REVISTA CUBANA
DE TRANSFORMACIÓN DIGITAL

RNPS 2487
Vol. 1 • Nro. 1 • enero-abril 2020 • págs. 108-118

ARTÍCULO ORIGINAL

Herramienta basada en aprendizaje reforzado y sistemas multi-agente para problemas de secuenciación de tareas

A Tool based on Reinforcement Learning and Multi-Agent Systems for Scheduling Problems

Jessica Coto Palacio
jcotopalacio@gmail.com • https://orcid.org/0000-0002-1977-145X
UEB HOTEL LOS CANEYES, VILLA CLARA, CUBA

Yailen Martínez Jiménez
yailenm@uclv.edu.cu • https://orcid.org/0000-0002-1223-0589
UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS, CUBA

Ann Nowé
ann.nowe@ai.vub.ac.be • https://orcid.org/0000-0001-6346-4564
VRIJE UNIVERSITEIT BRUSSEL, BÉLGICA

Recibido: 2019-12-24 • Aceptado: 2020-03-31

RESUMEN

La aparición de la Industria 4.0 permite que nuevos enfoques puedan resolver problemas industriales como la secuenciación de tareas de tipo *job-shop.* Se ha demostrado que los enfoques basados en Aprendizaje reforzado con múltiples agentes son altamente prometedores para manejar complejos escenarios de secuenciación. En este trabajo se propone una herramienta basada en Aprendizaje reforzado y Sistemas multi-agente la cual cuenta con una interfaz amigable e intuitiva, más atractiva para la Industria. Permite a los usuarios interactuar con los algoritmos de aprendizaje de manera tal que todas las restricciones de la planta de producción se incluyan cuidadosamente y los objetivos se puedan adaptar a los escenarios del mundo real. El usuario puede mantener la mejor solución obtenida por un algoritmo *Q-Learning* o ajustarla fijando algunas operaciones para cumplir con ciertas restricciones, luego la herramienta optimizará la solución modificada respetando las preferencias del usuario utilizando dos alternativas posibles. Estas alternativas se validan a través de juegos de datos de la librería de problemas de investigación de operaciones (*OR-Library).*

Palabras clave: *secuenciación de tareas; sistemas multi-agente; industria 4.0; aprendizaje reforzado.*

ABSTRACT

The emergence of Industry 4.0 allows for new approaches to solving industrial problems such as the Job Shop Scheduling Problem. It has been demonstrated that Multi-Agent Reinforcement Learning approaches are highly promising to handle complex scheduling scenarios. In this work we propose a user-friendly Multi-Agent Reinforcement Learning tool, more appealing for industry. It allows the users to interact with the learning algorithms in such a way that all the constraints on the production floor are carefully included and the objectives can be adapted to the real world scenarios. The user can either keep the best schedule obtained by a Q-Learning algorithm or adjust it by fixing some operations in order to meet certain constraints, then the tool will optimize the modified solution respecting the user preferences using two possible alternatives. These alternatives are validated using OR-Library benchmarks, the experiments show that the modified Q-Learning algorithm can obtain the best results.

Keywords: *scheduling problems; multi-agent systems; industry 4.0; reinforcement learning.*

INTRODUCCIÓN

Durante los últimos años los avances tecnológicos han beneficiado cada vez más el rendimiento de la Industria. La aparición de las nuevas tecnologías de la información ha dado lugar a fábricas inteligentes en lo que se denomina Industria 4.0 (i4.0) (Kuhnle, Röhrig & Lanza, 2019; Leitao, Colombo, & Karnouskos, 2016; Leitao, Rodrigues, Barbosa, Turrin, & Pagani, 2005). La Revolución i4.0 implica la combinación de sistemas inteligentes y adaptables que utilizan el conocimiento compartido entre diversas plataformas heterogéneas para la toma de decisiones computacionales (Leitao, *et al.,* 2016; Leusin, Frazzon, Uriona Maldonado, Kück, & Freitag, 2018; Vogel-Heuser, Lee, & Leitao, 2015). En este sentido, el uso de Sistemas Multi-Agente (SMA) constituye un enfoque prometedor para manejar problemas complejos y dinámicos (Leusin, *et al.,* 2018; Palombarini & Martínez, 2019). Un ejemplo típico de una oportunidad industrial de este tipo es la secuenciación de tareas (*scheduling),* cuyo objetivo es lograr la optimización de los recursos y la minimización del tiempo total de ejecución de las tareas (Shi, *et al.,* 2020; Toader, 2017).

