Ingeniería Industrial
ISSN: 1815-5936
Facultad de Ingeniería Industrial, Instituto Superior Politécnico José Antonio Echeverría, Cujae.

Fonseca-Reyna, Yunior César; Martínez-Jiménez, Yailen; Nowé, Ann
Aprendizaje reforzado aplicado a la programación de tareas bajo condiciones reales
Ingeniería Industrial, vol. XXXIX, núm. 1, 2018, Enero-Abril, pp. 36-45
Facultad de Ingeniería Industrial, Instituto Superior Politécnico José Antonio Echeverría, Cujae.

Ingeniería Industrial/ISSN 1815-5936/Vol. XXXIV/No. 1/enero-abril/2018/p. 36-45

ARTÍCULO ORIGINAL
INVESTIGACIÓN DE OPERACIONES

Aprendizaje reforzado aplicado a la programación de tareas bajo condiciones reales

Reinforcement learning applied to scheduling under real constraints

Yunior César Fonseca-Reyna[I], Yailen Martínez-Jiménez[II], Ann Nowé[III]

[I] Universidad de Granma, Facultad de Ciencias Informáticas y Exactas. Granma, Cuba
E-mail: fonseca@udg.co.cu
[II] Universidad Central Marta Abreu de las Villas Facultad de Física, Matemática y Computación. Las Villas, Cuba
E-mail: yailenm@uclv.edu.cu
[III] Vrije Universite it Brussel, Computational Modeling Lab. Bruselas, Bélgica.
E-mail: ann.nowe@vub.ac.be

Recibido: 3/05/2016
Aprobado: 5/09/2017

RESUMEN
La variante de flujo regular o *flow shop scheduling* es un problema clásico de programación de la producción. Las primeras publicaciones científicas sobre este problema aparecieron hace más de medio siglo. Sin embargo, muchos autores han reconocido una brecha entre la literatura especializada y la problemática en entornos reales. En este trabajo se modeló el *flow shop* considerando los tiempos de configuración inicial de las máquinas, tiempos de montaje dependientes de la secuencia, precedencia entre trabajos, omisión de etapas por los trabajos y como función objetivo minimización del tiempo total de procesamiento o *makespan*. Hasta el momento no existen métodos computacionales que solucionen este problema teniendo en cuenta todas las características mencionadas. Para la solución se adaptó un enfoque del Aprendizaje Reforzado conocido como Q-Learning. Finalmente, se usaron instancias de problemas de diferentes tamaños y complejidad demostrando la efectividad de este método en cuanto a la calidad de las soluciones.

Palabras Claves: aprendizajere forzado; flow shop scheduling; optimización combinatoria; q-learning.

Abstract
The flow shop scheduling is a classic problem of production scheduling. The first scientific publications on production scheduling appeared more than half a century ago. However, many authors have recognized a gap between the specialized literature and the real environment problems. In this paper, the flow shop is modeled taking into account preparation time machines, setup-time between two jobs, precedence between jobs, the possibility to skip stages and as objective function minimizing the total processing time or make span. At this moment, there are no computational methods that can solve this problem taking into account all the mentioned elements. The Reinforcement Learning approach known as Q-Learning was adapted to use it in the solution of this problem. Finally, the algorithm was tested with problems of different levels of complexity in order to obtain satisfactory results in terms of solutions quality.

Keywords: flow shop, optimization; q-learning; reinforcement learning.

I. INTRODUCCIÓN

La variante de flujo regular, secuenciación de tareas o *Flow Shop SchedulingProblem (FSSP)*, es un problema de optimización que se presenta con frecuencia en sistemas de producción convencionales automatizados donde existen máquinas-herramientas convencionales y se fabrican diferentes tipos de productos que presentan una misma ruta tecnológica. Este es un problema común donde está involucrada la toma de decisiones con respecto a la mejor asignación de recursos a procesos de información, en los cuales se tienen restricciones de temporalidad [1, 2]. Adicionalmente, hay que garantizar que estos tiempos asignados a dichos procesos deben cumplir con una serie de restricciones establecidas en la programación de la producción, así como con la optimización de determinados criterios [2-5]. De esta forma, la secuenciación de tareas está directamente asociada con la ejecutabilidad y optimalidad de un plan preestablecido.

En los entornos reales de producción, la secuenciación y asignación de trabajos son actividades críticas que inciden en los tiempos de respuesta y los costos de producción, y por lo tanto, inciden directamente en la efectividad y eficiencia de las empresas. Particularmente, esta secuenciación óptima de las órdenes de producción en recursos limitados es un tópico importante que ha fascinado a científicos por años. A pesar de los grandes avances en esta área, todavía la secuenciación de tareas presenta desafíos mayores.

