Aplicación de algoritmos de aprendizaje automático a la programación de órdenes de producción en talleres de trabajo: una revisión de la literatura reciente

Efraín Pérez-Cubero y Raúl Poler

Recibido: 3 de Septiembre de 2020
Aceptado: 3 de Noviembre de 2020

https://doi.org/10.37610/dyo.v0i72.588

Resumen

El propósito de este artículo es identificar y analizar los algoritmos de aprendizaje automático propuestos recientemente, para dar solución al problema de programación y secuenciación de órdenes de producción en talleres de trabajo, a través de una revisión sistemática de las últimas publicaciones realizadas en este campo. Este estudio pretende proporcionar a los investigadores un punto de partida para la selección de los algoritmos de aprendizaje automático más prometedores para la solución de este tipo de problema, identificar las áreas de investigación más interesantes en la actualidad y proponer líneas de investigación futuras.

Palabras clave

Taller de trabajo, programación, aprendizaje automático, aprendizaje reforzado, aprendizaje profundo.

1. Introducción

Este articulo presenta una revisión bibliográfica de las investigaciones publicadas en los últimos años sobre el uso de algoritmos de aprendizaje automático (Machine Learning – ML). En concreto el aprendizaje reforzado (Reinforced Learning - RL), aprendizaje profundo (Deep Learning - DL) y aprendizaje reforzado profundo (Deep Reinforced Learning - DRL) como herramientas para la solución del problema de programación del taller de trabajo (Job Shop Scheduling Problem - JSSP).

El objetivo de este estudio es establecer el estado actual de la investigación y proponer directrices útiles para la investigación y el desarrollo de este tipo de algoritmos en dicha área. Para la realización de este estudio se establecieron las siguientes preguntas de investigación:

PI_1: ¿Cuál es el estado actual de las investigaciones realizadas en la solución al problema de programación del taller de trabajo utilizando DL, RL o NN?

PI_2: ¿Cuáles son las propuestas de la literatura para la solución del problema de programación del taller de trabajo utilizando DL, RL o NN?

Efraín Pérez-Cubero *
efrain.perezcubero@ucr.ac.cr
https://orcid.org/0000-0003-3257-0457

Raúl Poler **
rpoler@cigip.upv.es
https://orcid.org/0000-0003-4475-6371

* Sede Interuniversitaria Alajuela, Universidad de Costa Rica
** Centro de Investigación en Gestión e Ingeniería de Producción, Universitat Politècnica de València

PI_3: ¿Cómo se pueden caracterizar los algoritmos usados en la solución al problema de programación del taller de trabajo y sus derivados?

PI_4: ¿Qué algoritmos de los tipos DL, RL o NN se han usado en la literatura para la solución al problema de programación del taller de trabajo?

PI_5: ¿Que líneas de investigación futura se pueden plantear en base a las investigaciones existentes y que áreas de investigación se pueden orientar para futuros estudios?

El artículo se estructura de la siguiente manera: en la sección 2 se detalla la metodología de revisión aplicada; en la sección 3 se presenta una definición del problema a investigar. La sección 4 ofrece una revisión de las diferentes soluciones que se han propuesto en la literatura para solucionar el problema de programación de taller de trabajo; desde los métodos tradicionales o clásicos hasta los más vanguardistas. La sección 5 presenta un análisis descriptivo de los hallazgos encontrados en el uso de las técnicas de DL, RL y DRL como mecanismos de solución. La sección 6 se analizan las propuestas presentadas hasta la actualidad, haciendo especial énfasis en sus ventajas y limitaciones; finalmente, en la sección 7 se presentan las conclusiones y las líneas potenciales para futuras investigaciones, así como un análisis de estas.

2. Metodología de búsqueda literaria

La revisión de la literatura es parte de la estructura básica de una investigación, en este sentido el objetivo de este estudio es: analizar sistemáticamente las recientes propuestas realizadas sobre el problema de la programación

en el taller de trabajo. Se tuvo en cuenta los tipos de algoritmos utilizados para la solución, así como el tipo de problema que se resuelve, identificando sus características clave y de esta forma dando respuesta a la primera pregunta de investigación, PI_1.

Para ello se ha utilizado una adaptación de la metodología propuesta por Ulhmann y Frezzon (2018), la cual consta de cinco etapas. En la primera etapa se seleccionaron documentos de la base de Scopus, entre los años 2012 y 2020, resultantes de la siguiente búsqueda:

sequencing AND "job shop" AND ("deep learning" OR "reinforced learning")

De este modo se obtuvieron un total de 39 artículos. Los criterios de búsqueda se detallan en la Tabla 1.

Tabla 1 Criterios de búsqueda

Identificación	Descripción
Base de Datos	Scopus
Lenguaje	Inglés
Áreas Científicas	Todas
Revistas	Todas
Tipos de Artículos	Review / Artículos
Fecha de Búsqueda	Mayo 2020
Años	2012-2020

En la segunda etapa se procedió a filtrar los artículos mediante la revisión de su resumen y conclusiones. Se descartaron artículos que no estuvieran relacionados con el objeto de la investigación, mediante este filtrado se eliminaron 29 artículos reduciendo la cantidad de artículos de interés a 10.

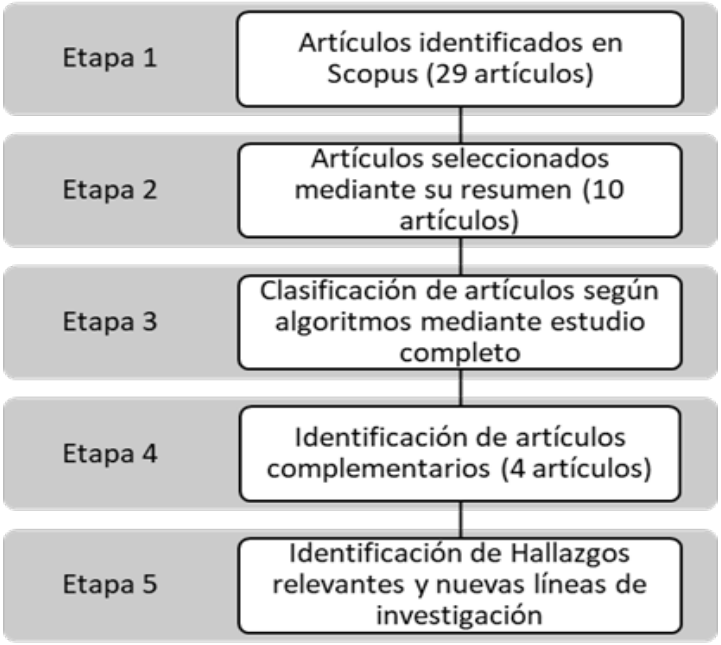
En la tercera etapa se clasificaron los artículos según el tipo de algoritmo y el enfoque de solución mediante un estudio completo de los mismos.

