HFFS) whichhas been extensively studied due to its application in industry.A series of restrictions are considered that include the possibility to skip stages,non-eligible machines, precedence constraints, positive and negative time lagsand sequence dependent setup times. Briefly, this paper studies a very realistic production scheduling problem.Theobjective is to determine a production schedule for all products to be completed in a minimum time (i.e., minimize the makespan).
A Multi-Agent Reinforcement Learning Approach(MARLA) is presented for the described problem with the purpose of obtaining good solutions to the problem in moregeneral cases. In order to evaluate the performance of the proposed algorithm, test cases of the specialized literature are used and the results obtained are compared with the reported optimal resultsby a Learning Automata Approach. The obtained resultsendorse the use of thisproposed method.
This paper is organized as follows. A literature review about scheduling is given in the next section, and the HFFS isintroduced in detail in section 3. Section 4 introduces general ideas when solving HFFS problems using RL. Then, the adaptation of a MARLA is proposed in section5, and the results of the computational experiments are presented in section 6. Section 7 presents an analysis of the obtained results. The final section is devoted to conclusions with the future research directions.

2. RELATED WORK

Although scheduling problems have been widely studied in literature, most of the studies are concentrated on basic scheduling problems like the single machine, the parallel machines, or the permutation flow shop problem. Unfortunately, they only represent a simplified version of most production environments.
As previously stated, the problem considered is a HFFS with a set of real-world constraints, which has attracted considerable attention in recent years. It has been solved using different exact and heuristic techniques as a solution methodology [21, 30, 25]. However, because the NP-Completeness of the problem, metaheuristics tested through simulations studies have been more popular[2, 1, 5, 39].
Since Johnson's pioneering work[12] on the two-machine regular permutation flow shop, a large number of studies have been published about scheduling.The problem considered in this paper has four main characteristics that are jointly considered: flow-line environment, where all the jobs are processed in the same order; hybridsetting, where each stage has parallel machines; flexibility, where stages might be skipped; and a set of real-world constraints. In literature, it is possible to find reviews of each one of these four characteristics[17, 30, 27, 31, 24].
The first reviews about the HFFS appeared in 1999 by Linn and Zhang[17],this work concludes that there is a gap betweentheory and practice and that there is a need of future research inthis direction. After this, more recent researches concentrated on the HFFS problem with real-world constraints due to the importance of the manufacturing productionprocess. In that direction, many approaches have been used to solve this problem.
A case study on the manufacture of a printedcircuit board for communication equipment was presented by Alisantoso et al. [2]. The authors illustrated the feasibilityof using a Genetic Algorithm (GA) to deal with the break-downs of the machines. The approach incorporates an accelerating mechanismas well as a restraining mechanism to assist the searchfor a near optimal solution.
Bertel and Billaut[4] proposed a linear programming formulation, heuristic algorithms based on priority dispatching rules, with new dispatching rules and a GA for an industrial multiprocessor HFFS problem with recirculation. In order to evaluate these heuristics, experiences on instances like industrial ones are computed.The proposed approaches are compared and the GA shows the best efficiency.
Ruiz and Maroto[28]propose the adaptation of a GA, which performed well in regular flowshops in an earlier studypresented in Ruiz et al. [29], to a much more realistic version of the problemwith sequence dependent setup times, unrelated parallel machines at each production stage, and machine eligibility. Such a problem is common in theproduction of textiles and ceramic tiles. After this, in 2008, Ruiz et al.[30]presented a Mixed Integer Programming Mathematical Model (MIP) and some heuristics for the HFFS. In this problem several realistic characteristics arejointly considered. The proposed approaches are tested againsta comprehensive benchmark. This paper contributesto recent research efforts to bridge the gap between the theory and the practice of scheduling.
Jungwattanakit et al. [14] have considered a HFFS problem with unrelated machines in each stage and sequence dependent setup timesfor minimizing a convex combination of makespan and the number of tardy jobs. The authorsproposed a MIP, a GA, a TabuSearch (TS) and a Simulated Annealing(SA) approach.The performance of the heuristics is compared using a set of test problems. They found that among the