Una revisión en profundidad al extenso trabajo realizado en más de 50 años de existencia de lo que se ha denominado, empleando la terminología en lengua inglesa como *Scheduling*, pone de manifiesto la existencia de una necesidad que debe ser cubierta. Se trata de reducir la distancia entre los problemas planteados por la comunidad científica y los que desean resolver los profesionales. En los problemas académicos, los sistemas productivos, las condiciones de trabajo, las interrelaciones organizativas, o la propia aparición de eventos son considerados de forma parcial, o incluso ignorados. Es por eso que algunos autores han reconocido una brecha entre la literatura y la problemática industrial [6-9]. La mayoría de las investigaciones se concentran en problemas de optimización que no son más que una versión muy simplificada de lo que ocurre en un entorno de manufactura real, permitiendo el uso de métodos sofisticados que garantizan la obtención de soluciones óptimas.

Desde la década de los 50, científicos en el área de la investigación de operaciones se han apoyado en métodos matemáticos de optimización para problemas de *scheduling*, contribuyendo con una variedad amplia de metodologías en las que se incluye la programación lineal, programación entera mixta, etc. [10, 11] hasta técnicas avanzadas de Inteligencia Artificial (IA)[7, 12, 13].

Una de las primeras publicaciones en el campo de la programación de la producción fue presentada en 1954 por S.M. Johnson [14] en la cual se presenta la implementación de un algoritmo que obtiene secuencias óptimas para un problema de *n* trabajos en 2-máquinas. A partir de ese momento, otros investigadores han extendido este notorio resultado, obteniendo algoritmos complejos para casos más generales. Ancâu [15] propuso dos variantes de algoritmos heurísticos para resolver el FSSP clásico. Ambos algoritmos son simples y eficientes. El primero implementa una heurística constructivista basada en una selección ávida, mientras que el segundo algoritmo es una versión modificada del primero. Los resultados obtenidos muestran una buena posición para estos dentro de los algoritmos basados en heurísticas en el campo de la programación de la producción. Por otro lado, Ekta Singhal et. al [16] proponen una modificación de la heurística NEH para resolver el *Flow Shop* Permutacional mejorando soluciones en comparación al algoritmo original. La función objetivo de este algoritmo es la minimización del *make span.*

La técnica de Ramificación y Poda (Branch & Bound o B&B) puede encontrar soluciones óptimas a problemas de *scheduling* pero a un alto costo computacional, por consiguiente, este método no es adecuado para problemas de grandes dimensiones. Varios autores han aplicado está técnica en busca de buenas soluciones, tal es el caso de la investigación de Peter Bruker et. al [17], en la que los autores aplican esta metodología para el *Open Shop Scheduling Problem* (OSSP) basada en un grafo disyuntivo. Los resultados computacionales muestran que este método arroja resultados aceptables. En esta investigación, para algunos problemas propuestos por la comunidad científica, se obtienen por primera vez los valores óptimos.

Con el desarrollo de la IA han emergido otras metodologías como es el caso de las metaheurísticas. Considerando el estudio y la importancia de los problemas de *scheduling*, estas últimas han sido muy usadas para dar solución a los mismos. Entre ellas podemos encontrar el Recocido Simulado (SimulatedAnneling o SA), la Búsqueda Tabú (TabuSearch o TS) y los

Algoritmos Genéticos (Genetic Algorithms o GA). Takeshi Yamada[13] en su tesis doctoral aplica SA, TS, y GA para el *Job Shop SchedulingProblem* (JSSP) y para el FSSP como caso especial del JSSP. Li Wang et. al [18] proponen un GA híbrido para el FSSP donde múltiples operadores genéticos son combinados con una estructura de vecindad basada en un grafo que mejora significativamente la búsqueda local. Este algoritmo balancea eficientemente la exploración y la explotación. Nagar et. al [19] combina B&B con un GA encontrando soluciones aproximadas de *makespan* y *flowtime* para 2-máquinas y *n* trabajos. Otros autores aplican estas metaheurísticas obteniendo soluciones con cierto grado de éxito [4, 9, 20].

Por otro lado, Tasgetiren et. al [21] proponen dos algoritmos basados en la técnica de optimización Nube de Partículas (Particle Swarm Optimization o PSO) llamados PSOvns y HCPSO respectivamente, los cuales optienen varias soluciones óptimas para las primeras 90 instancias de problemas de FSSP propuestas por Eric Taillard localizadas en la *OR-Library*[22]. Rahimi-Vahed y Mirghorbani [23] hacen uso de esta metodología con el objetivo de minimizar el *makespan* y la tardanza (*tardiness*) en el FSSP. Este PSO multiobjetivo es comparado con un GA siendo el primero el que obtiene mejores resultados.

Todos los métodos descritos anteriormente garantizan, en muchos de los casos, soluciones óptimas que son logradas en un período de tiempo razonable cuando el tamaño del problema es pequeño. Desafortunadamente, los problemas de *scheduling* en el mundo real son de difícil solución y están clasificados técnicamente como de solución en un tiempo no polinomial (NP-hard) [3, 24]. Además, la inclusión de restricciones existentes en un entorno real dificulta la aplicabilidad de dichos métodos.