Dada la complejidad y dinamismo de los entornos industriales, la resolución de este tipo de problemas puede implicar el uso de alternativas de solución muy complejas, ya que hay que

ejecutar los pedidos de los clientes, y cada pedido está compuesto por una serie de operaciones que deben ser procesadas en los recursos o máquinas disponibles. En los problemas de secuenciación del mundo real, el entorno es tan dinámico que normalmente no se conoce de antemano toda esta información. Por ejemplo, la secuenciación de tareas en la industria está sujeta a una incertidumbre constante, las máquinas pueden averiarse, los pedidos pueden tardar más de lo previsto, y estos acontecimientos inesperados pueden hacer que la planificación original falle (Hall & Potts, 2004; Xiang & Lee, 2008).

En consecuencia, el problema de crear una secuenciación de tipo *job-shop,* conocido como *Job-Shop Scheduling Problem* (JSSP), se considera uno de los problemas de la industria más difíciles de la literatura (Asadzadeh, 2015). Muchos problemas de secuenciación sugieren una formulación natural como tareas de toma de decisiones distribuidas, de ahí que el empleo de sistemas multi-agente represente un enfoque evidente (Gabel, 2009). Estos agentes suelen utilizar Aprendizaje reforzado (RL por sus siglas en inglés), que consiste en aprender qué hacer (cómo relacionar las situaciones con las acciones) para maximizar una señal numérica de recompensa (Sutton & Barto, 1998). Permite que un agente aprenda un comportamiento óptimo a través de interacciones a prueba y error con su entorno. Al probar acciones repetidamente en situaciones diferentes, el agente puede descubrir las consecuencias de su comportamiento e identificar la mejor acción para cada situación. Por ejemplo, cuando se trata de eventos inesperados, los métodos de aprendizaje pueden jugar un papel importante, ya que podrían "aprender" de resultados anteriores y cambiar los parámetros de las siguientes iteraciones, permitiendo obtener no sólo buenas soluciones, sino también soluciones más robustas.

Otro problema que ha sido identificado en la comunidad de secuenciación de tareas es el hecho de que la mayoría de las investigaciones se concentran en problemas de optimización que son una versión simplificada de la realidad. Como señala el autor en (Urlings, 2010): "Esto permite el uso de enfoques sofisticados y garantiza en muchos casos la obtención de soluciones óptimas. Sin embargo, la exclusión de las restricciones del mundo real perjudica la aplicabilidad de esos métodos. Lo que la industria necesita son sistemas para optimizar la secuenciación de la producción que se ajusten exactamente a las condiciones de la planta de producción y que generen buenas soluciones en muy poco tiempo". En esta investigación proponemos una herramienta basada en Aprendizaje reforzado y Sistemas multi-agente, que permite al usuario mantener el mejor resultado obtenido por un algoritmo de aprendizaje o incluir restricciones adicionales de la planta de producción. Esta primera versión permite fijar operaciones a intervalos de tiempo en los recursos correspondientes y optimizar posteriormente la solución en función de las nuevas restricciones añadidas por el usuario. Este es un primer enfoque que ayuda a cerrar la brecha existente entre la literatura y la práctica.

REVISIÓN DE LA LITERATURA

Como se ha mencionado anteriormente, la secuenciación de tareas es un proceso de toma de decisiones que tiene que ver con la asignación de recursos limitados (máquinas, equipos de

manipulación de materiales, operadores, herramientas, etc.) a tareas competitivas (operaciones de trabajos) a lo largo del tiempo con el fin de optimizar uno o más objetivos (Pinedo, 1995). La salida de este proceso son las asignaciones de tiempo/máquina/operación (Goren & Sabuncuoglu, 2008). La secuenciación es considerada como uno de los problemas clave en los sistemas de manufactura, y ha sido un tema de interés durante mucho tiempo. Sin embargo, es difícil hablar de un método que ofrezca soluciones óptimas para cada problema que surja (Aydin & Oztemel, 2000).

Se han aplicado diferentes técnicas de Investigación de Operaciones (IO) (Programación lineal, Programación entera mixta, etc.) a problemas de secuenciación. Estos enfoques suelen implicar la utilización de un modelo que contiene una función objetivo, un conjunto de variables y un conjunto de restricciones. Las técnicas basadas en IO han demostrado la capacidad de obtener soluciones óptimas para los problemas bien definidos, pero las soluciones se limitan a modelos estáticos. Los enfoques de la Inteligencia artificial, por otro lado, proporcionan más representaciones de problemas del mundo real, permitiendo que la experiencia humana esté presente en el ciclo (Gomes, 2000).