constructive algorithms, the insertion-based approach is superior to the others, whereas the proposed SA algorithmsare better than TS and GA among the iterative metaheuristic algorithms.
Naderi et al. [22]study a hybrid flexible flow shop with respect to sequence dependent setups. They propose a dynamic dispatching rule and aniterated local search algorithm. The new algorithms are evaluatedby comparison against seven other high performing algorithms from theliterature. Statistical experiments show that the proposed algorithms are very competitive for the studied problem.
Vallada et al. [37]developed two versions of GAs which include a limited local search procedure for the parallel machine scheduling problem with sequence dependent setup times with the objective to minimize the makespan. The authors also formulated a MIP model for the sameproblem. From the results, the authors concluded that both versions of the proposed genetic algorithm obtain the best results.
In 2012, Gicquel et al [10] proposed an exact solution approachfor a real-world scheduling problem arising from a bioprocess industry. The authors proposeda mixed-integer linear programming formulation based on adiscrete time representation. The obtained results shows that the proposed method provides goodfeasible solutions with a reasonable computation effort.
Kianfar et al. [16]proposed two methods to achieve near optimal solutions to the HFFS in a dynamical environment. The first method is a Dispatching Rule combined with neighborhoodsearch techniques. The second approach is a Hybrid Genetic Algorithm (HGA). The results of simulating different scenarios indicate that scheduling methods proposed in this paper have a significant impact on the shop performance and provide better performance under all of shop conditions.
Wang and Liu in [39] studied the two-stage no-wait HFFS with a single machine on the first stage andmultiple parallel identical machines on the second stage to minimize makespan. The authors proposed a GA that is tested under a four combination of parameters (combinations of different crossover and mutation operators). The results with different problem configurations demonstrated the effectiveness and efficiency of the proposed genetic algorithm. Theresults showed that the four genetic algorithms can reach the solutions with less than 5% mean percentage deviation for most cases.
A HFFS with assembly operations is studied by Fattahi et al. in[6]. The authors proposed a hierarchical Branch and Bound (B&B) algorithm that were extended in order to schedule the parts and assign them to machines in each stage of the HFFS. Some experiments are used to demonstrate the performance of the proposed algorithm.

3. PROBLEM DESCRIPTION

The HFFS problem consists of performing a set of n jobs $J = (J_1, J_{2,\dots,} J_n)$ on a set of stages $M = (1, 2, \dots, m)$, where each stage i contains a set of unrelated parallel machines $M_i = (1, 2, \dots, m_i)$. Flexible means that each job $j \in N$ visits a subset $F_j \subseteq M$ of the stages and skips the remaining ones. The processing time of job j on machine l at stage i is given by p_{ilj}. Each machine can only process one job at a time and each job $i$can only be processed on one machine at any time.Furthermore, in this paper, a number of additional real-world constrains are take into account. These constraints are studied by Ruiz et al. in [30]:

- Precedence constraints: enforce that a job j cannot be processed before all predecessors $P_j$have been completely finished.
- Set of eligible machines (E_{ij}): only some machines of M_i can process job j at stage i. $1 \leq |E_{ij}| \geq m_i$ if job j is processed at stage i and $E_{ij} = 0$ if job j skip stage i.
- Machines are not necessarily available from time zero, but have individual release dates. These release dates prohibit a machine l within stage i to process any job before its release date (rm_{il}).
- Between two consecutive jobs j and k on machine l at stage i, a setup time s_{iljk} needs to be taken into account. The time depends on the machine and on both jobs (hence called sequence dependent setup time). This setup can be either anticipatory or not, depending on the binary parameter A_{iljk}. In the case of an anticipatory setup, the setup can be performed directly when the previous job is completed at the current machine. In the contrary case, when setup is not anticipatory, setup can only be done when the job has arrived at the current machine.
- The time lag for job j between stage$i$ and the next visited stage, is given bylag_{ilj}, where l represents the machine job j is assigned to at stage i. This lag could be negative (overlap) or positive (waiting time).

The objective considered in this paper is to have all jobs completed as early as possible i.e., to minimize the maximum completion time or makespan. It is the most generic objective, and it does not depend on the data on due dates. Makespan is mainly production oriented, assuring efficiency by giving priority to compact schedules. Other objectives, such as tardiness, are not considered here.Therefore, makespan can formally be described as $C_{max} = max_{i\in M, j\in N} C_{ij}$, where C_{ij} denotes the completion time of job j at stage i[35].

Using the three field notation by Vignier et al. [38]and using some extensions of our own, we can define this HFFS problem as:

$$HFFS, \left(\left(RM^{(i)}\right)_{i=1}^{(m)}\right) | M_j, rm, prec, S_{ijk}, A_{iljk}, P_j, E_{ij}, lag | \ C_{max}$$

The HFFS is more complex than other flow shops. Due the possibility of skipping stages, the machine eligibility and precedence constraints among jobs, the number of feasible solutions depends on the instance and might be smaller. However, many simplification of the proposed problem are proved as NP-Hard [11, 26, 32, 7]. The case of Permutational Flow Shop Scheduling, which is the simplest case of scheduling problem is NP-Hard [9, 19, 8]. Therefore,we can conclude that the considered HFFS problem is also NP-Hard.

3. REINFORCEMENT LEARNING FOR THE HFFS

Reinforcement learning (RL), as noted in [15], dates back to the early days of cybernetics and work in statistics, psychology, neuroscience andcomputer science. During the last decades it also attracted increasing interest fromthe machine learning and artificial intelligence communities.

RL is learning what to do (how to map situations to actions) so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics, trial-and-error search and delayed reward, are the two most important distinguishing features of RL[33].