El modelado es un tema a considerar al solucionar problemas de *scheduling* en un entorno real. Los innumerables tipos de procesos de fabricación, cada uno con sus propias particularidades, hacen difícil construir modelos generalmente aplicables. Algunas restricciones son difíciles (o imposibles) de representar matemáticamente. Por otra parte, encontrar las restricciones correctas para modelar la realidad depende sobre todo del conocimiento extenso del dominio, que puede ser inasequible para los investigadores y que es solamente utilizable para un tipo particular de proceso de producción. Un segundo problema es tener en cuenta un conjunto de características presentes en entornos reales que dificulta y en muchos casos imposibilita la adaptación de muchas de las metodologías existentes al problema de la programación de la producción. Entre estas se pueden encontrar las siguientes:

- Tiempo de configuración inicial de las máquinas: En las empresas manufactureras antes de comenzar el proceso productivo, las máquinas son configuradas inicialmente según el tipo de trabajo que pueden realizar.
- Tiempos de montaje dependientes de la secuencia de cada uno de los trabajos: Las operaciones de cambio de referencia en las máquinas varían constantemente. Estas se demoran más cuando el trabajo entrante es muy diferente al saliente, impactando el desempeño total de las operaciones.
- Cada trabajo puede procesarse solo en un subconjunto de máquinas según sus características técnicas y su disponibilidad. Un trabajo puede saltar etapas de producción.
- Precedencia entre trabajos: Un trabajo, a veces, depende de la culminación de otro u otros para comenzar a ejecutarse. Esto se evidencia por ejemplo, cuando se tiene un trabajo auxiliar que forma parte de uno o varios trabajos ya culminados.

Todo esto da lugar a que en las empresas se tenga que revisar la política de generar buenas programaciones de la producción. A su vez, refuerza solamente la necesidad de automatizar el modelado de problemas aplicando algoritmos de aprendizaje para crear un sistema de previsión dinámica que pueda adaptarse a las características de un entorno real. La combinación de los distintos temas abordados ilustra la necesidad de investigar otras estrategias, así como nuevos y eficientes algoritmos para resolver el problema en cuestión. Por su parte, el Aprendizaje Reforzado (RL) puede ofrecer una buena solución pues es mucho menos dependiente en conocimiento del dominio y es capaz de "aprender" buenas políticas de planificación automáticamente. Este aparece en la literatura como un acercamiento prometedor para solucionar problemas de scheduling[25, 26].

Hasta el momento no existen métodos que resuelvan problemas de *scheduling* que contemplen las características de un entorno real mencionadas anteriormente al mismo tiempo, constituyendo esto el problema de esta investigación. Por ello se plantean las siguientes preguntas:

¿Es posible modelar cada una de las características que se presentan en un entorno real de programación de la producción ayudando a cerrar la brecha entre los problemas resueltos en la literatura especializada y los que se presentan en las empresas manufactureras?

¿Una vez realizado este modelado, es posible adaptar e implementar un enfoque del Aprendizaje Reforzado consecuente con dicha teoría que proporcione buenas soluciones para el FSSP?
El objetivo de este artículo consiste en solucionar problemas de programación de la producción más realistas, implementando un enfoque del Aprendizaje Reforzado (AR) llamado Q-Learning (QL) que evalúe su adaptabilidad a la secuenciación de la producción conocida como Flow Shop Scheduling teniendo en cuenta todas las características mencionadas anteriormente al mismo tiempo.

II. MÉTODOS

Variante de flujo regular o flow shop scheduling

Dentro de la teoría de scheduling se pueden distinguir un gran número de problemas. En estos se tiene un conjunto de ***N*** trabajos que han de ser procesados sobre un conjunto de ***M*** recursos o máquinas físicas siguiendo un patrón de flujo o ruta tecnológica. Una secuenciación consiste en encontrar para cada trabajo un tiempo o un intervalo de tiempos en los que este pueda procesarse en una o varias máquinas [4, 27]. El objetivo es encontrar una secuencia sujeta a una serie de restricciones que optimice una o varias funciones objetivo [20, 24, 27].
En general, en un problema de *scheduling* intervienen los siguientes elementos: trabajos, actividades, máquinas, tipo de *scheduling* (patrón de flujo) y objetivos. El problema que se analiza en este artículo está sujeto a las siguientes restricciones:

- Solo se cuenta con una máquina-herramienta de cada tipo por etapa.
- Las restricciones tecnológicas están bien definidas y son previamente conocidas, además de que son inviolables.
- No está permitido que dos operaciones del mismo trabajo se procesen simultáneamente.
- Un trabajo puede procesarse o no en una o varias etapas.
- Ningún trabajo puede ser procesado más de una vez en la misma máquina.
- Cada trabajo es procesado hasta concluirse, una vez que se inicia una operación esta se interrumpe solamente cuando se concluye.
- Ninguna máquina puede procesar más de un trabajo a la vez.
- Los tiempos de configuración se conocen de antemano.
- Existe precedencia entre trabajos.
- El tiempo de transportación entre etapas no es considerado.

Como se mencionó anteriormente, el objetivo es encontrar una secuencia de trabajos por etapas bajo la restricción de que el procesamiento de cada trabajo tiene que ser continuo con respecto al objetivo de minimizar el *makespan* o C_{max} como también se le conoce.
Bajo estas condiciones, el tiempo de procesamiento total corresponde al tiempo de culminación de procesamiento del último trabajo. En otras palabras, es el tiempo necesario para completar todos los trabajos.