La cuarta etapa de búsqueda consistió en identificar otros potenciales artículos de interés a partir de las referencias bibliográficas de los artículos seleccionados originalmente. El resultado fue un total de 35 artículos adicionales. Esas nuevas fuentes fueron sometidas a las fases 2 y 3 de la metodología de búsqueda agregando 4 artículos más, para un total de 14 artículos de interés.

La quinta etapa consistió en extraer de los artículos los hallazgos significativos para el tema en estudio, las propuestas de solución planteadas y las justificaciones de nuevas líneas de investigación.

La Figura 1 presenta un resumen de la metodología seguida y los artículos analizados y seleccionados en cada etapa.

Figura 1 Metodología de búsqueda



3. Definición del Problema JSSP

Con la intención de poder avanzar adecuadamente en la investigación fue necesario adoptar una definición del problema bajo investigación. Utilizando como fuentes varios de los artículos identificados para la investigación y algunas de sus referencias, se presentan las siguientes definiciones para taller de trabajo (Job Shop) y para el problema de programación del taller de trabajo (Job Shop Scheduling Problem - JSSP).

Un taller de trabajo puede describirse como una ubicación en la que existen varias estaciones de trabajo de propósito general, las que se utilizan para realizar una variedad de tareas (Cunha et al., 2020). En tanto que el problema de programación del taller de trabajo (JSSP), esencialmente, implica completar un conjunto de trabajos con un número limitado de recursos de fabricación, bajo una serie de restricciones para optimizar una función objetivo particular (Wang y Usher, 2005).

El JSSP es uno de los problemas de programación de producción más típicos y complejos (Sharma y Jain, 2016). A este problema se han aportado diferentes propuestas de resolución, entre las que se encuentran: métodos exactos, métodos híbridos y heurísticos.

Los métodos exactos proporcionan una solución óptima, pero con un tiempo de cálculo exponencial al ser el JSSP un problema del tipo "NP- completo" (Wang y Usher, 2005) o "NP-hard" (Cunha et al., 2020) (Sharma y Jain, 2016).

Formalmente, el JSSP clásico se puede definir de la siguiente manera: n trabajos $J=\{J_1,J_2,J_3,\ldots,J_n\}$ son procesados en m máquinas o estaciones $M=\{M_1,M_2,M_3,\ldots,M_m\}$ y cada trabajo se procesa en un determinado orden de visita de las máquinas en las que debe sufrir las operaciones. Cada trabajo puede ser procesado como máximo por una máquina a la vez, y cada máquina puede procesar como máximo un trabajo a la vez. No se permite la interrupción de procesamiento de trabajos (Sharma y Jain, 2016).

Existe una diversidad de modificaciones a JSSP entre las configuraciones principales se pueden encontrar y analizar las configuraciones de: máquina única, máquinas paralelas, taller de flujo (FS), taller de flujo flexible (FFS) y taller flexible (FJS) (González-Neira et al., 2017). También es posible clasificar el problema según la metodología de solución utilizada, pudiendo ser esta de manera determinista o estocástica.

Dada la gran cantidad de variantes que puede presentar el JSSP, se ha utilizado por diferentes autores una nomenclatura para parametrizarlo de forma funcional. Esta nomenclaruta consistente en la notación estándar de tres campos α/β/γ para describir problemas de programación. El primer campo "α" describe el tipo de la planta o taller de producción, el segundo campo "β" describe las condiciones del taller, así como la información de configuración de éste. El tercer campo "γ" está reservado para medidas de rendimiento (Sharma y Jain, 2016).

La Figura 2 muestra una representación de las variantes del JSSP presentada recientemente por Zang et al. (2019).

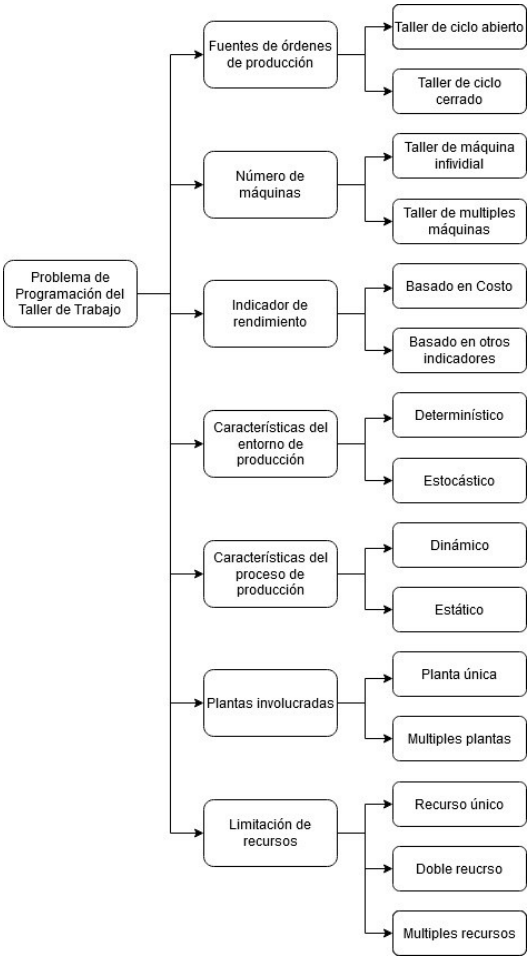


Figura 2 Variantes del JSSP.
Fuente: (Zhang et al., 2019)

4. Propuestas de la literatura para la solución del JSSP

Para dar respuesta a la segunda pregunta de investigación, PI_2, se presenta en esta sección una revisión de las propuestas de solución al JSSP. Se parte de un enfoque clásico y se avanza hasta llegar a las perspectivas relacionadas con la inteligencia artificial y el aprendizaje automático. También se muestran las definiciones de los conceptos más importantes dentro de la investigación.

El JSSP ha sido abordado históricamente desde diferentes enfoques, en la década de los noventa se presentó una recopilación de métodos separados en dos grandes grupos de soluciones: soluciones exactas y soluciones aproximadas. En el grupo de soluciones exactas se incluyeron métodos como ramificación y acotación, límite inferior, derivación y desigualdades válidas. Las soluciones aproximadas incluían: reglas de prioridad, heurísticas de cuello de botella móviles, programación oportunista, búsqueda local, búsqueda tabú, algoritmos genéticos, propagación de restricciones y cadenas de eyección entre otros (Błazewicz et al., 1996).