In the standard RL model, an agent is connected to its environment via perception and action, as depicted in Figure 1. In each interaction step, the agent perceives the current state s of its environment, and then selects an action a to change this state. This transition generates a reinforcement signal r, which is received by the agent. The task of the agent is to learn a policy for choosing actions in each state to receive the maximal long-run cumulative reward.Reinforcement Learning methods explore the environment over time to come up with a desired policy [42].

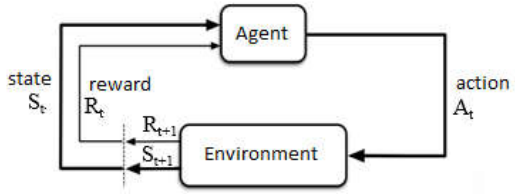

Figure 1. The standard reinforcement learning model

Formally, the basic reinforcement learning model consists of:

- a set of environment states S;
- a set of actions A;
- a set of scalar "rewards" in $\mathbb{R}$.
- a transitionfunction T.

RL provides a flexible approach to the design of intelligent agents in situations for which, for example, planning and supervised learning are impractical. RL can be applied to problems for which significant domain knowledge is either unavailable or costly to obtain[20].

After presenting the basic idea of RL, we present the main concepts that have to be taken into account when solving a scheduling problem using RL. Figure 2 shows an initial idea of the learning environment. Following the standard model presented in Figure 1, this is how we map agents and actions in the environment when solving a scheduling problem. The scheduling environment defines the number of agents in the system and the relations among them. Agents may not know the global state of the system. To achieve better performance of agents or the system, agents communicate with each other to determine their actions based on the limited information.

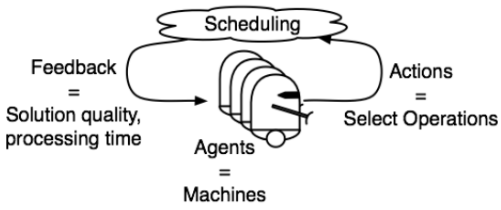

Figure 2. Agents in a scheduling environment

HFFS problems are well suited to be modeled using the idea shown in Figure 2 because the following information is always available at the beginning of the scheduling process:

- Number of machines.
- Number of jobs.
- Processing order of the operations belonging to the different jobs (ordering constraints).
- Processing time of each operation.

The idea presented previously can easily be adapted to the case of HFFS when there are unrelated parallel machines per stages that can execute the same type of task. We analyzed two possibilities in order to adapt the algorithm:

- Queues associated to individual machines, which means that there is one agent per resource, as was the case in the basic approach, and each of these agents will have a queue associated (see Figure 3). The difference is given by the fact that when one operation is fully executed on a machine, the next operation of the corresponding job has to be released for execution, i.e. it should be assigned to a specific agent, meaning that there is an extra decision to take, which was not the case in the previous approach.
- Having m stages of k machines, the queues can be considered to be associated to stages instead of being associated to a single resource. For example, if an operation needs to be processed by a machine of stage 3, then it is placed on the queue of the stage 3 (see Figure 4).
-

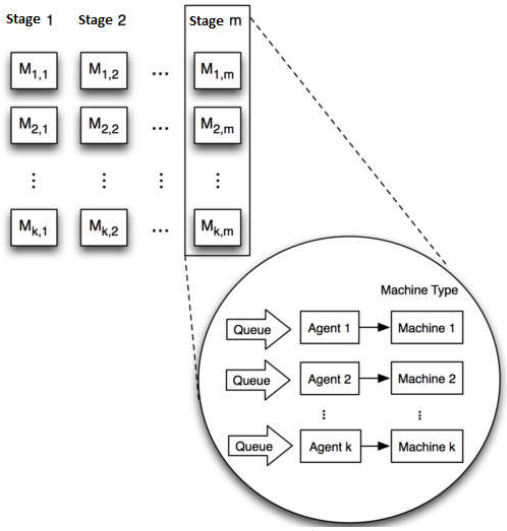

Figure 3. Queue per agent, and one agent per resource

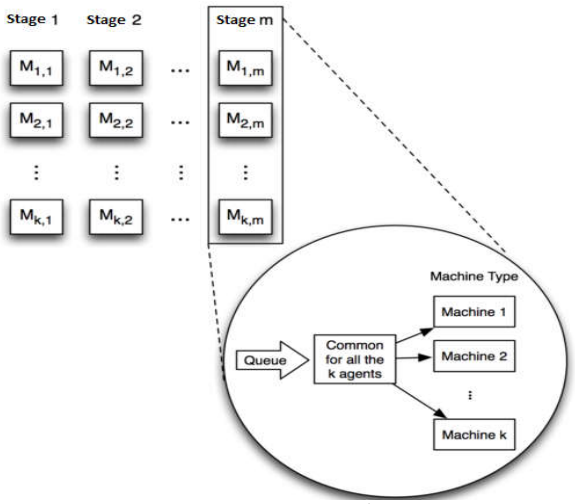

Figure 4. Queue per stage, agent per stage

After analyzing both points of view, we decided to keep the second one, having a queue per stage, as it fits better in the idea depicted in Figure 1.
The main characteristics that have to be considered when modeling the HFFS are adapted and will be summarized in the next section.