Aprendizaje reforzado

El AR es un enfoque de la IA en el que los agentes aprenden a partir de su interacción con el ambiente. Es aprender qué acción tomar dada una situación determinada con el objetivo de maximizar una señal numérica de recompensa que da la medida de cuán buena fue la acción elegida por el agente.

En el paradigma del AR un agente se conecta a su ambiente mediante la percepción y acción, como lo descrito en la Figura 1. En cada paso de la interacción, el agente percibe el estado actual "s" de su ambiente y selecciona una acción "a" para cambiar este estado. Esta transición genera una señal de refuerzo "r", que es recibida por el agente. La tarea del agente es aprender una política de elección de acciones en cada estado, que le posibilite recibir un número máximo de recompensas acumuladas. Los métodos del aprendizaje reforzado exploran el ambiente todo el tiempo para obtener una política deseada [28].

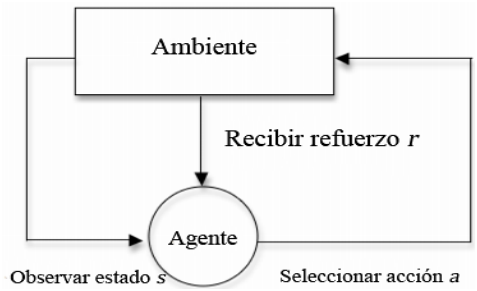

Fig. 1. Modelo estándar de Aprendizaje Reforzado

De manera formal, el modelo básico de AR consiste en:

- Un conjunto de estados del ambiente S.
- Un conjunto de acciones A.
- Un conjunto de "recompensas" en $\mathbb{R}$.
- Una función de transición T.

En cada instante t, el agente observa el estado $s_t \in S$ y el conjunto de posibles acciones $A(s_t)$. Escoge una acción $a \in A(s_t)$ y recibe del ambiente el nuevo estado s_{t+1} y una recompensa r_{t+1}, esto significa que el agente implementa una asociación entre estados y probabilidades de seleccionar cada posible acción. Esta asociación es la política del agente y se denota π_t, donde $\pi_t(s,a)$ es la probabilidad de seleccionar la acción a en el estado s en el instante t.
La función de recompensa define el objetivo en un problema de AR. Asocia cada estado observado (o par estado-acción) del ambiente con un valor numérico, indicando la deseabilidad de esa acción en ese estado. El objetivo de un agente usando Aprendizaje Reforzado es maximizar la cantidad total de recompensa que recibe a largo plazo. La función de recompensa define cuáles son los eventos buenos y malos para el agente.
De la misma forma en que la función de recompensa indica lo que es "inmediatamente" bueno, la función de valor especifica lo que es bueno a largo plazo. En otras palabras, el valor de un estado es la cantidad total de recompensa descontada que un agente puede esperar acumular en el futuro, comenzando en ese estado. Por ejemplo, un estado puede siempre conducir a bajas recompensas inmediatas, pero aún tener un alto valor porque es normalmente seguido por otros estados que conducen a recompensas altas.

Adaptación del algoritmo q-learning para la solución del FSSP
QL es un algoritmo del AR que se basa en la capacidad de aprendizaje de los agentes asociados al método.Este algoritmo trabaja aprendiendo de una función acción-valor que da la utilidad esperada de tomar una acción en un estado determinado. El núcleo del algoritmo es la actualización de un q-valor en cada iteración. Cada par estado-acción (s,a) tiene un q-valor asociado. Cuando la acción es seleccionada por el agente que se encuentra en un estado determinado, el q-valor para ese par estado-acción es actualizado en base a la recompensa recibida cuando se seleccionó esa acción y el mejor q-valor para el subsiguiente estado. La regla de actualización para cada par estado-acción es la siguiente:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))] \qquad \text{Ecuación 1}$$

En esta expresión $\alpha \in [0, 1]$ es la proporción de aprendizaje y r la recompensa o penalidad resultante de tomar una acción a en el estado s. La proporción de aprendizaje determina el grado por el cual el viejo valor es actualizado. Por ejemplo, si la proporción de aprendizaje es $\alpha = 0$, entonces no habrá actualización. Por otro lado, si $\alpha = 1$, el viejo valor es remplazado por el nuevo estimado. Usualmente es utilizado un valor pequeño, por ejemplo$\alpha = 0.1$. El factor de descuento (parámetro γ) tiene el rango de valores desde cero hasta uno. Si γ es cercano a cero el agente tiende a considerar solamente la recompensa inmediata. Si γ es cercana a uno el agente considerará la recompensa futura en mayor medida. En el algoritmo QL interviene además otro parámetro que permite el balance entre la exploración y explotación denotado por ε (épsilon). Todas las acciones a realizar tienen asociadas una probabilidad generada aleatoriamente. Si esta probabilidad está por debajo de ε se selecciona una acción aleatoria, de lo contrario se selecciona una acción de acuerdo con la política del agente [8].
El algoritmo 1 es usado por los agentes para aprender a partir de la experiencia y el entrenamiento. Cada episodio es equivalente a una sesión de entrenamiento. En cada uno de ellos el agente explora el ambiente y toma la recompensa hasta alcanzar el estado objetivo. El