Fuentes más recientes muestran que, en el entorno actual, la programación del taller de trabajo debe tratar con un sistema de fabricación inteligente respaldado por tecnologías de fabricación novedosas y emergentes. Dentro de estas tecnologías se incluyen la personalización en masa, los sistemas ciber-físicos (CPS), Big Data, Internet de las cosas (IoT), Inteligencia Artificial (AI), Gemelos Digitales y SMAC (Social, Mobile, Analytics, Cloud); por lo que la investigación cambia su enfoque hacia el modelado y la optimización con programación distribuida inteligente (Zhang et al., 2019).

En este mismo contexto Zhang et al. (2019) presentan un resumen de las características de los algoritmos existentes y sus desafíos, el cual se muestra en la Tabla 2 parcialmente modificado.

Tabla 2 Características de Algoritmos de Optimización.
Fuente: Modificado de Zhang et al. (2019)

Algoritmo de Optimización	Tipo	Ventajas	Limitaciones
Métodos Exactos de Optimización	Métodos de reglas eficientes, técnicas de programación matemática y métodos de ramificación y acotación	Logra la solución óptima exacta en tiempo polinómico para problemas especiales	Solo se pueden resolver problemas a pequeña escala
Métodos Aproximados	Método constructivo	Consigue soluciones en muy corto tiempo	Se pueden generar soluciones infactibles
	Inteligencia Artificial	Especialmente para la programación dinámica, para hacer frente a las llegadas de trabajo al azar y las averías de la máquina	Aproximación y generalización insatisfactoria del entrenamiento y la llamada secuenciación "distribuida"
	Métodos de Búsqueda Local	Se puede lograr una solución óptima si se da suficiente tiempo	Necesita mucho tiempo
	Metaheurísticas	Algunos tienen una buena eficiencia para la búsqueda global y otros caen en un óptimo local	Algunos caen fácilmente en un óptimo local otros tienen poca eficiencia

Lee y Loong (2019) muestran en su revisión bibliográfica, sobre algoritmos para la resolución de una variante específica del JSSP, que el 92% de los algoritmos propuestos corresponden a métodos de aproximación. En sus conclusiones, mencionan el desafío potencial que representa, para los investigadores actuales, el tratamiento desde la programación computacional. Profundizando sobre esta línea de pensamiento Zhang et al. (2019) afirman que el uso de una red neuronal profunda puede ser útil para la resolución dinámica del JSSP en tiempo real y, de hecho, para la programación distribuida en el futuro. Ósea usar la red neuronal profunda para hacer programación distribuida en la red apoyándose en internet de las cosas (internet of things – IoT) y en máquinas inteligentes dónde cada centro de trabajo puede hacer su propio secuenciamiento.

Por lo tanto, las investigaciones recientes se están enfocando en el uso de técnicas computacionales como es el aprendizaje automático y, en especial, en los algoritmos de RL, DL o DRL para la resolución del JSSP. El uso de estas técnicas está aumentando porque pueden manejar problemas NP-completos. Lo hacen al aprender relaciones complejas entre las variables de entrada y salida, que son difíciles de expresar con modelos analíticos (Kim et al., 2020). También presentan ventajas en cuanto a su costo computacional frente a otras técnicas, como los algoritmos metaheurísticos (Dong et al., 2020). Por ejemplo, Dong et al.(2020) muestra que para un cambio en el tamaño del problema de un 42% su algoritmo (DQN) solo necesita 9% de tiempo adicional para solucionar el problema contra un 176% de una heurística convencional (la heurística utilizada es una variación del algoritmo heterogéneo de tiempo de finalización más temprano). La razón de estas ventajas se da ya que permiten utilizar los conjuntos resueltos para mejorar los algoritmos o bien reciclar las soluciones para aplicarlas en muestras nuevas (Bouška et al., 2020).

Una vez revisadas las propuestas de solución al JSSP, a continuación, se analizan en detalle los tres tipos de algoritmos más interesantes para nuestra investigación: aprendizaje profundo, aprendizaje por refuerzo y aprendizaje reforzado profundo.

El aprendizaje profundo (Deep Learning - DL) permite que los modelos computacionales, que se componen de múltiples capas de procesamiento, aprendan a operar con representaciones de datos con múltiples niveles de abstracción. Los algoritmos de aprendizaje profundo son capaces de descubrir una estructura compleja oculta en grandes conjuntos de datos; esto lo hacen mediante el uso del algoritmo de retro-propagación. En un sistema típico de aprendizaje profundo, puede haber cientos de millones de pesos ajustables y cientos de millones de ejemplos etiquetados con los que entrenar la red neuronal. Para ajustar adecuadamente los vectores de pesos, el algoritmo de aprendizaje calcula un vector de gradiente que, para cada peso, indica en qué cantidad aumentaría o disminuiría el error si el peso aumentara en una pequeña cantidad. Dicho vector de peso se ajusta en la dirección opuesta al vector de gradiente. La función objetivo, promediada en todos los conjuntos de entrenamiento, puede verse como una especie de paisaje montañoso en un espacio de soluciones de alta dimensión. El vector de gradiente negativo indica la dirección del descenso más pronunciado en este paisaje, acercándolo al mínimo error de salida (Lecun et al., 2015).

El aprendizaje por refuerzo (Reinforced Learning - RL) es un área importante del aprendizaje automático inspirada en el estilo de aprendizaje inherente de los seres vivos. Un modelo de RL consta de un agente, un entorno, un espacio de estado finito, un conjunto de acciones disponibles para el agente y una función de recompensa. La idea básica de los algoritmos de RL es dejar que el agente aprenda a tomar

decisiones acertadas mediante interacciones de prueba y error con el entorno. En cada momento de decisión, el agente realiza una acción basada en la observación del estado actual en el entorno. Una vez que se ha realizado la acción, el entorno pasará a un nuevo estado. Al mismo tiempo, el agente recibirá una recompensa que refleja el valor de la transición de estado (Wei et al., 2018).

Una variante de RL es el aprendizaje reforzado profundo (Deep Reinforced Learning - DRL). El DRL combina RL y técnicas de aprendizaje profundo. DRL utiliza una red neuronal profunda (Deep Neural Network - DNN) para construir la correlación entre cada par estado-acción y su función de valor asociada. Además, basado en la DNN subyacente, se utiliza un marco de aprendizaje profundo Q para decisiones en línea, es decir, sugerir acciones opcionales para el agente (Wei et al., 2018). Q hace referencia a la función utilizada la cual se actualiza según el valor máximo esperado del siguiente estado y un valor de recompensa. (Aydin & Öztemel, 2000).

