ALGORITMOS PARA PROBLEMAS DE SECUENCIACIÓN DE TAREAS EN AMBIENTES ONLINE

Beatriz M. Méndez Hernández [*1], Yailen Martínez Jiménez[*], Jessica Coto Palacio[*], Erick D. Rodríguez-Bazan[*], Ann Nowé[**]
* Facultad de Matemática-Física-Computación, Universidad Central "Marta Abreu" de Las Villas, Cuba.
** Artificial Intelligence Lab, Vrije Universiteit Brussel, Bélgica.
1 Departamento de Computación, Universidad Central "Marta Abreu" de Las Villas, Carretera a Camajuaní km. 5 ½, Santa Clara Villa Clara, Cuba.

ABSTRACT
Scheduling problems are present in many processes that occur in the manufacturing industry, where it is necessary to perform a set of operations at certain periods of time and it also needs to allocate limited resources to perform these tasks. Some of these problems occur in online environments, because there is no prior knowledge of the arrival of the jobs or the time it would take for each job to be processed in each of the machines. In this paper we study and propose a solution to online scheduling problems based on an existing case study from the literature, using two Reinforcement Learning algorithms. Besides we extend the study case to two more complex scenarios. The results obtained show the superiority of Q-Learning algorithm over Learning Automata algorithm due to the flexibility of Q-Learning's parameters. These results were validated using statistical tests where an algorithm is better than other if its difference between the generated and the processed jobs is less. Friedman test applied among all variants to find significant differences and also we applied Wilcoxon test to determinate the best algorithm by scenario.

KEYWORDS: online environments, reinforcement learning, Q-Learning, Learning Automata

MSC: 68T05, 68T20, 68T37, 90C15, 90C59

RESUMEN
Los problemas de secuenciación de tareas se ven reflejados en muchos de los procesos que se llevan a cabo en la industria manufacturera, donde se hace necesario realizar un conjunto de operaciones en espacios de tiempo determinados y a su vez asignar recursos limitados. Algunos de esos problemas ocurren en ambientes online, debido a que no se tiene conocimiento previo de la llegada de los trabajos ni del tiempo que tardará cada trabajo en ser procesado en cada una de las máquinas. En este artículo se estudia y se plantea una solución a problemas de secuenciación de tareas en ambientes *online* a partir de un caso de estudio existente en la literatura, utilizando dos de los principales algoritmos de Aprendizaje Reforzado. Además se extiende el caso de estudio a dos escenarios más complejos. Los resultados obtenidos demuestran la superioridad del algoritmo Q-Learning sobre el algoritmo Learning Automata debido a la flexibilidad de sus parámetros. Estos resultados fueron validados mediante pruebas estadísticas considerándose un algoritmo mejor que otro si tenía menor cantidad de trabajos sin procesar. Se aplicó la prueba de Friedman entre todas las variantes de los algoritmos y la prueba de Wilcoxon dos a dos para determinar cuál era el mejor algoritmo por escenario.

1. INTRODUCCIÓN

La secuenciación de tareas no es más que la asignación de entidades (personas, tareas, vehículos, conferencias, exámenes, reuniones, etc.) a un modelo en el espacio de tiempo, de tal manera que las restricciones impuestas sean satisfechas y ciertas metas sean logradas, es decir, 'un plan para ejecutar un trabajo o alcanzar un objetivo, especificando el orden y el tiempo asignado para cada parte'. En los problemas del mundo real, expresar las condiciones que hacen una secuenciación preferible sobre otra e incorporar esta información en un sistema automatizado, no resulta una tarea fácil. En los procesos de manufactura la secuenciación se define como un proceso de optimización que asigna recursos limitados, específicamente máquinas, en el tiempo a actividades que se pueden ejecutar en paralelo. Esta asignación tiene que cumplir un conjunto de restricciones que reflejan las relaciones temporales entre las actividades y las limitaciones de capacidad del conjunto de recursos compartidos [5, 14].
Los problemas de secuenciación de tareas se pueden desarrollar en ambientes offline o ambientes online, donde su principal diferencia radica en que en ambientes offline se conoce toda la información necesaria de antemano, como por ejemplo, el tiempo de procesamiento de las operaciones, mientras que en ambientes online no se conoce cuando acaba un trabajo hasta que termina de ser procesado. En los últimos años el Aprendizaje Reforzado ha servido de plataforma para la solución de problemas de secuenciación de tareas [3, 9, 18], donde cada agente es entrenado para