Job Shop Scheduling

Un conocido problema de secuenciación de la industria manufacturera es el clásico JSSP (Zhang, 1996), que implica un conjunto de trabajos y un conjunto de máquinas con el propósito de buscar la mejor secuenciación, es decir, una asignación de las operaciones a intervalos de tiempo en las máquinas que tiene la duración mínima necesaria para completar todos los trabajos (conocido en la literatura como *makespan).* El número total de soluciones posibles para un problema con *n* trabajos y *m* máquinas es *m(n!).* En este caso, los métodos de optimización exactos no proporcionan soluciones oportunas. Por lo tanto, debemos dirigir nuestra atención a los métodos de búsqueda que pueden producir soluciones satisfactorias (pero no necesariamente óptimas) (Martínez Jiménez, 2012). Algunas de las restricciones inherentes en la definición del JSSP son las siguientes:

- Sólo se puede procesar simultáneamente una operación de cada trabajo.
- No se permite la anticipación (es decir, la interrupción del proceso) de las operaciones.
- Cada trabajo debe ser procesado hasta su finalización y ningún trabajo es procesado dos veces en la misma máquina.
- Los trabajos pueden iniciarse y terminarse en cualquier momento, es decir, no existen fechas de liberación o de terminación.
- Las máquinas no pueden procesar más de una operación a la vez.
- Sólo hay una máquina de cada tipo y pueden estar inactivas dentro del período de secuenciación.
- Los trabajos deben esperar a que la siguiente máquina en el orden de procesamiento esté disponible.
- El orden de procesamiento de cada trabajo se conoce de antemano y es inmutable.

La Investigación de operaciones ofrece diferentes enfoques matemáticos para resolver problemas de secuenciación, por ejemplo, Programación lineal, Programación dinámica y Méto-

dos de ramas y cotas. Cuando el tamaño del problema no es demasiado grande, estos métodos pueden proporcionar soluciones óptimas en un tiempo razonable. La mayoría de los problemas de secuenciación del mundo real son *NP-hard,* y el tamaño no suele ser pequeño, es por eso que los métodos de optimización no proporcionan soluciones óptimas en un tiempo razonable. Aquí es donde los métodos heurísticos se convierten en el centro de atención, estos métodos pueden obtener buenas soluciones de una manera eficiente. La Inteligencia artificial se convirtió en una herramienta importante para resolver problemas de secuenciación del mundo real a principios de los 80 (Zhang, 1996).

En (Gabel, 2009; Gabel & Riedmiller, 2007), los autores sugirieron y analizaron la aplicación de técnicas de aprendizaje reforzado para resolver problemas de secuenciación de tareas. Demostraron que interpretar y resolver este tipo de escenarios como un problema de aprendizaje multi-agente es de beneficio para obtener soluciones casi óptimas y puede competir muy bien con los enfoques de soluciones alternativas.

Aprendizaje reforzado con múltiples agentes

El paradigma de Aprendizaje reforzado es una forma popular de abordar problemas que sólo tienen una retroalimentación ambiental limitada, en lugar de ejemplos correctamente etiquetados, como es común en otros contextos de aprendizaje automático (Taylor & Stone, 2009). Existen muchos enfoques posibles para aprender políticas de comportamiento, por ejemplo, métodos de Diferencia temporal, como *Q-Learning* (QL) (Singh & Sutton, 1996; Watkins, 1989) y *Sarsa* (Gavin & Niranjan, 1994; Singh & Sutton, 1996), métodos de búsqueda de políticas, como iteración de políticas (programación dinámica), gradiente de políticas (Baxter & Bartlett, 2001; Williams, 1992), y búsqueda directa de políticas (Y. Ng & Jordan, 2000), entre otros. La idea general detrás de ellos es aprender a través de la interacción con un entorno y los pasos pueden resumirse de la siguiente manera:

1. El agente percibe un estado de entrada.
2. El agente determina una acción utilizando una función de toma de decisiones (política).
3. Se ejecuta la acción seleccionada.
4. El agente obtiene una recompensa escalar de su entorno (refuerzo).
5. Se procesa la información sobre la recompensa que se ha recibido por haber tomado la acción reciente en el estado actual.