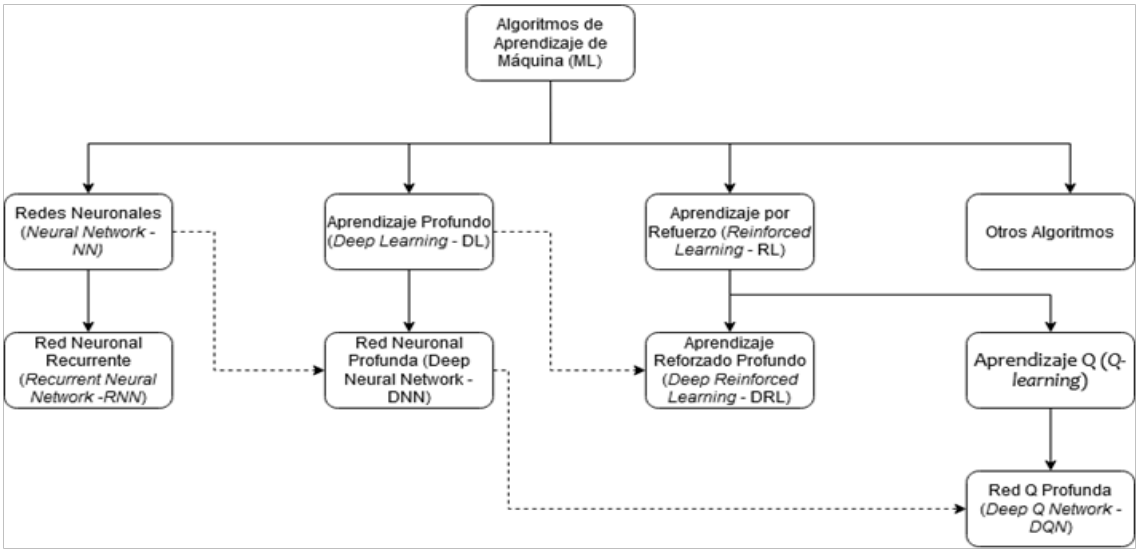
El concepto de red Q profunda (Deep Q Network - DQN) puede considerarse como un aproximador de función Q de red neuronal con pesos θ. Al tomar directamente los datos sin procesar (características de estado) como entrada y el valor de la función Q de cada par de estado-acción como salida. DQN puede manejar un proceso de decisión complicado con un espacio de estado grande y continuo. (Luo, 2020).

El algoritmo de aprendizaje Q (Q-learning), es una técnica de aprendizaje de refuerzo. En este algoritmo el agente intenta aprender una política óptima de estado-acción basada en una secuencia de: estado - acción - recompensa, que representan las interacciones que el agente tuvo con el mundo. La política óptima se ejecuta mediante la selección del mejor estado: acciones de acuerdo con los valores de utilidad aprendidos (Orhean et al., 2018).

Una red neuronal recurrente (Recurrent Neural Network -RNN) es una red neuronal que opera en el tiempo. En cada paso de tiempo, acepta un vector de entrada, actualiza su estado oculto (posiblemente de alta dimensión) a través de funciones de activación no lineales y lo usa para hacer una predicción de su salida (Martens & Sutskever, 2011).

Con la finalidad de explicar mejor las relaciones entre estos tipos de algoritmos se presenta la figura #3. Todos son parte del aprendizaje de máquina (Machine Learning – ML). Dentro de los tipos de algoritmos de interés para la investigación se consideran únicamente tres categorias: NN, DL y RL; dichas categorias agrupan varios algoritmos puros de estos tipos y otros que son mezclas como es el caso de DRL, DNN o DQN.

Figura 3 Relación entre Algoritmos de Solución



5. Caracterización de los algoritmos para la resolución del problema de programación del taller de trabajo y sus derivados

En esta fase, se analizaron los 14 artículos seleccionados con el fin de responder a las preguntas de investigación PI_3 y PI_4. Las características que se evaluaron fueron: tipo de problema, algoritmo de solución, método de validación, algoritmos contrastados, objetivo del algoritmo, tamaño del problema resuelto, CPU utilizado y tiempo de solución.

Tabla 3 Soluciones de la literatura al JSSP

Autor	Problema	Algoritmo de Solución	Método de Validación	Algoritmos contrastantes	Objetivo del Algoritmo	Tamaño del Problema Resuelto	Tiempo de Cálculo	CPU Utilizado
Shahrabi et al., 2017	Dynamic JobShop Scheduling (DJSS)	VNS Q-Learning	Simulación	SPT / LIFO / FIFO / GVNS / RLVNS	Tiempo medio de proceso	M:10 x J: 2000	No se indica	No se indica
Shiue et al., 2018	RTS	RL: Q-Learning	Simulación	SOM / GA+DT / GA+SVM / DS / EDD / SPT / SIO / SRPT	Throughput Mean cycle time Number of tardy parts	No se Indica	No se indica	Core i7-4790 3.6 GHz CPU with the Windows 7 operating system
Wei et al., 2018	JSSP	DRL	Simulación	Random / Round-robin / Earliest / Best-Fit / Sensible	Tasa de éxito Tiempo de respuesta	J:5000	0,35 segundos	5 high-CPU VM
Stricker et al., 2018	Dispatching	RL: Q-Learning	Caso de estudio	FIFO	Utilización del Sistema	1 millón de acciones	2,6 días	Intel1 Core i7-6700 with 3.40 GHz and 16 GB RAM
Waschneck et al., 2018 (a)	Flexible JobShop Scheduling	DQN	Simulación	Heurísticas de despacho	Utilización del tiempo disponible	4 workcenter in a plant	No se indica	No se indica
Lin et al., 2019	JSP Smart mfg	Multiclass DQN (MDQN)	Simulación	FIFO / SPT / LPT / MOPNR / LOPT / SQN / LQN	Makespan	M: 15 X J:20	No se indica	No se Indica
Zang et al., 2019	JSSP	HDNNS	Simulación	DQN / DRL / HNN ANN / SPT / STPT	Makespan	M:13 X J:13	177,2 segundos	Lenovo k4450, Ubuntu 16, and CPU i4700 2.1 MHZ.
Y. Wang et al., 2019	Multi objective workflow scheduling	DQN	Simulación	GTBGA / NSGA-II / MOPSO	Makespan Costo de entrega	138 tareas	No se indica	No se indica
Kim et al., 2020	Production scheduling problem-automated material handling system (AMHS)	DNN	Simulación	Minimum workload (MW) / Closest / Shortest processing time (SPT) / Deterministic Optimization (OPT)	Throughput Utilización de máquinas	M: 2, 3	No se indica	i5-4310M 2.70GHz processor, and 32GB RAM