alcanzar una meta u objetivo específico. El aprendizaje denota cambios en un sistema que lo hacen capaz de realizar la misma tarea o tareas de forma más eficiente y más efectiva la próxima vez [16]. El Aprendizaje Reforzado también es conocido como un *framework* muy popular para el diseño de agentes que interactúan con su ambiente en aras de aprender a resolver una tarea específica a través de esas repetidas interacciones. En muchas ocasiones resulta más eficiente contar con más de un agente para llevar a cabo la tarea prevista, esto es conocido como sistemas multi-agente, el cual consta de una red de agentes que trabajan juntos para resolver uno o diversos problemas.
En este artículo se presenta un algoritmo para resolver problemas de secuenciación de tareas en ambientes online usando Aprendizaje Reforzado y aproximaciones estadísticas. En el siguiente epígrafe se describen los ambientes online y los diferentes enfoques que han sido desarrollados para resolverlos. Luego se exponen los fundamentos del Aprendizaje Reforzado y dos de sus principales algoritmos, Q-Learning y Learning Automata. En la próxima sección se describe el problema a resolver y la configuración de estos dos algoritmos para adaptarse al problema. En la sección 5 se exponen los resultados obtenidos por los algoritmos desarrollados, con diferentes combinaciones de parámetros, para el problema descrito y dos extensiones del mismo. El artículo culmina con las conclusiones.

2. AMBIENTES *ONLINE*

Los algoritmos de secuenciación *online* empezaron a ser investigados en los años 60, pero un estudio detenido y extensivo sólo se ha comenzado a llevar a cabo hace 15 años, después de introducirse formalmente el concepto de análisis competitivo. Actualmente existe una gran variedad de literatura en el área de la secuenciación *online*. En problemas de secuenciación *online* una secuencia de trabajos $\sigma = J_1, J_2, \ldots, J_n$ tiene que procesarse en un número determinado de máquinas. Los trabajos arriban al sistema uno a uno, y cuando un nuevo trabajo llega tiene que ser inmediatamente despachado hacia una de las máquinas, sin tener conocimiento sobre trabajos futuros. En caso de que todas las máquinas estén ocupadas el trabajo se mantiene en la cola de espera de la máquina a la que fue asignado hasta que pueda ser despachado. El objetivo es optimizar una función objetivo dada.

En [2] se presenta un estudio experimental de algoritmos para uno de los principales problemas de secuenciación online. A este problema se le conoce como el problema de Graham y ha sido extensamente investigado desde un punto de vista teórico [1, 4, 12]. En [13] se estudia el problema de secuenciación en una sola máquina, donde se definen dos algoritmos *online*: *First-Come-First-Served* (FCFS) y *Shortest-Available-Job-First* (SAJF), este tipo de problemas también es abordado en [8], donde los autores abarcan desde los problemas de una sola máquina hasta los que presentan múltiples máquinas. Sin embargo, este problema no se ha abordado utilizando métodos de Aprendizaje Reforzado combinado con reglas clásicas de despacho, de ahí la idea de utilizar dichas técnicas en la solución del mismo, ya que este tipo de aprendizaje es uno de los que está siendo actualmente aplicado en la solución de problemas de secuenciación de forma *offline*.