4. THE PROPOSED APPROACH

Various RL strategies have been proposed that can be used by agents to develop a policy for maximizing the rewards accumulated over time. One of the early breakthroughs in RL was the development of an algorithm known as Q-Learning (QL), which was proposed by Watkins in [40]. The goal of this algorithm is to learn the state-action pair $Q(s,a)$, which represents the long-term expected reward for each pair of state and action (denoted by s and a, respectively), defined by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))]$$

In this expression $\alpha \in [0, 1]$ is the learning rate and r the reward or penalty resulting from taking action α in state s. The learning rate α determines 'the degree' by which the old value is updated.For example, if the learning rate $\alpha = 0$, then nothing is updated at all. If, on the other hand, $\alpha = 1$, then the old value is replaced by the new estimate. The discount factor (parameter γ) has range value of 0 to 1 ($0 \leq \gamma \leq 1$).
If γ is closer to zero, the agent will tend to consider only immediate reward. If it is closer to one, the agent will consider future reward to be more important.
QL has the advantage that is proven to converge to the optimal policy in Markov Decision Processes (MDP) under some restrictions [41, 34].The optimal state-action values for a system represent the optimal policy that the agent intends to learn. The standard procedure of the QL algorithm is analyzed in the Algorithm 1.

Algorithm 1. Q-Learning Algorithm

1 Initialize Q-values arbitrarily
2 **for** each episode **do**
3 Initialize s
4 **for** each episode step **do**
5 Choose $a$fromsusing policy derived from Q(e.g., $\varepsilon - greedy$)
6 Take action a, observe state s' and r
7 Update Q-value, $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))]$
8 $s \leftarrow s'$
9 **end for**
10 **end for**

Each iteration of steps 2–10 represents a learning cycle,also called an ''episode''. Each episode is equivalent to one training session. In each training session, the agent explores the environment and gets the rewards until

it reaches to goal state. The purpose of the training is to enhance the knowledge of the agent represented by the Q-values. More training will give better values that can be used by the agent to move in more optimal way.The ε-greedy action selection method instructs the agent to follow the current policy π most of the time, but sometimes, to choose an action at random (with equal probability for each possible action a in the current state s). The probability ε determines when to choose a random action; this allows some balance between exploration and exploitation.

4.1. APPLYING QL TO SOLVE THE HFFS

In the real production environment, scheduling problems can be seen critical activities that have repercussions in the efficiency and the effectivity of the manufacturing process. The HFFS consists of set of jobs that flow through a number of production stages. At each of the stages, one of the machines belonging to the stage is visited. A series of restrictions that include the possibility to skip stages, non-eligible machines, precedence constraints, positive and negative time lags and sequence dependent setup times, are considered.
When QL is applied to solve the HFFS, we use one agent per stage(see Figure 4). These agents select the actions they must execute.While an agent is executing an operation, the next agents, which represent the next stages, wait for that operation to finish. For example, if we have three jobs and two stages, the agent associated to the second stage must wait until the first agent executes all the jobs.
There are important elements to be decided, which are summarized asfollow:

States and Actions: There is an agent associated with each stage, and this agent will make decisions about future actions. For an agent to take an action means that it must decide which is going to be the next job to be processed from the set of possible jobs (these are the ones waiting in the queue of the agent of the corresponding stage), and decide which of the k machines in this stage processes the job (taking into account the relationships of precedence, if they exist). When dealing with this problem as an MDP, each agent has access to the information associated with its stage (quantity and availability of the machines) and the operations that are waiting to be processed as well as those that have already been executed. Each agent selects the next operation to be processed using a policy π and decides which machine in the stage will process it by choosing a machine in such a way that the value of makespan is minimal.
Rewards: In the case of the HFFS, the state is represented by the sequence that each agent has been building. An action is then to insert a new job in the sequence that has been built by each agent in each of the stages. Then, considering that the goal is to minimize the makespan, thereward is defined as:$r = 1/makespan$.
Q-values: For each of the m stages (number of agents) there are at most n possible jobs to process, then to store the q-values, a matrix with n rows and m columns is built, where each agent can only modify the row k, which represents the number of machines present in that stage. As jobs can skip stages, and in fact, it may be the case that only visit one of them, the makespan will be determined by the greater of the partial values obtained in each stage.
Algorithm 2 shows the general workflow of the QL algorithm for the HFFS with the goal of minimizing the makespan.