Bouška et al., 2020	SMTTP	Regression-based Decomposition Algorithm (HORDA) Recurrent neural network (RNN)	No se indica	NBR / HORDA + NBR / HORDA + NN / TTBR	Tardanza Total	M:1 X J: 500	5,16 segundos	single-core Xeon(R) Gold 6140 CPU and 8GB RAM and Nvidia GTX 1080 Ti.
Dong et al., 2020	Task Scheduling Problem	DQN	Simulación	HEFT / CPOP / LOOKAHEAD / PEFT	Makespan	Tareas: 30, 50, 70, 100 X servidores: 2, 3, 4, 5	0,0267 segundos	3.6GHz Intel Core i7 CPU and 8GB RAM
Luo, 2020	DFLSP	DQN	Simulación	Std Q-Learning / FIFO / MRT / EDD / SPT / LPT / Random	Tardanza Total	M:50 X J:200	No se indica	Intel Core i7-6700 @ 4.0 GHz CPU and 8 GB RAM
Ottoni et al., 2020	Sequential ordering problem (SOP)	RL: Q-learning RL: SARSA	Simulación	TSP / MKP / K-Server	Unidad de distancia	Hasta 11 nodos X 5548 Restricciones	No se indica	No se indica
Y. F. Wang, 2020	Dynamic Scheduling Problem	Q - Learning (WQ_CDS)	Simulación	BQ_ASCP / CSQ / SMGWQ /	Tardanza mínima promedio	J: 300 M: 6	No se indica	Core I7 2,9 GB

"En la Tabla 3 es posible observar hallazgos relevantes en la investigación reciente sobre el uso de NN, DL y RL o sus derivados como mecanismos para la solución del JSSP. El primero de los hallazgos es la escasa investigación reciente sobre el tema. Solo se han localizado 14 artículos publicados en años recientes (mediante la estrategia de búsqueda utilizada), lo cual evidencia que es un campo del conocimiento donde aún hay espacio para la investigación tal cual afirma Stricker et al. (2018). Se concluye, por tanto, que es necesaria más investigación en el área de modelado y diseño de algoritmos de aprendizaje automático como solución del JSSP. También es necesaria la investigación para lograr una aplicación amplia en otras áreas de control de producción, como la asignación de empleados y el control de capacidad."

Los métodos de validación y prueba empleados han sido mayoritariamente la simulación mediante la utilización de datos sintéticos. Aproximadamente en el 80% de las investigaciones se ha utilizado la simulación cómo mecanismo de validación. En el 10% se ha hecho mediante un caso de estudio, también simulado; finalmente el 10% restante no lo indica. Esto muestra, en primera instancia, que los desarrollos realizados hasta ahora no han llegado a la fase de implementación en entornos reales, tal cual evidencian varios de los autores, lo que implica que, o no se tiene la madurez suficiente para realizar un ensayo en un entorno real, o los investigadores no han encontrado empresas interesadas en someter a pruebas estos enfoques en sus talleres de trabajo.

Es también posible afirmar que los indicadores predilectos utilizados como objetivos de la programación son: el tiempo de finalización de todas las tareas (makespan) con un 30%, así como el rendimiento del taller (Throughput) y la tardanza total (Total Tardiness) con un 20% cada uno. No obstante, ninguno de los artículos argumenta o justifica la selección de estos indicadores sobre otros también de uso frecuente en la industria.

Otra conclusión evidente es que solo el 20% de los artículos trabaja sobre el JSSP general, en tanto que los demás abordan problemas derivados de este. Esto se considera positivo ya que muestra que el campo de estudio posee espacio para nuevas investigaciones.

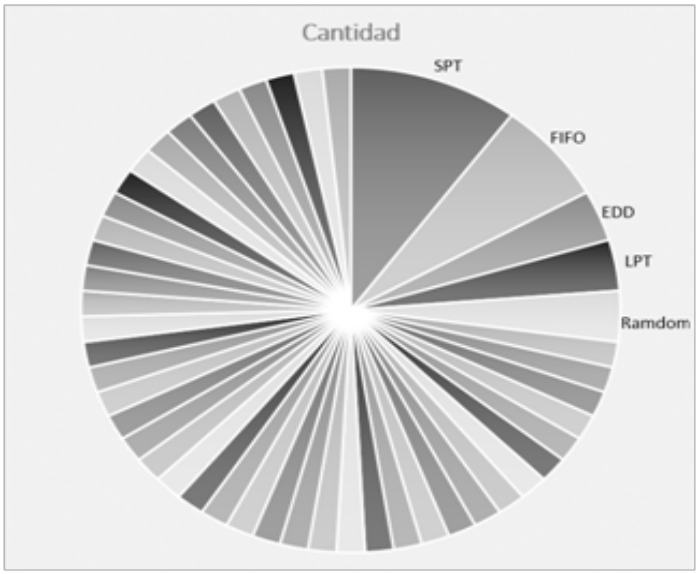
En cuanto a los equipos de computación utilizados, el 90% utilizó equipo de uso general, es decir, computadoras que fácilmente se pueden encontrar en cualquier empresa o inclusive podrían ser de uso personal. Esto es también una ventaja importante porque facilitaría una eventual aplicación a problemas reales, al no ser el equipo de computación una limitación para la implementación de los algoritmos a nivel industrial.

El tamaño de problema y el tiempo de cálculo están estrechamente relacionados. En los artículos estudiados existe una alta diversidad de configuraciones de problemas y, por ende, de tiempos de solución, habiendo problemas tan sencillos como 2 máquinas hasta problemas tan complejos como 1 millón de acciones (trabajos). Esta diversidad de

situaciones y la variedad de problemas abordados muestra la diversidad de ámbitos de aplicación que tiene el tema objeto de estudio, así como sus posibles variaciones. Estas variaciones están a su vez relacionadas con los tiempos de solución requeridos, yendo desde tiempos de computación exponenciales para algoritmos optimizadores hasta tiempos polinómicos para algoritmos heurísticos.