3. APRENDIZAJE REFORZADO

El Aprendizaje Reforzado o RL (terminología en inglés) se asocia con aprender qué hacer (cómo asociar situaciones con acciones) de forma tal que se maximice una señal numérica de recompensa. Es el problema encarado por agente que debe aprender comportamientos a través de interacciones a 'prueba y error' con un ambiente dinámico. Un agente es un sistema computacional situado en algún ambiente, que es capaz de actuar de manera autónoma y flexible en dicho ambiente en aras de cumplir con su objetivo [11, 10]. Uno de los desafíos del RL es el intercambio entre la exploración y la explotación. Para obtener una recompensa alta, un agente de RL debe preferir acciones que ha probado en el pasado y ha encontrado que son efectivas a la hora de producir recompensa. Pero para descubrir esas acciones, tiene que probar otras que no ha seleccionado antes.
El modelo básico de RL consiste en:

- Conjunto de estados del ambiente S
- Conjunto de acciones A
- Conjunto de recompensas R
- Función de transición T

3.1 Algoritmo Q-Learning

Uno de los adelantos más importantes en el Aprendizaje Reforzado fue el de Watkins en 1992 [17] con el desarrollo del algoritmo Q-*Learning* (QL). Dicho algoritmo se basa en aprender una función acción-valor que brinda la utilidad esperada de tomar una acción determinada en un estado específico. El centro del algoritmo es una simple actualización de valores, cada par *(s, a)* tiene un Q-valor asociado, cuando la acción *a* es seleccionada mientras el agente está en el estado *s*, el Q-valor para ese par estado-acción se actualiza basado en la recompensa recibida por el agente al tomar la

acción. También se tiene en cuenta el mejor Q-valor para el próximo estado s'. La regla de actualización completa es la siguiente:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma * \max Q(s',a') - Q(s,a)] \quad (3.1)$$

En la expresión anterior, $\alpha \in [0,1]$ representa la velocidad del aprendizaje y *r* la recompensa o penalización resultante de ejecutar la acción *a* en el estado *s*. El factor de descuento γ, determina el grado por el cual el valor anterior es actualizado. Normalmente se utiliza un valor pequeño para la velocidad de aprendizaje, por ejemplo, $\alpha = 0.1$. El factor de descuento (parámetro γ) también toma un valor entre 0 y 1, si está cercano a 0 entonces el agente tiende a considerar sólo la recompensa inmediata, si está cercano a 1 el agente considerará la recompensa futura como más importante.

3.2 Algoritmo Learning Automata

El *Learning Automata* (LA) representa un enfoque de iteración por política que fue introducido en los años cincuenta [6]. Un LA toma decisiones manteniendo una distribución de probabilidad sobre sus acciones. Formalmente puede ser definido como *(A, B, p, U)*, donde $A = \{a_1, a_2, \ldots, a_r\}$ es el conjunto de posibles acciones que el autómata puede ejecutar, *p* es la distribución de probabilidad de las acciones, *B* es el conjunto de respuestas del ambiente (valores entre 0 y 1) y *U* es el esquema de aprendizaje utilizado para actualizar *p*.

En cada intervalo de tiempo, el agente selecciona una de las acciones posibles según su distribución de probabilidad. Después de llevar a cabo la acción seleccionada *i*, su probabilidad p_i es actualizada basada en la recompensa $r \in \{0,1\}$, vea Ecuación 3.2. Las demás probabilidades p_j (para todas las acciones $i \neq j$) se ajustan de forma que la suma de todas las probabilidades sea 1 ($\sum_i p_i = 1$), vea Ecuación 3.3. Este algoritmo se basa en la simple idea de que cada vez que la acción seleccionada resulte en una respuesta favorable, la probabilidad de esa acción es aumentada, de lo contrario es disminuida.

$$p_i \leftarrow p_i + \alpha r(1 - p_i) - \beta(1 - r)p_i \quad (3.2)$$

$$p_j \leftarrow p_j - \alpha r p_j + \beta(1 - r)\left(\frac{1}{n-1} - p_j\right), \forall j \neq i \quad (3.3)$$

Los parámetros α y β (α, β∈[0,1]) son los factores de aprendizaje de la recompensa y la penalización respectivamente. En la literatura se definen tres esquemas comunes de actualización basados en los valores de α y β:

- Lineal Reward-Inaction (L_{R-I}) para β = 0,
- Lineal Reward-Penalty (L_{R-P}) para β=α,
- Lineal Reward-e-Penalty ($L_{R-\epsilon P}$) para β≪α.