propósito del entrenamiento es aumentar el conocimiento del agente representado por los q-valores. Entre mayor sea la cantidad de episodios se alcanzarán mejores resultados.
Los principales elementos que intervienen en el desempeño del método de solución propuesto son detallados a continuación:
Estados y acciones: Existe un agente asociado a cada etapa (en nuestro caso se asocia un agente por máquina pues solo existe una por etapa) y este agente tomará decisiones sobre futuras acciones. Para un agente el tomar una acción significa que debe decidir cuál va a ser el próximo trabajo a procesar del conjunto de trabajos posibles (estos son los que están esperando en la cola asociada al recurso correspondiente teniendo siempre en cuenta la precedencia entre ellos). El agente además de tener una visión local con la información asociada a su recurso (como los trabajos que están esperando por él), tiene conocimiento acerca de las colas de espera de los demás agentes así como la información asociada a estos.
Q-Valores: una matriz de ***m*** filas y ***n*** columnas es construida para almacenar los q-valores ya que para cada una de las ***m*** máquinas existen, al menos, ***n*** posibles trabajos a procesar.

En resumen, el algoritmo propuesto es el siguiente:

Algoritmo 1. Q-Learning aplicado al FSSP

Inicializar:
1. $Q(s,a) = \{\}$
2. $Mejor\ Secuencia = \{\}$
3. Para cada episodio:
   3.1. $s = \{\}$
   3.2. Mientras existan trabajos sin procesar:
      3.2.1 Por cada agente:
         3.2.1.1. Seleccionar entre los trabajos pendientes que puede ejecutar, el de mayor tiempo de procesamiento total J_m teniendo en cuenta su precedencia.
         3.2.1.2. Inicializar $acciones$ como el conjunto de posibles puntos de inserción de J_m dentro de s.
         3.2.1.3. Seleccionar a dentro de $acciones$ usando una política ε-*greedy*.
         3.2.1.4. Tomar acción a, observar s' y r como $1/\text{makespan}(s')$.
         3.2.1.5. $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
         3.2.1.6. $s \leftarrow s'$
   3.3. Si $makespan(s) < \text{makespan}(Mejor\ Secuencia)$
      $Mejor\ Secuencia \leftarrow s$

III. RESULTADOS

Al no existir instancias disponibles de problemas que contemplen las restricciones presentes en este artículo, los datos para los experimentos computacionales fueron generados aleatoriamente. Para comprobar la calidad de los resultados obtenidos se usaron 10 instancias de diferentes tamaños y complejidad que sirvieron para ejecutar el algoritmo propuesto. El tamaño de las instancias fue de 5x3, 5x4, 5x5, 7x6, 7x7, 8x8, 9x4, 9x9, 10x8 y 10x10 respectivamente. Se generaron números aleatorios para crear los tiempos iniciales de preparación de las máquinas y los tiempos de configuración entre trabajos. La tabla 1 muestra los tiempos de procesamiento de cada trabajo por máquina. La tabla 2, por su parte, detalla los tiempos de preparación inicial de las máquinas. La tabla 3 muestra los tiempos de configuración entre trabajos en cada una de las máquinas. Por último, la tabla 4 muestra la precedencia entre trabajos. Todos los datos mostrados en las siguientes tablas corresponden a la instancia 5x5.

Tabla 1. Tiempo de procesamiento

	Tiempo de Procesamiento				
Máquina	J_0	J_1	J_2	J_3	J_4
M_0	10	6	11	0	11
M_1	15	9	14	10	0
M_2	12	11	9	10	6
M_3	8	4	8	9	12
M_4	6	6	8	6	3

Tabla 2. Tiempo de preparación de las máquinas

Máquina	Tiempo de Preparación
M_0	9
M_1	3
M_2	8
M_3	16
M_4	23

Tabla 3. Tiempos de configuración entre trabajos en cada máquina

		Tiempo de Configuración							Tiempo de Configuración							Tiempo de Configuración						
Máquina		J_0	J_1	J_2	J_3	J_4	Máquina		J_0	J_1	J_2	J_3	J_4	Máquina		J_0	J_1	J_2	J_3	J_4
M_0	J_0	0	2	4	0	5	M_1	J_0	0	3	6	8	0	M_2	J_0	0	4	6	3	3
	J_1	3	0	7	0	4		J_1	2	0	5	4	0		J_1	2	0	5	7	3
	J_2	3	2	0	0	6		J_2	1	1	0	3	0		J_2	4	7	0	3	5
	J_3	0	0	0	0	0		J_3	4	4	5	0	0		J_3	8	8	5	0	12
	J_4	3	3	2	0	0		J_4	0	0	0	0	0		J_4	4	4	3	10	0