El paradigma básico de RL es aprender la relación de estados con acciones sólo en base a las recompensas que el agente recibe de su entorno. Ejecutando repetidamente las acciones y observando las recompensas resultantes, el agente trata de mejorar y afinar su política. El Aprendizaje reforzado es considerado como un método robusto para el aprendizaje en entornos de sistemas multi-agente, área de investigación en rápido crecimiento que unifica ideas de varias disciplinas, incluyendo la Inteligencia artificial, Ciencias de la computación, Ciencias cognitivas, Sociología y Ciencias de la gestión. Recientemente, ha habido un considerable interés en el campo motivado por el hecho de que muchos problemas del mundo real como el Diseño de ingeniería, la Búsqueda inteligente, el Diagnóstico médico y la Robótica pueden

modelarse mejor utilizando un grupo de solucionadores de problemas en lugar de uno, cada uno de ellos denominado agente (Stone & Veloso, 2000).

HERRAMIENTA BASADA EN APRENDIZAJE REFORZADO Y SISTEMAS MULTI-AGENTE

La herramienta propuesta agrupa varios algoritmos destinados a resolver problemas de secuenciación en la industria manufacturera. Este trabajo propone una primera versión de una herramienta que se centra en la necesidad de construir una secuencia más flexible, con el fin de ajustarlo a las peticiones del usuario sin violar las restricciones del escenario JSSP. La figura 1 muestra la interfaz principal, donde el usuario debe elegir primero el archivo donde se describe la información relacionada con el problema, básicamente los trabajos que necesitan ser procesados, los recursos disponibles para ejecutarlos y los tiempos de procesamiento (botón de abrir). Los algoritmos originales se basan en la resolución del JSSP.

El enfoque utilizado para obtener la solución original que el usuario puede modificar posteriormente es el propuesto en (Martínez Jiménez, 2012), es un enfoque genérico de aprendizaje reforzado con múltiples agentes que puede adaptarse fácilmente a diferentes problemas de secuenciación de tareas, como el *Job Shop Flexible* (FJSSP) (Gavin & Niranjan, 1994) o el *Parallel Machines Job Shop Scheduling* (PMJSSP) (Williams, 1992). El algoritmo utilizado es el *Q-Learning,* que funciona aprendiendo una función de valor-acción que da la utilidad esperada de tomar una acción dada en un estado determinado. Básicamente existe un agente por máquina que se encarga de asignar las operaciones que deben ser ejecutadas por el recurso correspondiente. La figura 2 muestra los agentes en un entorno de secuenciación, y los parámetros a la izquierda de la interfaz principal se explican en detalle en (Martínez Jiménez, 2012).