4. DESCRIPCIÓN DEL PROBLEMA

Un ejemplo real de problema de secuenciación de tareas en ambientes *online* fue expuesto en [15], el cual está basado en la planta química de producción de Castillo y Roberts [7]. En esta planta, como muestra la Figura 1, se producen dos tipos de productos y se cuenta con una serie de una o más etapas procesadoras con unidades paralelas de procesamiento (máquinas) en cada etapa. El orden de llegada de cada producto y el tiempo de permanencia en cada máquina es generado siguiendo una distribución exponencial con una media determinada.

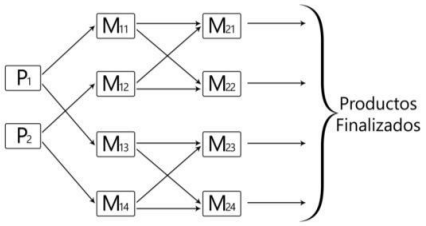

Figura 1. Ejemplo de planta química de producción con dos etapas de procesamiento. En cada etapa existen cuatro máquinas paralelas.

Los dos algoritmos de Aprendizaje Reforzado vistos anteriormente se adaptan a este tipo de problemas de manera muy similar, la única diferencia esta en la forma de actualizar la función de transición. Sin embargo, en el próximo epígrafe se verá la importancia de esta función en el desempeño de los algoritmos. En la figura 2 se muestra un diagrama general para representar los pasos de los algoritmos.

4.1 Aplicación del Algoritmo QL a problemas de secuenciación de tareas

Cuando se aplica el algoritmo QL a problemas de secuenciación de tareas, se deben tener en cuenta elementos importantes que ya han sido aludidos con anterioridad, pero que serán mejor explicados a continuación.

Estados: posibles fases o etapas por las que puede pasar cada trabajo dentro del sistema. En este caso existirá un estado en lo que se llama punto de decisión, que no es más que cada lugar donde haya que decidir la próxima máquina que ejecutará cada trabajo a procesar por el sistema.

Acciones: conjunto de posibles elementos a escoger desde el estado actual, en el problema de secuenciación de tareas online que se aborda en este artículo, seleccionar una acción equivale a seleccionar una máquina para procesar el trabajo actual.

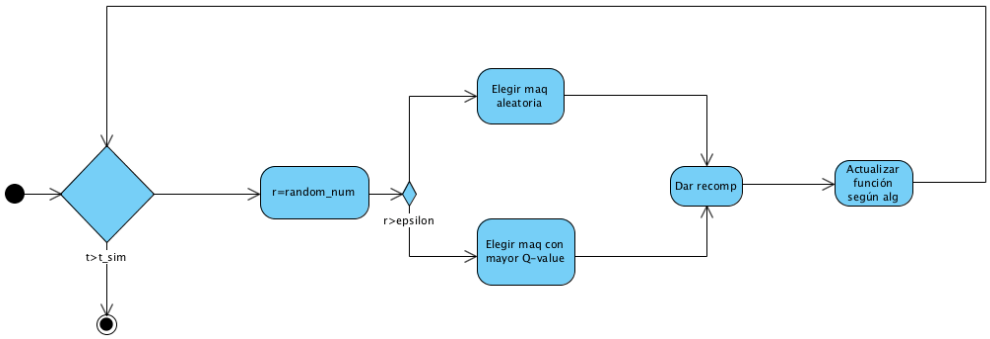

Figura 2. Diagrama para representar el comportamiento general de los algoritmos.

Q-Valores: arreglo de pares estado-acción que refleja, por cada estado, cuán buena resulta cada una de las posibles acciones que pueden tomarse. Este valor se actualiza cada vez que se selecciona una acción de acuerdo a la Ecuación 3.1.