Algorithm 2. Applying QL to solve the HFFS

input: HFFSinstance, number of episodes, learning rate α, discount factor γ, value of ε

output: jobs sequence of minimal makespan$S$

Initialize: Q-values arbitrarily
$Q(s,a) \leftarrow [][]$
$S \leftarrow [][]$
foreach episode **do**
Initialize $s \leftarrow [][]$
while (*existJobsNoProcessed*())
foreachagent **IN** Agents **do**
while(*existPendentOperations*())
Choose $a$fromsusing policy derived from Q(e.g., $\varepsilon - greedy$)
Take action a, observe state s' and r
Update Q-value, $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))]$
$s \leftarrow s'$
end while

14 **endforeach**
15 **end while**
16 **if***makespan*(s)<*makespan*(S)**then**
17 $S \leftarrow s$
18 **endforeach**

5. COMPUTATIONAL EVALUATION

In this section, a presentation and thorough analysis of the results acquired on the Ruiz et al.[2][30]small benchmark suite are provided. In this case, the authors subdivide the instances according to the number of jobs, specifically 5, 7, 9, 11, 13, 15, 50 and 100. For each subset, there are 96 instances with other specific restrictions (e.g. precedence constraints or not, skipping stages or not, etc.). The main restrictions considered in these instances can be analyzed in [30].
The solution quality is mainly measured in terms of the relative increase in makespan with respect to the best known solutions. Also, the performance of the proposed algorithm iscompared with the results reported by a method based in Learning Automata (AL) proposed by Bert et al. in[3].

5.1. CALIBRATION

Typically, optimization algorithms are measured using two criteria: accuracy (i.e., solution quality) and speed. In our implementation, a total of five parameters are required to set up the proposed algorithm: number of episodes, learning rate α, discount factorγ, action selection mechanism ($\varepsilon - greedy$ in our case) and the maximum computational time allowed t_{max}.
In this section we study the behavior of the proposed approach using different combinations of parameters. All the different combinations of the afore mentioned parameters can be seen as alternative QL algorithm. In order to calibrate the algorithm some initial experiments were performed to analyze the learning process under the effect ofdifferent parameters values. Typical combinations for the QL algorithm are the following [18]:

- C_1: episodes = $n * m$, α= 0.1, γ= 0.8, ε= 0.2
- C_2: episodes = $n * m$, α= 0.1, γ= 0.9, ε= 0.1
- C_3: episodes = $n * m$, α= 0.1, γ= 0.8, ε= 0.1
- C_4: episodes = $n * m$, α= 0.1, γ= 0.9, ε= 0.2

The algorithm is tested using all these combinations, with thementioned set of HFFS instances randomly selected. This set comprises 60 instances (10 per subset of 5, 7, 9, 11, 13 and15jobs). We executed 10 runs every instance.
Also, the stopping criterion used is the CPU time limit (t_{max}) fixed to $n \cdot (m/10)$ seconds. This stopping criterion gives more time when the number of jobs or machines increases. All the experiments were carried out on a Pentium IV CPU (running at 2.4 GHz) and 2GB of RAM.
The response variable of the experiment is then calculated with the following expression:

$$Relative\ Percentage\ Deviation\ (RPD) = \sum_{i=1}^{10} \left[\frac{MK_i - UB_i}{UB_i} * 100 \right] / 10$$

where MK is the solution obtained by the proposed algorithm on a given instance out of the 60 and UB is the lower bound for the HFFS reported in [30] for that specific instance.
Table1 shows the QL performance under the mentioned parameters combinations to the selected problem.

Table 1. Q-Learning Performance under four parameters combinations.

	RPD (%)						
Combination/Instances set	5	7	9	11	13	15	AVERAGE
C_1	0.0332	1.4522	1.9003	1.4718	3.6932	7.7761	2.0409

[2]Available at http://soa.iti.es/problem-instances

C_2	0.0297	1.4211	1.9221	1.4611	3.6813	7.7463	2.0327
C_3	0.0351	1.4902	1.9332	1.4642	3.6799	7.7564	2.0449
C_4	0.0312	1.4413	1.9320	1.4673	3.6825	7.7861	2.0426

After executing the experiments, as shown in Table 1, $C_2$is identified as best combination among the four, reporting the lowest value of RPD (this value is colored in the table with light gray). However, we also performed a nonparametric test to determine and verify the best combination using the results from these executions. Then, we applied the Friedman test to determine if there are significant differences and a Holm's test to do a multiple comparison. The results obtained show that C_2 is the best combination. In this case, we decided to keep the second combination for all the experiments:episodes $=n * m$,α= 0.1,γ= 0.9, ε= 0.1.

5.2. COMPUTATIONAL RESULTS

For the evaluation, we use the benchmark set previously mentioned, proposed inRuiz et al. [30]. Taking into account that set of instances has 96 problems, we have 576instances. In order to compare the results obtained, the RPD is calculated using C_2 for each instance.