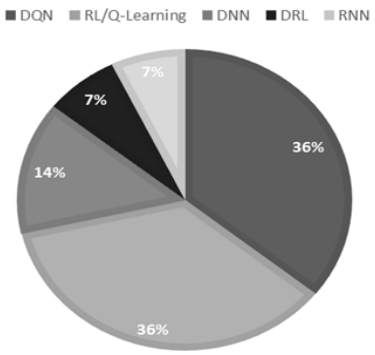
Los algoritmos de contraste (normalmente heurísticas de baja complejidad) utilizados para comparar los algoritmos propuestos son abundantes. En la figura 4 se muestra que del total de 59 algoritmos utilizados el más frecuente (en el 10% de los casos) fue el tiempo de procesamiento más corto (Shortest Processing Time - SPT), primero en entrar primero en salir (First In First Out - FIFO) es el segundo más utilizado (en un 7% de los casos) y tiempo de procesamiento más largo (Longest Processing Time - LPT) junto con secuenciación aleatoria (Random) comparten el cuarto puesto (en un 3% de los casos). Todos los demás solo aparecen en una ocasión, tal como se muestra en la tabla 3.

Figura 4 Algoritmos empleados para contrastar



Finalmente, en cuanto a los algoritmos utilizados, predominan RL: Q-Learning y DQN con un 36% de los casos cada uno, DNN en un 14% de los casos y RNN y DRL en un 7% cada uno (Figura 5). Es decir, estos dos tipos de algoritmos (RL: Q-Learning y DQN) concentran el 70% de las soluciones planteadas en años recientes en el campo de estudio. De este análisis se infiere que los algoritmos que se deben contemplar como posibles soluciones novedosas al problema de JSSP deben ser del tipo: RL/Q-Learning o DQN. Igualmente se evidencia que es en estos tipos de algoritmos dónde se debe profundizar en futuras investigaciones para identificar cuáles son las ventajas que ofrecen frente a otros tipos de algoritmos. Igualmente, las futuras investigaciones deben responder por qué son los más utilizados, también deberá entenderse sus requerimientos técnicos para poder ser implementados exitosamente en la industria.

Figura 5 Algoritmos empleados en la solución



6. Análisis de los principales hallazgos y futuras líneas de investigación

Para responder de forma adecuada a la pregunta de investigación PI_5, es necesario realizar un análisis crítico de los principales hallazgos encontrados en la revisión de la literatura. Con el objetivo de identificar las líneas de investigación futuras de forma clara y que respondan a los planteamientos de varios autores cuando sea posible, por lo tanto, esta sección está dividida en dos apartados. En el primero se realiza una discusión crítica de los hallazgos expuestos por los autores. En el segundo apartado se detallan las principales líneas de investigación que surgen de dicho análisis.

6.1. Discusión crítica de los hallazgos

Kim et al. (2020) afirman que su modelo alcanzó un desempeño mejor que las reglas heurísticas y métodos de optimización deterministas en los ejercicios de simulación desarrollados. Esto es alentador en tanto que denota resultados favorables en una eventual aplicación a nivel industrial. No obstante, la desventaja que presenta el desarrollo es que responde a una aplicación muy específica, donde el mecanismo para mejorar el programa está relacionado a los agentes distribuidores de materiales, sean estos humanos o automáticos, por tanto, su aplicación en procesos en los que no predominen las líneas de ensamble o donde las distancias entre centros de trabajo no sean despreciables es limitada. Por otro lado, una fortaleza de la investigación de Kim et al. (2020) es su desarrollo en un entorno de producción dinámico, lo que garantiza la capacidad de su método para funcionar en condiciones cambiantes, situación muy usual las plantas de producción.

Bouška et al. (2020) plantean una dificultad intrínseca al uso del aprendizaje automático (ML), la cual consiste en la necesidad de obtener una muestra de entrenamiento con características semejantes al problema que se quiere resolver. Es por esta razón que plantean el uso de aprendizaje por refuerzo para salvar este obstáculo, lo cual es un enfoque destacable para el despliegue de estas herramientas de forma práctica en la industria. La solución planteada por Bouška et al. (2020) combina un método de investigación de operaciones de vanguardia (Optimizador heurístico mediante el algoritmo de descomposición basado en regresión) con una DNN. Los resultados experimentales muestran que su enfoque proporciona soluciones casi óptimas muy rápidamente y también es capaz de generalizar el conocimiento adquirido a conjuntos más grandes sin afectar significativamente la calidad de las soluciones. La mayor consideración que debe tenerse al estudiar este resultado tan prometedor es el problema que resuelve: el SMTTP es un problema relativamente simple dentro de la familia de problemas de secuenciación en el taller. No obstante, su planteamiento de usar aprendizaje por refuerzo es una observación válida que debería tenerse en cuenta para futuros trabajos.

Luo (2020) ahonda en la línea sugerida por Bouška et al. (2020) al utilizar un algoritmo del tipo DRL para resolver el problema dinámico y flexible de programación de talleres (Dynamic Flexible Job Shop Scheduling Problem – DFJSP) bajo nuevas inserciones de trabajos con el objetivo de minimizar la tardanza total. Sus resultados demuestran que el DQN utilizado se desempeña significativamente mejor que las reglas de despacho compuestas y otras reglas de despacho bien conocidas para configuraciones de producción entrenadas y no entrenadas. Las comparaciones entre DQN y el agente de Q-learning estándar verificaron aún más la superioridad de DQN en el manejo del espacio de estado continuo. Estos resultados respaldan que la propuesta de utilizar el aprendizaje por refuerzo arroja mejores resultados a utilizar únicamente el aprendizaje profundo para la solución del problema de secuenciación del taller, teniendo en consideración que el problema resuelto por Luo (2020) es un caso específico del problema general, pero sin lugar a duda más complejo que el resuelto por Bouška et al. (2020).

Zang et al. (2019) aportan dos innovaciones al JSSP utilizando algoritmos de DL. La primera es una transformación bidimensional de convolución, que convierte los datos irregulares en el proceso de programación del taller en datos regulares; lo que permite que se utilice una operación convolucional profunda para resolver el JSSP. La segunda es la estructura híbrida de la red neuronal profunda

que incluye una capa de convolución, una capa totalmente conectada y una capa atenuante, esta estructura puede completar efectivamente la extracción del conocimiento del programa de producción. Estas innovaciones permiten que su algoritmo consiga mejores resultados, usando el Makespan como indicador, frente a soluciones basadas en HNN y ANN. La propuesta de Zang et al. (2019) es prometedora dentro de las soluciones basadas en DL, sería interesante validarlas usando otros indicadores frecuentes para el JSSP. No obstante, esta solución presenta la limitación para su aplicación práctica debido, precisamente al tipo de algoritmo que usa (DL) el cual requiere de una cantidad importante de datos para poder realizar la fase de entrenamiento. Esto representa una limitación para pasar del ejercicio teórico de la investigación a la implementación práctica en los procesos productivos ya que grandes conjuntos de datos no están normalmente disponibles como menciona Bouška et al. (2020).