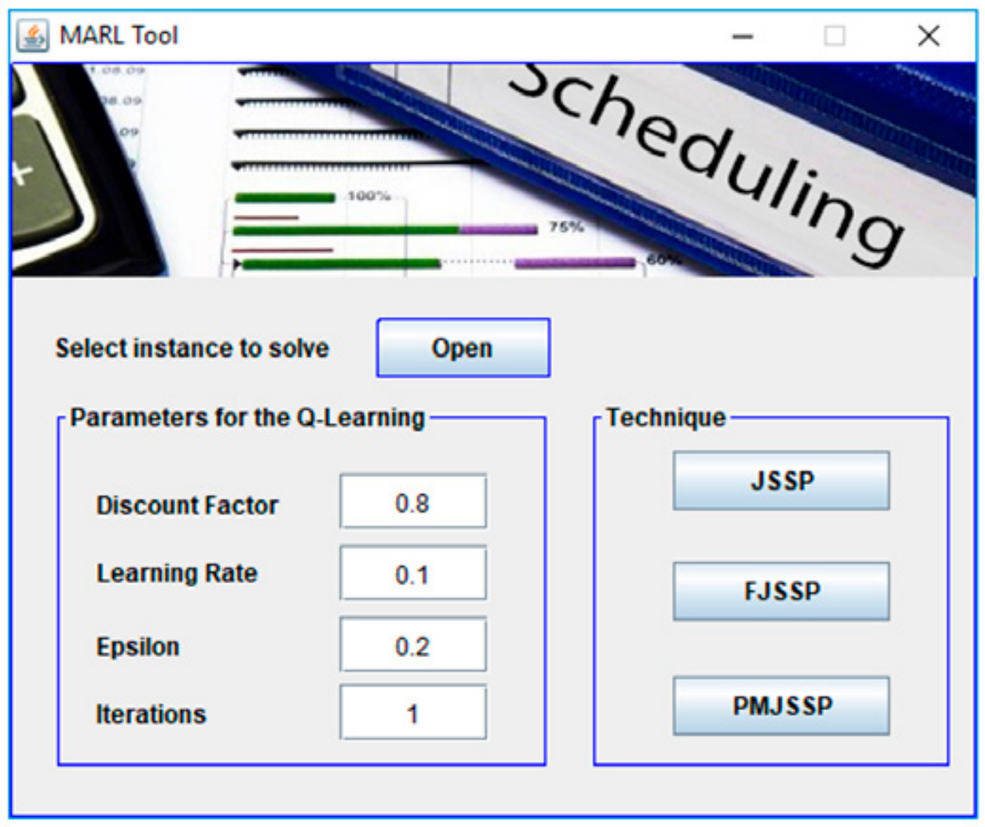

Figura 1. Interfaz principal de la herramienta.

Una vez que el usuario elige el escenario a resolver (JSSP, FJSSP o PMJSSP), la herramienta propone una solución inicial como la mostrada en la figura 3, basada en el algoritmo QL original descrito anteriormente, y al mismo tiempo habilita un conjunto de opciones que son la base de esta investigación. El

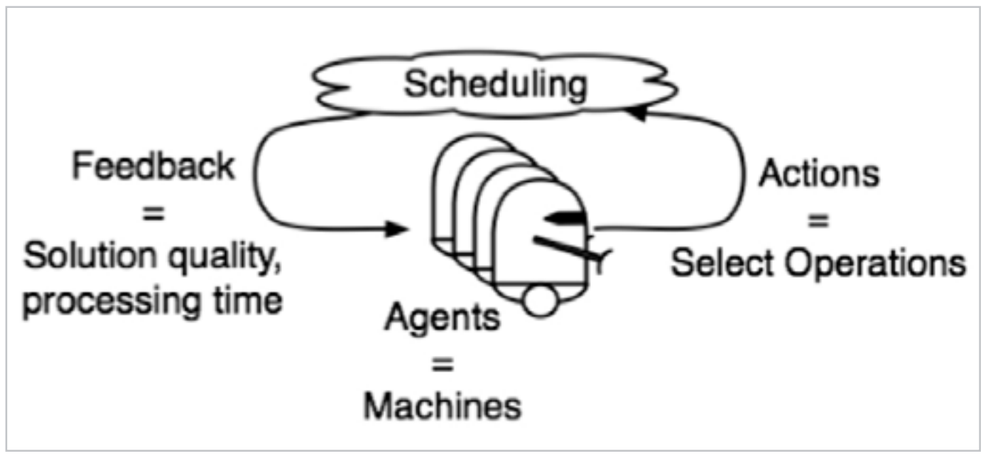

Figura 2. Agentes en un ambiente de secuenciación de tareas, propuesta en (Martínez Jiménez, 2012).