Recompensa: valor recibido por el agente como señal de retroalimentación una vez que ha seleccionado una acción desde un estado determinado y que indica la calidad de la misma.

Estrategia de selección de acciones: Para seleccionar las acciones la estrategia que se usa es la ε-*greedy*, pues el uso de la misma proporciona una mayor exploración del espacio de soluciones. El parámetro ε, como se indica en la literatura, debe ser un valor pequeño, idealmente 0.1, lo que indica que el 10% de las veces se explora y el 90% restante se explota, es decir, se escoge la mejor de las opciones existentes en el momento de la decisión.

El algoritmo QL aplicado al problema en cuestión funciona de la siguiente manera:

Cada vez que un agente selecciona una acción se genera un número aleatorio, el cual es comparado con el valor de ε, si es menor que dicho valor el agente seleccionará una acción al azar, de lo contrario, la acción escogida será la mejor máquina posible, es decir, la que tenga el mayor Q-valor. Si varias máquinas tienen el mismo y el mejor Q-valor, se seleccionará la que tenga menos trabajos en cola esperando por ser procesados.

Otro aspecto importante durante la ejecución del algoritmo es la recompensa que recibe cada agente al seleccionar una acción, si la opción escogida es la mejor recibirá una recompensa, sino una penalización. En este caso se analizaron tres formas distintas de recompensar, la primera fue teniendo en cuenta el tiempo que estuvo el trabajo procesándose en la máquina (tiempo de procesamiento), en la segunda se consideró el tamaño de la cola y en la tercera el tamaño de la cola multiplicado por la velocidad de las máquinas. La tercera forma de recompensa fue la implementada ya que muestra en mejor medida cuál máquina sería la indicada para llevar a cabo el procesamiento del trabajo, pues presenta un equilibrio entre posible velocidad de procesamiento y posible tiempo de espera en cola. La penalización viene dada por la disminución del Q-valor al recibir poca recompensa. El punto cumbre del algoritmo es la actualización de los Q-valores que son contenidos en arreglos con dimensiones proporcionales a la cantidad de máquinas que maneja cada agente.

4.2 Aplicación del Algoritmo LA a problemas de secuenciación de tareas

La selección de acciones en cada punto de decisión es basada en probabilidades, por tanto, cada agente contiene un conjunto de probabilidades de tamaño igual a la cantidad de posibles acciones a escoger, las cuales se relacionan con cada una de las máquinas. Todas las probabilidades son inicializadas a partes iguales de manera tal que la suma total por agente sea igual a uno, a medida que el algoritmo va siendo ejecutado las probabilidades son actualizadas teniendo en cuenta la acción seleccionada y la recompensa obtenida.

A cada acción tomada se le asocia una recompensa que viene dada por la expresión:

$$r = \begin{cases} 0 & si\ F > F_{avg} \\ 1 & en\ otro\ caso \end{cases} \quad (4.1)$$

donde *F* es el tiempo que pasa un trabajo en el sistema. Es decir, si el tiempo que pasó el trabajo procesándose más el tiempo que pasó en cola es mayor que el tiempo promedio de todos los trabajos que han pasado por el sistema (tiempo total de trabajos procesados más el total de tiempo en cola de todos los trabajos entre la cantidad de trabajos finalizados), entonces la recompensa es equivalente a 0 o mejor dicho, se penaliza con 0, en caso contrario se recompensa con 1.

Durante el proceso de simulación, una vez que ocurra el evento de terminación de un trabajo, se actualizan todas las probabilidades de las acciones asociadas a los agentes que se encargaron de distribuir el procesamiento del trabajo en las distintas etapas. Esta tarea es llevada a cabo en cada etapa y en la última etapa se realiza una actualización especial, es decir, las probabilidades de las primeras etapas sufren modificaciones, ya que son re-calculadas, con el objetivo de valorar mejor la decisión del agente midiendo la trayectoria total del trabajo.