Otro trabajo que investigó un algoritmo de RL fue el desarrollado por Shiue et al. (2018), en el que utiliza un mecanismo de selección de reglas de despacho múltiples (Multiple Dispatching Rules – MDR) que responde de manera eficiente a los cambios en el entorno del taller, a la vez que se muestra adecuado para incorporar en la operación de un sistema con programación en tiempo real (Real Time Scheduling – RTS) para una fábrica inteligente. Además, ese enfoque basado en RL emplea un método inteligente y dinámico para seleccionar MDR, que se basa en el estado de un sistema de fabricación al final de un período de programación dado, donde luego determina los MDR apropiados para el siguiente período. Los resultados producto de la simulación realizada muestran que, durante un largo período, de acuerdo con varios criterios de rendimiento, este enfoque funcionó mejor que un mecanismo de selección de MDR basado en mapas autoorganizados (Self-Organizing Map – SOM) previamente propuesto, que un RTS basado en aprendizaje automático que usa el enfoque reglas de despacho único (Single Dispatching Rule - SDR) y que las reglas de despacho individuales heurísticas.

Ottoni et al. (2020) presentan algunos hallazgos importantes, pese a no resolver el problema de interés para nuestra investigación, su solución aborda la aplicación de RL para resolver el problema de orden secuencial (Sequential Order Problem - SOP) utilizando herramientas estadísticas, como ANOVA y el método Scott-Knott, analizan el rendimiento del algoritmo de RL para elegir el mejor conjunto de parámetros. Este enfoque podría ser valioso y utilizarse en futuras soluciones para el problema de JSSP, para presentar evidencia estadística que sustente los resultados presentados. Por otro lado, también es novedoso el uso de un experimento factorial completo y el método Scott-Knott para encontrar la mejor combinación de niveles de factores.

Dong et al. (2020) muestran el uso de un algoritmo basado en DQN para solucionar el problema de programación de tareas (Task Schedulling - TS) que proporciona mejores resultados, medidos a través del makespan y el tiempo de computación, que las heurísticas convencionales. De este trabajo es destacable el hecho que el problema que resuelve es matemáticamente más sencillo que el problema general de JSSP, lo que explica los tiempos computacionales bajos con respecto a otras soluciones. También es importante tener presente que el análisis que se realiza en el trabajo no es sobre un flujo de producción o ensamble, sino sobre requerimientos para un sistema computacional y, dada la naturaleza del contexto en que se aplica la solución, puede generar resultados diferentes a aplicar el mismo enfoque y algoritmo a un problema de programación de tareas en un proceso productivo convencional. Pese a este último punto, es claro que el uso de un algoritmo del tipo DQN es prometedor para solucionar el problema de JSSP, tal y como proponen Dong et al (2020), Bouška et al. (2020), Luo (2020) y Lin et al., (2019).

Stricker et al. (2018) muestran un uso particular de RL. Concluyen que la aplicación de algoritmos basados en datos puede mejorar la eficiencia operativa de los sistemas de control de producción. En concreto indican que un algoritmo de despacho de orden adaptativo basado en RL puede superar los enfoques heurísticos basados en reglas existentes. Sin embargo también mencionan que las limitaciones de los algoritmos de aprendizaje automático aún prevalecen en términos de reproducibilidad y solidez de la solución y que su trabajo lleva la aplicación de algoritmos de aprendizaje y la transición hacia sistemas de producción autónomos un paso más cerca de la realidad , pero no lo consiguen aún, es decir su solución no es factible de implementarse a nivel industrial, por lo que se queda en un desarrollo teórico más que en una contribución aplicable en el mundo real. Esta conclusión nos lleva a proponer que una investigación futura sobre este tema debe alejarse de los algoritmos de RL puros y acercarse a los algoritmos de DL y DRL los que, hasta el momento, han mostrado un mejor desempeño con problemas específicos de secuenciación.

6.2. Líneas futuras de investigación

La primera línea de investigación futura que se deriva del estudio realizado consiste en la construcción entornos de simulación. Estos entornos deben reflejar de mejor manera la realidad de los talleres de trabajo, para poder experimentar con los algoritmos de aprendizaje automático aplicados a la programación de órdenes de producción. Destacan dos líneas de trabajo a tener en cuenta en la construcción de dichos entornos de simulación. La primera consiste en agregar a las simulaciones situaciones que ocurren en la realidad y que, en ocasiones, se han obviado en los estudios previos; por ejemplo: averías de máquinas, variaciones en el tiempo de procesamiento, entre otros. También deben incluirse objetivos más allá del makespan como, por ejemplo, el equilibrado de cargas, como lo indican Kim et al. (2020), Wang et al. (2019), Wei et al. (2018) y Luo (2020).

La segunda línea de trabajo a considerar consiste en incrementar el tamaño del problema a resolver para acercar las simulaciones a los entornos reales, tal como indican Shiue et al. (2018), Kim et al. (2020), Lin et al. (2019) y Waschneck et al. (2018).”

La segunda línea de investigación futura que se propone es el desarrollo de algoritmos de aprendizaje automático que tengan la capacidad de ser multi objetivo Dong et al. (2020) ya que, en entornos reales, se utilizan varios indicadores para la toma de decisiones con relación a la programación de las tareas del taller de trabajo.

Como tercera línea de investigación futura se plantea el desarrollo de algoritmos de aprendizaje automático que sean capaces de reaccionar ante cambios en las condiciones del sistema de forma autónoma (Shiue et al., 2018), (Wang, 2020), (Kim et al., 2020), esto cobra especial importancia en entornos de la Industria 4.0.

7. Conclusiones

El uso del ML, específicamente de algoritmos del tipo DL, RL y DRL están empezando a ser considerados como mecanismos de solución para el JSSP y sus derivados. No obstante, estos aún no han alcanzado un nivel de desarrollo tal que haya permitido su validación en procesos productivos reales. Por lo tanto, es posible afirmar que este es un campo de investigación fértil para ser desarrollado.

Todas las investigaciones analizadas concuerdan en que los enfoques basados en aprendizaje automático muestran un mejor desempeño; según los indicadores seleccionados en cada caso, que las heurísticas convencionales. En la mayoría de los casos, también, lo hacen con tiempos de procesamiento aceptables. Por lo que se espera que el futuro de la resolución de los problemas de programación en el taller de trabajo pase por el uso del aprendizaje automático. Cuando se contempla el problema JSSP en conjunto con la industria 4.0 toma aún más relevancia dicho enfoque.