		Tiempo de Configuración							Tiempo de Configuración					
Máquina		J_0	J_1	J_2	J_3	J_4	Máquina		J_0	J_1	J_2	J_3	J_4
M_3	J_0	0	2	4	6	5	M_4	J_0	0	4	6	3	3
	J_1	3	0	7	9	4		J_1	2	0	5	7	3
	J_2	3	2	0	5	6		J_2	4	7	0	3	5
	J_3	2	4	6	0	7		J_3	8	8	5	0	12
	J_4	3	3	2	5	0		J_4	4	4	3	10	0

Tabla 4. Precedencia entre trabajos

Trabajo	Precedencia
J_0	-1
J_1	3 4
J_2	1
J_3	4
J_4	-1

Inicialmente se realizaron numerosos experimentos para analizar el proceso de aprendizaje utilizando típicas combinaciones de valores para los parámetros que intervienen en el algoritmo [25]. Las combinaciones propuestas para el algoritmo QL son las siguientes:

- Combinación 1: episodios = $n * m$, α = 0.1, γ= 0.8, ε = 0.2
- Combinación 2: episodios = $n * m$, α = 0.1, γ= 0.9, ε = 0.1
- Combinación 3: episodios = $n * m$, α = 0.1, γ= 0.8, ε = 0.1
- Combinación 4: episodios = $n * m$, α = 0.1, γ= 0.9, ε = 0.2

El comportamiento no determinístico de los algoritmos sobre múltiples conjuntos de datos es una razón por la cual no existe un procedimiento establecido para poder compararlos. Distintas técnicas estadísticas son utilizadas para tratar de determinar si las diferencias encontradas entre dos algoritmos son significativas. Debido a la no disponibilidad de problemas que contemplen las restricciones presentadas en el problema que se analiza en este artículo solo se realizó una comparación entre los rendimientos de cada combinación para las instancias propuestas y así determinar cuál de ellas tiene mejor desempeño. Para esto se tuvo en cuenta la moda de C_{max} obtenida después de 10 ejecuciones para 4 instancias escogidas aleatoriamente. La tabla 5 muestra estos resultados.

Tabla 5. Resultados de C_{max} por combinación de cada instancia

Combinación/Instancia	5x5	7x7	8x8	10x8
Combinación 1	172	183	361	244
Combinación 2	175	186	343	242
Combinación 3	172	183	343	241
Combinación 4	177	183	350	241

Luego de los experimentos realizados se decidió mantener la Combinación 3 para todas las instancias de problemas teniendo en cuenta la calidad de las soluciones obtenidas.
La figura 2 muestra el proceso de aprendizaje en la solución de la instancia 5x5 y 10x8 con la combinación de parámetros escogida. Las soluciones para dichas instancias son C_{max} = 172 y C_{max} = 241 respectivamente.

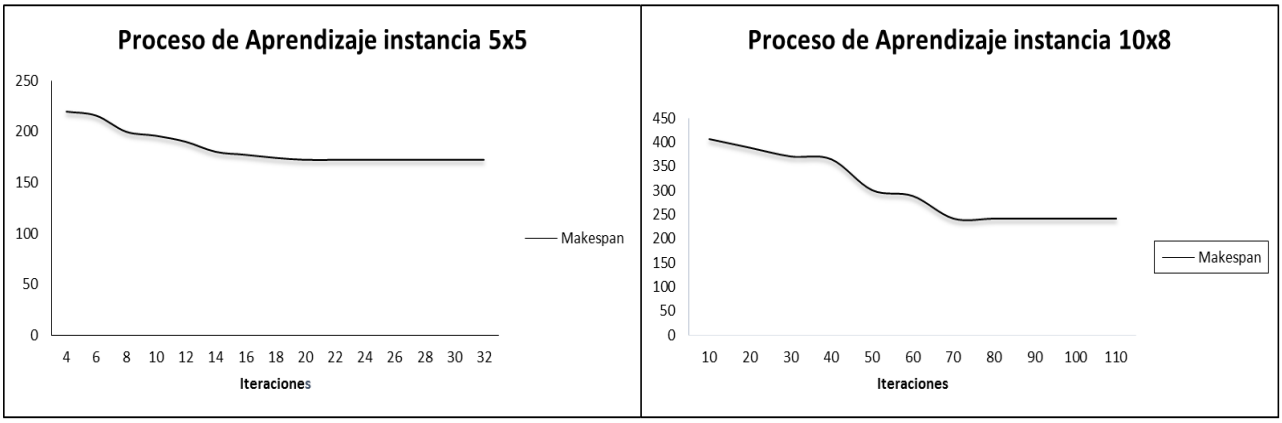

Fig. 2. Proceso de Aprendizaje para la instancia 5x5 y la instancia 10x8 utilizando la Combinación 3

El algoritmo QL fue implementado en Java, se ejecutó en una PC con un Corei3 a 3.4 GHz y 2 GB de memoria RAM. La Figura 3 muestra la solución de la instancia 5x5 donde el C_{max}= 172 a través un diagrama de Gantt. La Tabla 6 muestra los valores de C_{max}para las 10 instancias.