Según la acción escogida por el agente su probabilidad es actualizada mediante la Ecuación 3.2 expuesta con anterioridad y las probabilidades de las demás acciones no escogidas son recalculadas utilizando la Ecuación 3.3. Si la recompensa que se obtuvo fue de 1 entonces la probabilidad de la máquina escogida va a aumentar en dependencia del esquema de actualización y a su vez del valor asignado a α, y la probabilidad de las demás máquinas disminuirá. En caso de una recompensa igual a 0 ocurrirá lo contrario, excepto en el esquema actualización L_{R-I} donde las probabilidades se mantendrán iguales.

5. RESULTADOS OBTENIDOS

Para medir el rendimiento de los diferentes algoritmos indicados con anterioridad se utilizaron tres escenarios diseñados de la siguiente manera:

- Escenario 1 (Problema original): Conjunto de 2 etapas y 4 máquinas en cada etapa.
- Escenario 2: Conjunto de 3 etapas y 4 máquinas por etapas.
- Escenario 3: Conjunto de 2 etapas y 6 máquinas por etapas.

Se tuvieron en cuenta los valores de los siguientes parámetros para la simulación en la ejecución de los algoritmos.

- Tiempo de la simulación: 400 horas = 24000 minutos
- Media del fin de procesamiento en cada máquina: velocidad de cada máquina (esta media se utiliza como parámetro de la distribución exponencial)
- Velocidades de las máquinas: {M_{11}…M_{1n}, M_{21}…M_{2n}, M_{31}…M_{3n}}
    - Escenario 1: {30, 50, 100, 100, 100, 50, 30, 50}
    - Escenario 2: {30, 50, 100, 100, 100, 50, 30, 50, 60, 90, 40, 80}
    - Escenario 3: {30, 50, 100, 100, 80, 60, 100, 50, 30, 50, 100, 80}
- Media del arribo de los productos tipo 1 y tipo 2: 45

Para los parámetros del algoritmo QL:

- Velocidad de aprendizaje: α = 0.1
- Factor de descuento: γ = 0.8, γ = 0.9
- Épsilon: ε = 0.1, ε = 0.2

Para los parámetros del algoritmo LA:

- Lineal Reward-Inaction (L_{R-I}) β = 0, α = 0.1
- Lineal Reward-Penalty (L_{R-P}) β = α = 0.1
- Lineal Reward-ϵ-Penalty ($L_{R-\epsilon P}$) β = 0.1, α = 0.3

Los resultados para el estudio de los algoritmos se obtuvieron a partir de 15 corridas realizadas a cada combinación posible de parámetros, en los distintos escenarios. De cada corrida se reporta un conjunto de datos necesarios, primeramente se obtiene la cantidad de trabajos generados, de ese total se analiza cuántos fueron realmente procesados y de este número cuántos correspondieron a productos tipo 1 y cuántos a productos tipo 2. Por último se consideró el porciento que representan los trabajos procesados de los trabajos generados.

Se calculó, por cada escenario y posible combinación de parámetros de cada algoritmo, el porciento promedio de los trabajos procesados, expresados en la Tabla 1.