Table 2 summarize the results using the following structure. Each columns represents an instance sets, according to the number ofjobs. The top row shows the mean relative deviations (in percentages) w.r.t. the best known values. Next, the best relative deviations for each set are given (also in percentages). Negative values are improvements over the best known results. The three bottom rows denote the number of instances where we got improvements, the same, and worse results w.r.t. best known values.

Table 2. Summary of the results

Instances set	5	7	9	11	13	15	Total
RPD (%)	0.0304	1.4279	1.9537	1.4688	3.6823	7.7560	2.7199
Best RPD (%)	-35.70	-24.71	-20.29	-20.00	-40.16	-15.05	-40.16
instances improved	11	13	12	6	6	4	52
instances equal	73	46	18	26	15	13	191
instances worst	12	37	66	64	75	79	333

In order to compare the performance of the proposed approach, we selected the LA algorithm proposed by Bert et al. [3].It is important to mention that the authors report the results for the instances of 5, 7, 9, 11, 13 and 15 set of jobs respectively. In these case, the RPD is calculated.

Tables3 and 4 show these results.Table 5shows a summary of the comparison of the proposed method (QL-HFFS) and the LA algorithm.

Table 3. Comparison of QL-HFFS with LA on 5, 7 and 9 jobs Ruiz et al.[30] instances.

Instances set	5		7		9	
	LA	QL-HFFS	LA	QL-HFFS	LA	QL-HFFS
RPD (%)	0.0697	**0.0304**	2.0131	**1.4279**	**1.1568**	1.9537
Best RPD (%)	-35.70	-35.70	-24.71	-24.71	-26.92	-20.29
instances improved	11	11	12	13	18	12
instances equal	62	73	40	46	19	18
instances worst	23	12	44	37	59	66

Table 4. Comparison of QL-HFFS with LA on 11, 12 and 15 jobs Ruiz et al.[30] instances.

Instances set	11		13		15	
	LA	QL-HFFS	LA	QL-HFFS	LA	QL-HFFS
RPD (%)	1.6565	**1.4688**	3.7294	**3.7181**	7.9189	**7.7994**
Best RPD (%)	-21.10	-20.00	-43.34	-40.16	-10.46	-15.05
instances improved	12	6	9	6	6	4
instances equal	18	26	8	15	7	13
instances worst	66	64	79	75	82	79

Table 5. Summary of the results obtained by the algorithms QL-HFFS and the LA.

	Total	
	LA	QL-HFFS
RPD (%)	2.7484	**2.7199**
Best RPD (%)	-40.34	-40.16
instances improved	68	52
instances equal	155	191
instances worst	353	333

6. DISCUSSION

Based on the results from Table 2 we can see that the proposed algorithm is able to obtain good results.For the 576 instances, the proposed algorithm obtained 194 sequences of jobs with the optimal makespan, representing the 33.68%; 51 sequences where the optimal was improved (8.85%) and 333 (57.81%) instances where the makespan was worst. Figure 5 showsthe Gantt diagram of a solution found for an instance of 13 jobs, 2-stages and 3-machines per stage, where the best known value (976) is improved (580). The average of RPD was 2.7199%.

One may expect that results will be worse as the number of jobs increases, but this is evidently not the case in the set of instances with 11 jobs that yield better results than the ones with 9 jobs. The main causes of these results are the additional real-world restrictions (e.g. precedence constraints, skipping stages, machine eligibility and others).

Analyzing the results of Table 3 and 4, we can see that the proposed algorithm, except for 9-jobs instances, reached better results than LA method. In these case, the RPD of QL-HFFS was slightly less than LA. The best values of RPD are colored in the tables with dark gray.

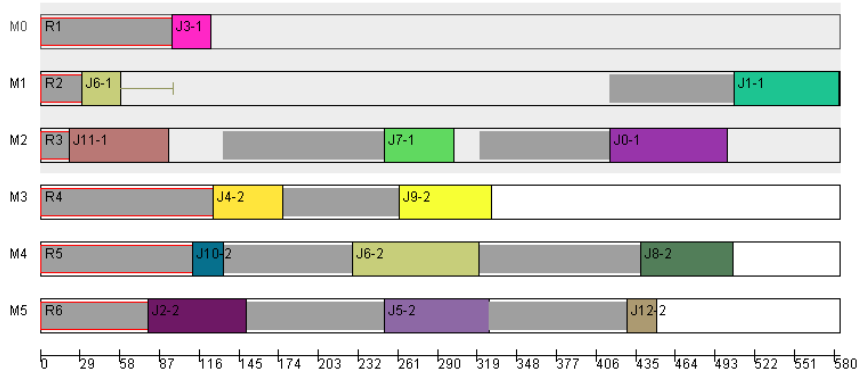

Figure 5. Sequence obtained for the instance of 13 jobs"Ism_13_2_3_1-200_50_50_75-125_50-100_-99-99_1-3_3" where $C_{max} = 580$.