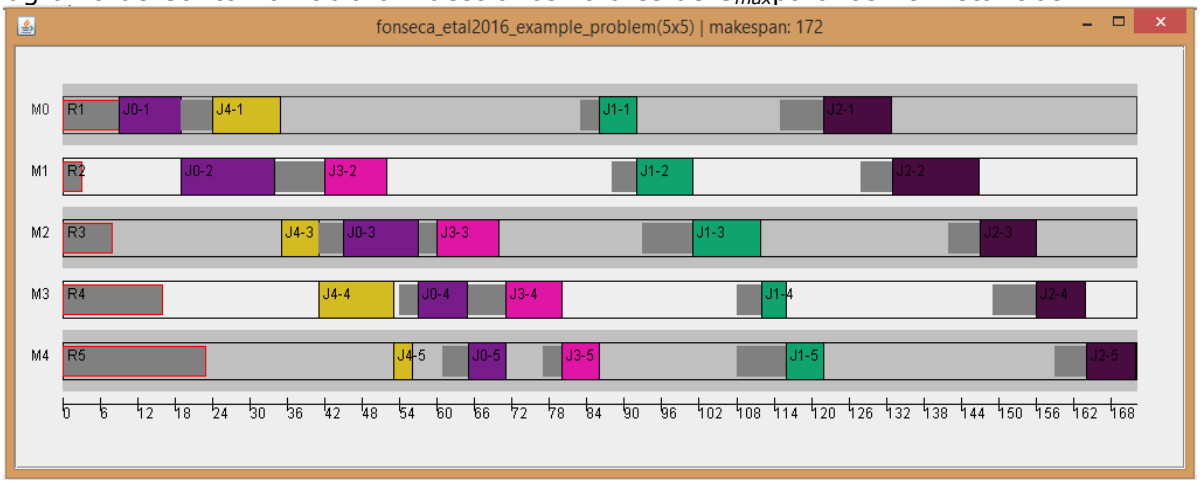

Fig. 3. Diagrama de Gantt para la instancia 5x5 donde C_{max} = 172

Tabla 6. Resultados del algoritmo Q-Learning para las intancias de problemas propuestas.

Instancia	C_{max}	Instancia	C_{max}
5x3	106	8x8	343
5x4	116	9x4	209
5x5	172	9x9	460
7x6	195	10x8	412
7x7	183	10x10	261

IV. DISCUSIÓN

La tabla 3 es interpretada como el tiempo que se necesita para configurar una máquina una vez que termina de procesar un trabajo y comenzar a ejecutar otro. En la tabla 4, el valor -1 significa que el trabajo no depende de la culminación de otro para comenzar su procesamiento. En caso contrario, el mismo deberá esperar por el procesamiento total de los trabajos que se especifican. Como se describe en la tabla 1, J_3 no es procesado por M_0. De igual manera, J_4 no es procesado por M_1. Por otra parte, en la tabla 4 se muestra que J_1 debe esperar a que culmine el procesamiento de J_3 y J_4, además, J_2 debe esperar por J_1y por último, J_3 debe esperar por la ejecución de J_4.

Una vez obtenidos los resultados mostrados en la tabla 5 se pudo apreciar que la combinación que mejores resultados de C_{max} obtuvo para las instancias escogidas fue la Combinación 3. Por su parte, la figura 2 muestra que el algoritmo necesita como mínimo $n * m$iteraciones para obtener soluciones óptimas. La determinación de estos parámetros permite la disminución del tiempo de procesamiento de las tareas lo cual contribuiría en un entorno manufacturero del mundo real a la satisfacción de los clientes, disminución del consumo eléctrico teniendo en cuenta el uso de las máquinas así como al incremento de los ingresos. Estos resultados fueron evaluados por especialistas de la fábrica Comandante Manuel "Piti" Fajardo perteneciente a la Empresa de Servicios Técnicos Industriales (ZETI) del sector azucarero localizada en la ciudad de Manzanillo, provincia Granma, determinando que los mismos presentan una excelente calidad.

Por último, los resultados obtenidos en la tabla 6 muestran la aplicabilidad del algoritmo QL a problemas de scheduling teniendo en cuenta un conjunto de restricciones que se presentan en un entorno real. Estos corresponden con lo planteado en la literatura especializada teniendo en cuenta la brecha que existe entre los problemas de la programación de la producción que comúnmente se resuelven por los especialistas en el tema y los que se presentan en ambientes manufactureros reales.

V. CONCLUSIONES

Se implementó un algoritmo basado en el Aprendizaje Reforzado denominado Q-Learning. Esta metodología fue adaptada al FSSP teniendo en cuenta un conjunto de características que se presentan en entornos reales de la producción. La misma fue evaluada con diez instancias de prueba de dicho problema proporcionando excelentes secuencias de *scheduling*. De acuerdo a los resultados obtenidos se llega a las siguientes conclusiones:

1. El método constituye una interesante alternativa para resolver problemas complejos de optimización.
2. La adaptación del algoritmo Q-Learning para el FSSP teniendo en cuenta características presentes en entornos reales proporciona resultados de excelente calidad.
3. Finalmente, se debe resaltar que el método propuesto es simple y fácil de implementar.