En el primer escenario, el cual es considerado como el básico y el más sencillo, el LA con $\alpha = 0.3$ y $\beta = 0.1$ arroja mejores resultados que los demás, mientras que en el segundo y tercer escenarios el QL muestra un mejor comportamiento con $\gamma = 0.9$, $\alpha = 0.1$, $\varepsilon = 0.1$ y $\gamma = 0.8$, $\alpha = 0.1$, $\varepsilon = 0.2$ respectivamente.
Los resultados obtenidos fueron además validados utilizando pruebas estadísticas, teniendo en cuenta los trabajos generados menos los trabajos procesados, para cada combinación de las mostradas en la Tabla 1. A cada escenario se le aplicaron pruebas no paramétricas, específicamente el test de *Friedman* para una primera valoración de las diferencias significativas entre los algoritmos, luego de comprobar dichas diferencias se identificó el mejor algoritmo según el que tuviera menor rango promedio, y se le aplicó el test de Wilcoxon dos a dos con los demás algoritmos.
En los tres escenarios la significación asintótica equivale a $0.000 < 0.05$ (Tabla 2, Tabla 3 y Tabla 4), por lo que se rechaza la hipótesis nula y se acepta la hipótesis alternativa, concluyendo que existen diferencias significativas entre al menos dos de los algoritmos. Luego de analizar los rangos promedios mostrados en la Tabla 2 se ratifica que el algoritmo LA con $\alpha = 0.3$ y $\beta = 0.1$ arroja mejores resultados que los demás. Aunque el LA con $\alpha =$ $\beta = 0.1$ muestras resultados similares, se decidió tomar el primero como el mejor ya que anteriormente también fue considerado el más eficiente. En el escenario 2 el algoritmo QL con $\alpha = 0.8$, $\gamma = 0.1$, $\varepsilon = 0.2$ es el que brinda un mejor resultado, mientras que en el escenario 3 es el QL con $\alpha = 0.9$, $\gamma = 0.1$, $\varepsilon = 0.2$, pues contiene un rango promedio de 2.47, lo que constituye el menor de todos.

Tabla 1 Promedio de trabajos procesados por algoritmo y escenario.

Algoritmos y parámetros	Escenario 1	Escenario 2	Escenario 3
	% promedio de trabajos procesados	**% promedio de trabajos procesados**	**% promedio de trabajos procesados**
LA-0.3-0.1	**95,8417163**	96,22178	98,6019931
LA-0.1-0.1	95,7155701	96,81898	98,7710197
LA-0.1-0.0	76,3279984	85,74716	95,2315647
QL-0.9-0.1-0.2	92,8891818	98,39792	99,19530067
QL-0.8-0.1-0.1	92,320968	98,32991	99,1764605
QL-0.8-0.1-0.2	91,8365638	98,15031	**99,3217521**
QL-0.9-0.1-0.1	91,4412211	**98,42221**	99,256839

La comparación en cada escenario mediante la prueba de *Wilcoxon* dos a dos, se realizó teniendo en cuenta el mejor algoritmo en cada caso, comparándolo así con cada combinación posible de algoritmo. En el escenario 1 existen diferencias significativas entre su mejor algoritmo y los algoritmos LA02, QL01, QL02, QL03 y QL04, mientras que no hay diferencias con respecto al LA03. En el caso del escenario 2 y el escenario 3 no es notoria la diferencia entre el algoritmo QL con sus posibles combinaciones, contrariamente sucede con el algoritmo QL y LA, donde se comprueban las diferencias significativas existentes.

Tabla 2. Prueba de Friedman. Escenario 1

	Rango Promedio
LA01	**1.87**
LA02	7.00
LA03	**1.87**
QL01	4.57
QL02	3.67
QL03	4.77
QL04	4.27
N	**15**
Chi-cuad.	**61.840**
gl	**6**
Sig.	**0.000**

Tabla 3. Prueba de Friedman. Escenario 2

	Rango Promedio
LA01	4.83
LA02	7.00
LA03	5.23
QL01	3.57
QL02	2.47
QL03	**2.37**
QL04	2.53
N	**15**
Chi-cuad.	**59.914**
gl	**6**
Sig.	**0.000**

Tabla 4. Prueba de Friedman. Escenario 3

	Rango Promedio
LA01	5.07
LA02	6.73
LA03	4.63
QL01	**2.47**
QL02	3.43
QL03	2.57
QL04	3.10
N	**15**
Chi-cuad.	**48.251**
gl	**6**
Sig.	**0.000**

6. CONCLUSIONES

En este trabajo se adaptaron dos algoritmos de Aprendizaje Reforzado, QL y LA, para dar solución a un problema real relacionado con una planta de productos químicos. Se seleccionaron tres posibles escenarios, a los cuales se le aplicaron distintas pruebas y se arribaron a conclusiones.