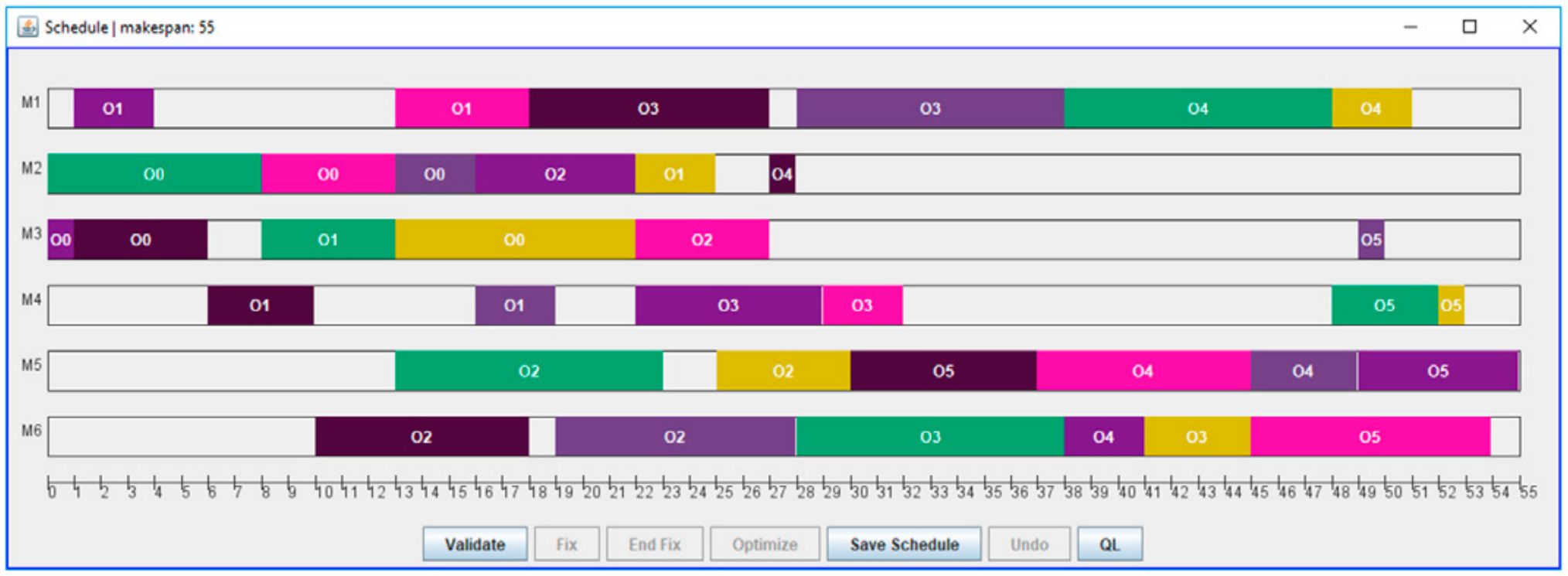

Figura. 3. Ejemplo de una secuencia obtenida usando la herramienta MARL para la instancia ft06.

usuario tiene la posibilidad de desplazar las operaciones mediante el *mouse* o la pantalla táctil, y estos movimientos deben ser validados una vez decididas las nuevas posiciones.

Las opciones de la herramienta se explican en detalle a continuación:

- **Save Schedule:** permite guardar la secuenciación de tareas como una imagen (.png) a través de un cuadro de diálogo para elegir la ruta y especificar el nombre del archivo.
- **Validate:** una vez que una operación se mueve de su posición original, la nueva secuencia debe ser validada con un desplazamiento a la derecha o a la izquierda, para que luego la herramienta permita realizar nuevos cambios. Si se aumenta el tiempo de inicio de una operación (se desplaza a la derecha), se verifica el tiempo de inicio de la siguiente operación del mismo trabajo y si se inicia antes de la nueva hora final de la operación anterior, entonces se deben realizar ajustes en la solución. Como consecuencia, lo primero es aspirar a ubicar esa operación justo después de su predecesora, en caso de que la nueva colocación obstruya el procesamiento de otra operación en el mismo recurso, la nueva hora de inicio se convierte en la hora final de esa otra operación, y así sucesivamente, se analizan las posibles ubicaciones hasta que se encuentre un lugar disponible. El desplazamiento hacia la izquierda se produce de forma similar con la excepción de que la operación se coloca de tal forma que su ejecución comienza antes. El algoritmo siempre comprueba que se trata de un movimiento válido, es decir, que no se inicia antes de la hora de inicio mínima posible para esa operación. En cuanto a las siguientes operaciones del mismo trabajo, el algoritmo intenta acercarlas lo más posible a su predecesora, con el fin de minimizar el *makespan*.
- **Fix:** esta opción se habilita una vez que se valida la nueva secuencia, con el fin de optimizarla posteriormente con los nuevos cambios. Las operaciones fijas se resaltan en negro y existe la posibilidad de volver a pulsarlas para dejar de fijar su posición.
- **End Fix:** el usuario tiene que elegir esta opción una vez finalizado el proceso de fijación de las operaciones, y luego proceder a optimizar la secuencia, ya sea utilizando los corrimientos a la izquierda o utilizando el algoritmo *Q-Learning*.