VI. REFERENCIAS

1. Márquez JE. Optimización de la programación (scheduling) en Talleres de Mecanizado [Tesis Doctoral]. Madrid, España: Universidad Politécnica de Madrid; 2012.
2. Tavares-Neto RF, Godinho-Filho M. An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. Computers & Operations Research. 2011;38:1286-93. DOI 10.1016.
3. Fattahi P, Hassan-Hosseini SM, Jolai F, et al. A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. Applied Mathematical Modelling. 2014;38:119-34.
4. Seido Naganoa M, Almeida da Silva A, Nogueira Lorena LA. A new evolutionary clustering

search for a no-wait flow shop problem with set-up times. Engineering Applications of Artificial Intelligence. 2012;25(1114-1120).
5. Jun Joo B, Chan Choi Y, Xirouchakis P. Dispatching rule-based algorithms for a dynamic flexible flow shop scheduling problem with time-dependent process defect rate and quality feedback. En: Forty Sixth CIRP Conference on Manufacturing Systems. Lausanne, Switzerland. p. 163-8.
6. Urlings T, Ruiz R, Stützle T. Shifting representation search for hybrid flexible flowline problems. European Journal of Operation Research. 2010;2007:1086-5.
7. Naderi B, Ruiz R, Zandieh M. Algorithms for a realistic variant of flowshop scheduling. Computers & Operations Research. 2010;37:236-46.
8. Bert Van Vreckem B, Borodin D, De Bruyn W, et al. A Reinforcement Learning Approach to Solving Hybrid Flexible Flowline Scheduling Problems. En: 6th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA). Gent, Belgium. p. 402-9. ISBN 13:978-0954582104.
9. Chaudhry IA, Munem khan A. Minimizing makespan for a no-wait flowshop using genetic algorithm. Sadhana. 2012;36(6):695-707.
10. 10. Manne AS. On the Job-Shop Scheduling Problem. Operations Research. 1960;8(2):219.
11. 11. Šeda M. Mathematical Models of Flow Shop and Job Shop Scheduling Problems. World Academy of Science. Engineering and Technology. 2007;1(31):122-7.
12. González M. Soluciones Metaheurísticas al Job-Shop Scheduling Problem with Sequence-Dependent Setup Times [Tesis Doctoral]. Oviedo: Universidad de Oviedo; 2011.
13. Yamada T. Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems [Tesis Doctoral]. Kyoto University: Kyoto, Japan; 2003.
14. Johnson SM. Optimal two and three stage production schedules with setup times included. Naval Research Logistics Quarterly. 1954;1:402-52.
15. Ancâu M. On Solving Flow Shop Scheduling Problems. Proceedings of the Romanian Academy. 2012;13(1):71-9.
16. Singhal E, Singh S, Dayma A. An Improved Heuristic for Permutation Flow Shop Scheduling (NEH ALGORITHM). International Journal Of Computational Engineering Research. 2012;2(6):95-100.
17. Brucker P, Hurink J, Jurisch B, et al. A branch & bound algorithm for the open-shop problem. Elsevier Science Discrete Applied Mathematics. 1997;76:43-59. ISSN. DOI
18. Ling Wang L, Zhang L, Zheng DZ. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. Computers & Operations Research. 2006;33:2960-71.
19. Nagar A, Heragu S, Haddock J. A branch and bound approach for two-machine flowshop scheduling problem. Journal of the Operational Research Society. 1995;46:721-34.
20. Chie-Wun C, Wen-Min C, Chin-Min L, et al. A genetic algorithm for scheduling dual flow shops. Expert Systems with Applications. 2012;39:1306-14.
21. Tasgetiren MF, Liang YC, Sevkli M, et al. A particle swarm optimization algorithm for make span and total flow time minimization in the permutation flowshop sequencing problem. European Journal of Operational Research. 2007;177:1930-47.
22. Taillard E. Benchmarks for basic scheduling problems. European Journal of Operational Research. 1993;64(2):278-85.
23. Rahimi-Vahed A, Mirghorbani S. A multi-objective particle swarm for a flowshop scheduling problem. Journal of Combinatorial Optimization. 2007;13(1):79-102.
24. Pinedo M. Scheduling Theory, Algorithms, and Systems. 3ra ed. New Jersey: U.S.A: Prentice Hall Inc; 2008. p. 586.
25. Martínez Y. A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems [Thesis]. Brussel, Belgium: Universiteit Brussel; 2012.
26. Suárez Y. Solución al problema de secuenciación en máquinas paralelas utilizando Aprendizaje Reforzado Villa Clara, Cuba: Universidad Central de las Villas; 2010.
27. Mehmet Y, Betul Y. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. Omega. 2014;45:119-35.
28. Fonseca Y, Martínez M, Bermudez J, et al. A Reinforcement Learning Approach for Scheduling Problems. Revista Investigación Operacional. 2015;36(3):225-35.