Los entornos de solución utilizados hasta ahora carecen aún de las características de diversidad y complejidad propias de las aplicaciones reales. Esta es otra línea de investigación a desarrollar: entornos de prueba que emulen de una mejor manera la realidad de los talleres de trabajo y sus condiciones.

Finalmente, es también patente que los algoritmos que más se están utilizado en la actualidad son del tipo DQN y RL/Q-Learning. Son también los que han mostrado mejores resultados y con mayor capacidad de ser implementados en procesos reales. Esta será en una importante línea de investigación en los próximos años, presentando un potencial importante para trasladar los algoritmos a entornos reales en la industria.

Referencias

1. Aydin, M. E., & Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33(2), 169–178. https://doi.org/10.1016/S0921-8890(00)00087-7

2. Błazewicz, J., Domschke, W., y Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1), 1–33. https://doi.org/10.1016/0377-2217(95)00362-2

3. Bouška, M., Novák, A., Šůcha, P., Módos, I., y Hanzálek, Z. (2020). Data-driven algorithm for scheduling with total tardiness. ICORES 2020 - *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems*, Icores, 59–68. https://doi.org/10.5220/0008915300590068

4. Cunha, B., Madureira, A. M., Fonseca, B., y Coelho, D. (2020). *Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review. Advances in Intelligent Systems and Computing*, 923, 350–359. https://doi.org/10.1007/978-3-030-14347-3_34

5. Dong, T., Xue, F., Xiao, C., y Li, J. (2020). Task scheduling based on deep reinforcement learning in a cloud manufacturing environment. *Concurrency Computation*, 32(11), 1–12. https://doi.org/10.1002/cpe.5654

6. González-Neira, E. M., Montoya-Torres, J. R., y Barrera, D. (2017). Flow-shop scheduling problem under uncertainties: *Review and trends. International Journal of Industrial Engineering Computations*, 8(4), 399–426. https://doi.org/10.5267/j.ijiec.2017.2.001

7. Kim, H., Lim, D. E., y Lee, S. (2020). *Deep Learning-Based Dynamic Scheduling for Semiconductor Manufacturing with High Uncertainty of Automated Material Handling System Capability. IEEE Transactions on Semiconductor Manufacturing*, 33(1), 13–22. https://doi.org/10.1109/TSM.2020.2965293

8. Lecun, Y., Bengio, Y., y Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. https://doi.org/10.1038/nature14539

9. Lee, T. S., y Loong, Y. T. (2019). A review of scheduling problem and resolution methods in flexible flow shop. *International Journal of Industrial Engineering Computations*, 10(1), 67–88. https://doi.org/10.5267/j.ijiec.2018.4.001

10. Lin, C. C., Deng, D. J., Chih, Y. L., y Chiu, H. T. (2019). Smart Manufacturing Scheduling with Edge Computing Using Multiclass Deep Q Network. *IEEE Transactions on Industrial Informatics*, 15(7), 4276–4284. https://doi.org/10.1109/TII.2019.2908210

11. Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing Journal*, 91, 106208. https://doi.org/10.1016/j.asoc.2020.106208

12. Martens, J., & Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. *Proceedings of the 28th International Conference on Machine Learning*, ICML 2011, 1033–1040.

13. Orhean, A. I., Pop, F., & Raicu, I. (2018). New scheduling approach using reinforcement learning for heterogeneous distributed systems. *Journal of Parallel and Distributed Computing*, 117, 292–302. https://doi.org/10.1016/j.jpdc.2017.05.001

14. Ottoni, A. L. C., Nepomuceno, E. G., de Oliveira, M. S., y de Oliveira, D. C. R. (2020). Tuning of reinforcement learning parameters applied to SOP using the Scott–Knott method. *Soft Computing*, 24(6), 4441–4453. https://doi.org/10.1007/s00500-019-04206-w

15. Shahrabi, J., Adibi, M. A., & Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers and Industrial Engineering*, 110, 75–82. https://doi.org/10.1016/j.cie.2017.05.026

16. Sharma, P., y Jain, A. (2016). A review on job shop scheduling with setup times. Proceedings of the Institution of Mechanical Engineers, Part B: *Journal of Engineering Manufacture*, 230(3), 517–533. https://doi.org/10.1177/0954405414560617

17. Shiue, Y. R., Lee, K. C., y Su, C. T. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers and Industrial Engineering*, 125(101), 604–614. https://doi.org/10.1016/j.cie.2018.03.039

18. Stricker, N., Kuhnle, A., Sturm, R., y Friess, S. (2018). Reinforcement learning for adaptive order dispatching in the semiconductor industry. *CIRP Annals*, 67(1), 511–514. https://doi.org/10.1016/j.cirp.2018.04.041

19. Uhlmann, I. R., y Frazzon, E. M. (2018). Production rescheduling review: Opportunities for industrial integration and practical applications. *Journal of Manufacturing Systems*, 49(October), 186–193. https://doi.org/10.1016/j.jmsy.2018.10.004

20. Wang, Y. C., y Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, 18(1), 73–82. https://doi.org/10.1016/j.engappai.2004.08.018

21. Wang, Y. F. (2020). Adaptive job shop scheduling strategy based on weighted Q-learning algorithm. *Journal of Intelligent Manufacturing*, 31(2), 417–432. https://doi.org/10.1007/s10845-018-1454-3

22. Wang, Y., Liu, H., Zheng, W., Xia, Y., Li, Y., Chen, P., Guo, K., & Xie, H. (2019). Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. *IEEE Access*, 7, 39974–39982. https://doi.org/10.1109/ACCESS.2019.2902846

23. Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP*, 72, 1264–1269. https://doi.org/10.1016/j.procir.2018.03.212

24. Wei, Y., Pan, L., Liu, S., Wu, L., y Meng, X. (2018). DRL-Scheduling: An intelligent QoS-Aware job scheduling framework for applications in clouds. *IEEE Access*, 6, 55112–55125. https://doi.org/10.1109/ACCESS.2018.2872674

25. Zang, Z., Wang, W., Song, Y., Lu, L., Li, W., Wang, Y., y Zhao, Y. (2019). *Hybrid Deep Neural Network Scheduler for Job-Shop Problem Based on Convolution Two-Dimensional Transformation. Computational Intelligence and Neuroscience*, 2019(ii). https://doi.org/10.1155/2019/7172842

26. Zhang, J., Ding, G., Zou, Y., Qin, S., y Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing*, 30(4), 1809–1830. https://doi.org/10.1007/s10845-017-1350-2