- **Optimize:** después de fijar las operaciones que el usuario desea mantener en las posiciones especificadas, se puede optimizar el resto de la secuencia. Esto se basa en la realización de un corrimiento a la izquierda de todas las operaciones movibles, respetando las restricciones de la secuenciación de tareas de tipo *job-shop* y también las horas de inicio de las operaciones fijas. El procedimiento se realiza según la posición de las operaciones en el eje x, en orden creciente según sus tiempos de inicio. Cuando se selecciona una operación diferente a la primera de cada trabajo, su nuevo tiempo inicial será el tiempo final de su predecesora, si este no es un movimiento válido porque interfiere con la ejecución de otra operación que se está procesando en la misma máquina, entonces se desplaza al primer intervalo disponible donde quepa en ese recurso, y si no es posible, mantiene su posición original.
- **Q-Learning:** esta optimización se basa en la aplicación del algoritmo QL descrito anteriormente, incluyendo una nueva restricción, en este caso el algoritmo aprenderá una secuencia teniendo en cuenta las operaciones que fueron fijadas por el usuario.
- **Undo:** es posible retroceder tantas secuencias como validaciones se hayan realizado.

En este trabajo se compara el rendimiento de las dos alternativas para optimizar las secuencias una vez que el usuario ha fijado algunas operaciones, el clásico desplazamiento a la izquierda que se ejecuta al pulsar el botón *"Optimize"* y la versión modificada del algoritmo *Q-Learning,* que incluye la posición de las operaciones fijas en el proceso de aprendizaje.

RESULTADOS EXPERIMENTALES

Para medir el rendimiento de las dos alternativas se utilizaron varios problemas de referencia de la Biblioteca de investigación de operaciones (*OR-Library)* (Beasley, 1990), la cual es una biblioteca que incluye juegos de datos para varios problemas de Investigación de operaciones. La tabla 1 muestra los resultados de once instancias de tipo JSSP usando como objetivo el *makespan,* con diferentes números de trabajos y máquinas.

Tabla 1. Resultados experimentales usando instancias de la librería OR.

Instancia	Óptimo	QL Original	Optimizar	QL con operaciones fijas
ft06	55	55	82	76
la01	666	666	849	810
la02	655	667	848	801
la03	597	610	752	730
la04	590	611	647	640
la05	593	593	603	603
la06	926	926	1012	1008
la07	890	890	1016	1010
la08	863	863	1060	1043
la09	951	951	1144	1096
la10	958	958	1016	1016

La columna "óptimo" representa la mejor solución conocida para la instancia correspondiente; "QL original" se refiere a la mejor solución obtenida por la versión original del algoritmo QL sin ninguna restricción adicional. Para los resultados mostrados en las dos últimas columnas se hicieron algunas modificaciones a la solución obtenida por el QL original, para cada caso se fijaron las mismas operaciones, y cada alternativa de optimización tuvo que ajustar la secuenciación para minimizar el tiempo de espera. Para determinar si hay diferencias significativas en los resultados obtenidos por las alternativas, se realizó un análisis estadístico y los resultados se muestran en la figura 4.

Rangos

		N	Mean Rank	Sum of Ranks
QL_modif - Optimizar	Rangos Negativos	9[a]	5,00	45,00
	Rangos Positivos	0[b]	,00	,00
	Empates	2[c]		
	Total	11		

a. QL_modif < Optimizar, b. QL_modif > Optimizar, c. QL_modif = Optimizar

Test Statistics[a]

	QL_modified - Optimize
Z	-2,668[b]
Asymp. Sig. (2-tailed)	,008

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

Figura 4. Análisis estadístico usando el Test de Wilcoxon.

Como se puede apreciar, la prueba de Wilcoxon muestra que hay diferencias significativas entre las dos alternativas (sig=0.008), los rangos medios confirman que la versión QL con operaciones fijas es capaz de obtener mejores resultados que el proceso clásico de optimización del desplazamiento de las operaciones (optimizar). Esto se debe principalmente a que el desplazamiento a la izquierda respeta el orden en el que las operaciones se ubicaron inicialmente a lo largo del eje *x*. El algoritmo QL, por otro lado, mantiene las posiciones fijas, y durante el proceso de aprendizaje el orden en que se secuencian las operaciones en los recursos no tiene que ser el mismo, esto permite que el enfoque obtenga mejores soluciones en términos de *makespan* o tiempo de completamiento de las tareas.