As seen in Table 5, QL-HFFS and LA perform very well in Ruiz et al. instances without importantdifferences between them. Specifically, QL-HFFS has the lowest RPD from the best-known. This methods reached 16 improved instances less than LA algorithm, however, our approach found 39 optimum values more than LA. The difference of instances worst is 23 pro QL-HFFS.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a Reinforcement Learning Approach known as Q-Learning to solve a scheduling problem that comes from industry, with the objective of minimizing the makespan. The algorithm was calibrated by means of extensive experiments using a comprehensive set of 576 HFFS benchmark problems. The performance of this method was evaluated comparatively with an algorithm based on Learning Automata. The obtained computational results for the HFFS problems are very promising and we can conclude that the proposed method constitutes an interesting alternative to solve complex scheduling problems. In this sense, the experimental results indicated that the proposed QL-HFFS outperformed the LA in almost all the test problems. We want to highlight that the proposed algorithm is simple and easy to implement.

We are currently working on installing this algorithm in a company responsible for the production of spare pieces for industrial machines, specifically in the repair shop 'Manual Fajardo', located in Granma, a province at the east of Cuba, which is responsible for the manufacturing (and repairing) of several pieces of equipment that are used by the different machines involved in the sugar production process. On the other hand, we are considering other constraints such as transportation times, production priorities and machines breakdown with the objective of apply the proposed method to instances of greater complexity. These extensions will contribute to close the existing gap between the theory of scheduling and its applications in real industrial settings.

RECEIVED: MAY, 2017
REVISED: FEBRUARY, 2018

REFERENCES

[1] AKHSHABI, M. and J. KHALATBARI (2011):Solving flexible job-shop scheduling problem using clonal selection algorithm. **Indian Journal of Science and Technology,** 10, 1248-1251.
[2] ALISANTOSO, D., L. P. KHOO and P. Y. JIANG (2003):An immune algorithm approach to the scheduling of a flexible PCB flow shop. **International Journal of Advanced Manufacturing Technology,** 22, 819–827.
[3] BERT VAN VRECKEM, B., D. BORODIN, W. DE BRUYN and A. NOWÉ (2013): A Reinforcement Learning Approach to Solving Hybrid Flexible Flowline Scheduling Problems. **6th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA)**. **Gent, Belgium**.
[4] BERTEL, S. and J. C. BILLAUT (2004):A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. **European Journal of Operational Research,** 159, 651–662.
[5] BOZEJKO, W., J. PEMPERA and C. SMUTNICKI (2013):Parallel tabu search algorithm for the hybrid flow shop problem. **Computers & Industrial Engineering,** 65 466–474.
[6] FATTAHI, P., S. M. HASSAN-HOSSEINI, F. JOLAI, R. TAVAKKOLI-MOGHADDAM and C. D. TARANTILIS (2014):A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. **Applied Mathematical Modelling,** 119-134.
[7] FONSECA, Y. and M. MARTÍNEZ (2016): Adapting a Reinforcement Learning Approach for the Flow Shop Environment with Ssequence-Dependent Setup Time. **12th International Conference on Operations Research**. **Havana, Cuba**.
[8] FONSECA, Y., M. MARTÍNEZ, J. BERMUDEZ and B. MÉNDEZ (2015):A Reinforcement Learning Approach for Scheduling Problems. **Revista Investigación Operacional,** 36, 225-231.
[9] FONSECA, Y., Y. MARTÍNEZ, A. E. FIGUEREDO and L. A. PERNÍA (2014):Behavior of the main parameters of the Genetic Algorithm for Flow Shop Scheduling Problems. **Revista Cubana de Ciencias Informáticas,** 8, 99-111.
[10] GICQUEL, C., L. HEGE, M. MINOUX and W. VAN-CANNEYT (2012):A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-wait constraints. **Computers & Operations Research,** 39, 629–636.
[11] GUPTA, D. (1988):Two-stage, hybrid flow shop scheduling problem. **Journal of the Operational Research Society,** 39, 359–364.
[12] JOHNSON, S. M. (1954):Optimal two and three stage production schedules with setup times included. **Naval Research Logistics Quarterly,** 1, 402-452.
[13] JUN JOO, B., Y. CHAN CHOI and P. XIROUCHAKIS (2013): Dispatching rule-based algorithms for a dynamic flexible flow shop scheduling problem with time-dependent process defect rate and quality feedback. **Forty Sixth CIRP Conference on Manufacturing Systems**. **Lausanne, Switzerland**.
[14] JUNGWATTANAKIT, J., M. REODECHA, P. CHAOVALITWONGSE and F. WERNER (2009):A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. **Computers & Operations Research,** 36, 358-378.
[15] KAELBLING, L. P. and M. LITTMAN (1996):Reinforcement Learning: A Survey. **Journal of Artificial Intelligence Research,** 4, 237-285.
[16] KIANFAR, K., S. M. T. FATEMI-GHOMI and A. OROOJLOOY-JADID (2012):Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA. **Engineering Applications of Artificial Intelligence,** 25, 494–506.
[17] LINN, R. and W. ZHANG (1999):Hybrid flow shop scheduling: a survey. **Computers and Industrial Engineering,** 37, 57–61.