Para todos los escenarios el algoritmo LA con esquema de actualización L_{R-I} no muestra un buen desempeño, esto se debe a que este esquema no penaliza las malas acciones, es decir, no toma ninguna decisión cuando una acción de este tipo es tomada.

De los algoritmos implementados, el QL es capaz de procesar una mayor cantidad de trabajos del total de trabajos generados cuando el escenario se torna más complejo (mayor cantidad de máquinas y/o etapas), esto se debe a que el QL utiliza valores más flexibles para asignar la recompensa a diferencia del algoritmo LA que solo da recompensas de 0 o 1.

RECEIVED: JULY, 2016
REVISED: DECEMBER, 2016

BIBLIOGRAFIA

[1] ALBERS, S. (1999): Better bounds for online scheduling. **SIAM Journal on Computing**, 29, 459-473.
[2] ALBERS, S. and SCHRÖDER, B. (2002): An experimental study of online scheduling algorithms. **Journal of Experimental Algorithmics** , 7, 3.
[3] ARVIV, K., STERN, H. and EDAN, Y. (2016): Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem. **International Journal of Production Research**, 54, 1196-1209.
[4] BARTAL, Y., FIAT, A., KARLOFF, H. and VOHR, R. (1995): New algorithms for an ancient scheduling problem. **Journal of Computing and System Sciences**, 359–366.
[5] BŁAŻEWICZ, J., ECKER, K. H., PESCH, E., SCHMIDT, G. and WEGLARZ, J. (2013): **Scheduling computer and manufacturing processes.** Springer Science & Business Media, Berlin.
[6] BUSH, R. R. and MOSTELLER, F. (1955): Stochastic models for learning. Oxford, England, xvi, 365.
[7] CASTILLO, I. and ROBERTS, C. A. (2001): Real-time control/scheduling for multi-purpose batch plants. **Computers & industrial engineering,** 41, 211-225.
[8] CORREA, J. E. R. and WAGNER, M. R. (2009): LP-based online scheduling: from single to parallel machines. **Mathematical Programming**, 119, 109-136.
[9] GABEL, T. and RIEDMILLER, M. (2012): Distributed policy search reinforcement learning for job-shop scheduling tasks. **International Journal of Production Research**, 50, 41-61.
[10] GELFOND, M. and KAHL, Y. (2014): **Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach**. Cambridge University Press, Cambridge, England.
[11] JENNINGS, N. R., SYCARA, K. and WOOLDRIDGE, M. (1998): A roadmap of agent research and development. **Autonomous agents and multi-agent system**s, 1, 7-38.
[12] KARGER, D. R., PHILLIPS, S. J. and TORNG, E. (1996): A Better Algorithm for an Ancient Scheduling Problem. **Journal of Algorithms**, 20, 400-430.
[13] MAO, W., KINCAID, R. K. and RIFKIN, A. (1995): On-line algorithms for a single machine scheduling proble*m*. In S. G. N. a. A. Sofer, ed., **The impact of Emerging Technologies on Computer Science and Operations Research**, Springer US, Boston.
[14] MICHAEL, P. (20015): **Scheduling**, Springer, Berlin.
[15] PEETERS, M. (2008): **Solving Multi-Agent Sequential Decision Problems Using Learning Automata.** Vrije Universiteit Brussel.
[16] SIMON, H. A. and LEA, G. (1974)l: Problem solving and rule induction: A unified view. In L. W. Gregg, ed., **Knowledge and cognition**, 105-127l. Lawence Erlbaum, Oxford, England.
[17] WATKINS, C. and DAYAN, P. (1992): Technical note: Q-Learning. **Mahine Learning**, 8, 3-4.
[18] ZHANG, T. and LI, J. (2015): Online task scheduling for LiDAR data preprocessing on hybrid GPU/CPU devices: A reinforcement learning approach. **Online task scheduling for LiDAR data preprocessing on hybrid GPU/CPU devices: A reinforcement learning approach**, 8, 386-397.