CONCLUSIONES

Este trabajo propone una herramienta basada en Aprendizaje reforzado y Sistemas multi-agente para el problema de secuenciación tipo *job-shop,* que puede ser adaptada a otros escenarios

de secuenciación como el JSSP flexible y el JSSP con máquinas paralelas. Esta herramienta permite al usuario mantener la mejor secuencia obtenida por el algoritmo QL original o realizar ajustes para mover las operaciones a intervalos fijos, de acuerdo con las restricciones de la planta de producción. Una vez realizados todos los ajustes, se inicia un nuevo proceso de secuenciación para optimizar al máximo la solución modificada. Esta optimización puede realizarse desplazando a la izquierda todas las posibles operaciones móviles o utilizando una versión modificada del algoritmo QL. Las alternativas se evaluaron utilizando datos de referencia de la biblioteca OR y los resultados mostraron que el algoritmo QL modificado es capaz de obtener los mejores resultados.

REFERENCIAS

Asadzadeh, L. (2015). A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering,* 85, 376–383.

Aydin, M. E., & Oztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems,* 33, 169–178.

Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research,* 15, 319–350.

Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society,* 41(11), 1069–1072.

Gabel, T. (2009). *Multi-Agent Reinforcement Learning Approaches for Distributed Job-Shop Scheduling Problems.* PhD Thesis. Universität Osnabrück.

Gabel, T., & Riedmiller, M. (2007). Scaling Adaptive Agent-Based Reactive Job-Shop Scheduling to Large-Scale Problems. *IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007),* 259–266.

Gavin, R., & Niranjan, M. (1994). *Online Q-learning using connectionist systems.* Technical Report. Cambridge University, Engineering Department.

Gomes, C. P. (2000). Artificial intelligence and operations research: challenges and opportunities in planning and scheduling. *The Knowledge Engineering Review, 15*(1), 1–10.

Goren, S., & Sabuncuoglu, I. (2008). Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions, 40*(1), 66–83.

Hall, N. G., & Potts, C. N. (2004). Rescheduling for new orders. *Operations Research,* 52, 440–453.

Kuhnle, A., Röhrig, N., Lanza, G. (2019). Autonomous order dispatching in the semiconductor industry using reinforcement learning. *Procedia CIRP,* Volume 79, Pages 391-396, ISSN 2212-8271.

Leitao, P., Colombo, A. W., & Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers Industry, 81,* 11–25.

Leitao, P., Rodrigues, N., Barbosa, J., Turrin, C., & Pagani, A. (2005). Intelligent products: The grace experience. *Control Engineering Practice, 42,* 95–105.

Leusin, M. E., Frazzon, E. M., Uriona Maldonado, M., Kück, M., & Freitag, M. (2018). Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era. *Technologies, 6*(4).

Martínez Jiménez, Y. (2012). *A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems.* PhD Thesis. Vrije Universiteit Brussel, Brussels.

Palombarini, J. A., Martínez, E. C. (2019). Closed-loop Rescheduling using Deep Reinforcement Learning. IFAC-PapersOnLine, Volume 52, Issue 1, Pages 231-236, ISSN 2405-8963.

Pinedo, M. (1995). Scheduling: theory, algorithms and systems. *Englewood Cliffs, NJ: PrenticeHall.*

Shi D., Fan W., Xiao Y., Lin T., Xing C. (2020). Intelligent scheduling of discrete automated production line via deep reinforcement learning. International Journal of Production Research.

Singh, S., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning, 22,* 123–158.

Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots, 8*(3), 345–383.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* The MIT Press.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research, 10,* 1633–1685.

Toader, F. A. (2017). *Production Scheduling in Flexible Manufacturing Systems: A State of the Art Survey. 3*(7), 1–6.

Urlings, T. (2010). *Heuristics and metaheuristics for heavily constrained hybrid flowshop problems.* Universidad Politécnica de Valencia.

Vogel-Heuser, B., Lee, J., & Leitao, P. (2015). Agents enabling cyber-physical production systems. *AT-Autom., 63,* 777–789.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards.* PhD Thesis. King's College.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8,* 229–256.

Xiang, W., & Lee, H. P. (2008). Ant colony intelligence in multi-agent dynamic manufacturing scheduling. *Engineering Applications of Artificial Intelligence, 21,* 73–85.

Y. Ng, A., & Jordan, M. (2000). PEGASUS: apolicy search method for large MDPs and POMDPs. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence.*

Zhang, W. (1996). Reinforcement Learning for Job Shop Scheduling. PhD Thesis. Oregon State University.

Copyright © 2020 Coto-Palacios, J., Martinez-Jimenez, Y., Nowe, A.

Este obra está bajo una licencia de Creative Commons Reconocimiento 4.0 Internacional.