[18] MARTÍNEZ, Y. (2012): **A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems**. PhD Thesis, Vrije Universiteit Brussel, 169 p.
[19] MEHMET, Y. and Y. BETUL (2014):Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. **Omega,** 45 119-135.
[20]MORIARTY, D., A. SCHULTZ and J. GREFENSTETTE (1999):Evolutionary Algorithms for Reinforcement Learning. **Journal of Artificial Intelligence Research,** 11, 241-276.
[21] MOURSLI, O. and Y. POCHET (2000):A branch-and-bound algorithm for the hybrid flowshop. **International Journal of Production Economics,** 64, 113-125.
[22] NADERI, B. and V. ROSHANAEI (2009):Scheduling hybrid flowshops with sequence-dependent setup times to minimize makespan and maximum tardiness. **International Journal of Advanced Manufacturing Technology,** 41, 1186–1198.
[23] NADERI, B., R. RUIZ and M. ZANDIEH (2010):Algorithms for a realistic variant of flowshop scheduling. **Computers & Operations Research,** 37**,** 236-246.
[24] NEUFELD, J. S., J. N. D. GUPTA and U. BUSCHE (2016):A comprehensive review of flowshop group scheduling literature. **Computers and Operations Research,** 43**,** 251-269
[25] PARVIZ, F., H. H. SEYED MOHAMMAD, J. FARIBORZ and T.-M. REZA (2014):A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. **Applied Mathematical Modelling,** 38, 119-134.
[26] PINEDO, M. (2008): **Scheduling Theory, Algorithms, and Systems**. Prentice Hall Inc., New Jersey.
[27] RIBAS, I., R. LEISTEN and J. FRAMINAN (2010):Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. **Computers and Operations Research,** 37, 1439–1454.
[28] RUIZ, R. and C. MAROTO (2006):A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. **European Journal of Operational Research,** 169, 781–800.
[29] RUIZ, R., C. MAROTO and J. ALCARAZ (2006):Two new robust genetic algorithms for the flowshop scheduling problem. **Omega-International Journal of Management Science,** 35, 461–476.
[30] RUIZ, R., F. SIVRIKAYA and T. URLINGS (2008):Modeling realistic hybrid flexible flowshop scheduling problems. **Computers & Operations Research,** 35, 1151 – 1175.
[31] RUIZ, R. and J. VÁZQUEZ-RODRÍGUEZ (2010):The hybrid flow shop scheduling problem. **European Journal of Operation Research,** 205, 1-18.
[32] SEIDO NAGANOA, M., A. ALMEIDA DA SILVA and L. A. NOGUEIRA LORENA (2012):A new evolutionary clustering search for a no-wait flow shop problem with set-up times. **Engineering Applications of Artificial Intelligence,** 25, 1114–1120.
[33] SUTTON, R. and A. BARTO (2016): **Reinforcement Learning (An Introduction)**. The MIT Press, Cambridge, Massachusetts.
[34] TSITSIKLIS, J. (1994):Asynchronous stochastic approximation an Q-learning. **Machine Learning,** 16, 185-202.
[35] URLINGS, T. (2010): **Heuristics and metaheuristicsfor heavily constrained hybrid flowshop problems** Universidad Politécnica de Valencia, 349 p.
[36] URLINGS, T., R. RUIZ and T. STÜTZLE (2010):Shifting representation search for hybrid flexible flowline problems. **European Journal of Operation Research,** 205**,** 1086-1095.
[37] VALLADA, E. and R. RUIZ (2011):A Genetic algorithms for the unrelated parallel machine scheduling problem with sequence dependent setup times. **European Journal of Operation Research,** 206, 612–622.
[38] VIGNIER, A., J. C. BILLAUT and C. PROUST (1999):Hybrid flowshop scheduling problems: State of the art. **Rairo-Recherche Operationnelle-Operations Research,** 33, 117–183.
[39] WANG, S. and M. LIU (2013):A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. **Computers & Operations Research,** 40, 1064–1075.
[40] WATKINS, C. (1989):**Learning from delayed rewards**. PhD Thesis, University of Cambridge, 152 p
[41] WATKINS, C. and P. DAYAN (1992):Q-learning. **Machine Learning,** 8, 279-292.
[42] ZHANG, W. (1996): **Reinforcement Learning for Job Shop Scheduling** Oregon State University, 191 p.