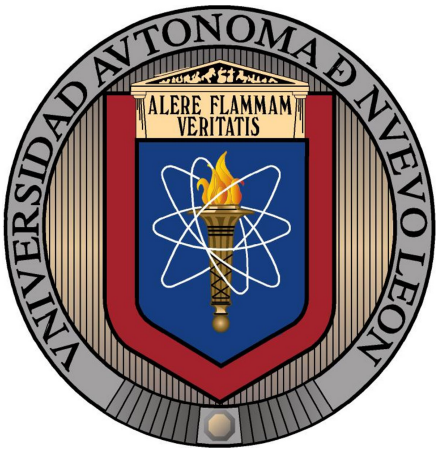
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Uso de Q-learning como alternativa a la solución del problema de programación de taller de trabajo flexible



Arnoldo Del Toro Peña
Director de tesis:
Dr. Vincent Andre Lionel Boyer

Propuesta de Investigación
Maestría en Ciencias de la Ingeniería con Orientación en Sistemas

Monterrey, Nuevo León
30 de mayo de 2022

Índice

Resumen

En esta propuesta de tesis, se propone el uso de una búsqueda de aprendizaje por refuerzo (Q-learning) en un problema de programación de taller de trabajo flexible. Se obtienen instancias a partir de la literatura, el planteamiento metodológico de una búsqueda de soluciones basándose en el algoritmo aprendizaje por refuerzo (Q-learning), resultados con base en un algoritmo genético (Genetic Algorithm) y la discusión de los posibles resultados de una búsqueda de aprendizaje por refuerzo (Q-learning).

In this thesis proposal, the use of a reinforcement learning search (Q-learning) in a flexible job shop scheduling programming problem is proposed. Instances were obtained from the literature, the methodological approach of a search for extreme solutions in the reinforcement learning algorithm (Q-learning), results based on a genetic algorithm and the discussion of the possible results of a quest for reinforcement learning (Q-learning).

Palabras clave— Aprendizaje por refuerzo, metaheurístico, python
Keywords— Q-learning, metaheuristic, python

1. Introducción

En los últimos años se puede decir que ha existido un uso casi exclusivo de métodos exactos y metaheurísticos en el área de investigación de operaciones, dejando de lado el campo que a tomado mucho auge estos últimos años a favor de la inteligencia artificial, es por eso que existe el motivo de una implementación de inteligencia artificial en esta misma área.

Esto representa una oportunidad interesante al momento de pensar en alternativas a métodos metaheurísticos, lo cual contiene la circunstancia de implementar un aprendizaje por refuerzo (Q-learning), siguiendo esta posibilidad se busca la actualización a las emergentes tecnologías como lo es la inteligencia artificial con el fin de lograr la innovación en métodos de aproximación en el campo de la investigación de operaciones.

La pregunta principal que se aborda es la siguiente: ¿Es viable utilizar la búsqueda aprendizaje por refuerzo (Q-learning) en el problema de programación de taller de trabajo flexible?, y de ser así ¿Bajo que condiciones es favorable utilizar una búsqueda aprendizaje por refuerzo (Q-learning)?, bajo estas cuestiones se plantea la hipótesis: "El aprendizaje por refuerzo (Q-learning) es una alternativa de calidad satisfactoria para obtener una solución al problema de programación de taller de trabajo flexible", esto define como propósito principal generar evidencia fundamentada para descartar o aceptar la hipótesis antes definida, y con objetivo específico en el problema de programación de taller de trabajo flexible.

En las secciones siguientes se encuentran: la descripción de la literatura consultada, el marco metodológico, resultados, conclusiones y las respectivas referencias bibliográficas. En la sección de la literatura se tiene un pequeño resumen de los artículos más relevantes para nuestro estudio, en el marco metodológico se describe el proceso y procedimientos llevados a cabo, llegando al apartado de resultados se presentan los datos obtenidos hasta este momento y por último se finaliza con una sección para presentar conclusiones.

2. Marco teórico

2.1. Conceptos

A continuación, se definen unos conceptos que se utilizarán a lo largo de este escrito:

1. **Agente** (Agent): entidad que puede desenvolverse de manera autónoma en un entorno determinado. Los agentes tienen la misión de (usando datos obtenidos del entorno) realizar una tarea lo mejor que puedan. Las acciones realizadas por el agente tienen repercusión en el entorno: producen un cambio de estado y proporcionan al agente una recompensa que le indica lo bien o mal que está realizando la tarea (reward) (Fernández-Vizcaíno y Gallego-Durán, 2016).
2. **Aprendizaje por refuerzo**: (Reinforcement Learning) tipo específico de aprendizaje dentro del campo del aprendizaje automático (Machine Learning) muy utilizado para juegos y entornos interactivos. Se caracteriza por realizar un entrenamiento mediante señales de refuerzo, por no poder determinarse a priori la mejor decisión absoluta ante un estado cualquiera. En este caso, las señales de refuerzo representan una evaluación de las acciones tomadas similar a una función de idoneidad (Fernández-Vizcaíno y Gallego-Durán, 2016).
3. **Inteligencia artificial** (Artificial intelligence): "la habilidad de los ordenadores para hacer actividades que normalmente requieren inteligencia humana". Para brindar una definición más detallada se puede decir que la AI (por sus siglas en inglés) es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano Rouhiainen (2018).
4. **Python**: lenguaje de programación fácil de aprender y potente. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple pero eficaz. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de él el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas Van Rossum y Drake Jr (1991).
5. **Redes Neuronales** (Neural Networks): Existen numerosas formas de definir a las redes neuronales; desde las definiciones cortas y genéricas hasta las que intentan explicar más detalladamente qué son las redes neuronales. Por ejemplo; un sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas Matich (2001).
6. **Aprendizaje por refuerzo** (Q-learning): Watkins y Dayan (1992) es una forma de aprendizaje por refuerzo sin modelos. También puede ser visto como un método de programación dinámica asíncrona (DP). Proporciona a los agentes la capacidad de aprender a operar de manera óptima en los dominios markovianos experimentando las consecuencias de acciones, sin exigirles que construyan mapas de los dominios.
7. **Problema de programación de taller de trabajo flexible (Flexible Job Shop Scheduling Problem)**: Pezzella *et al.* (2008) es una extensión del problema de programación de taller clásico y se considera fuertemente NP-difícil (NP-hard). En FJSSP (por sus siglas en inglés) la misma operación podría ser procesada en más de una máquina por lo que hay máquinas alternativas disponibles para procesar un trabajo en particular. FJSSP consta de dos sub-problemas. Lo primero es asignar cada operación a una salida de máquina, mientras que el segundo se ocupa de la secuenciación de las operaciones asignadas en las máquinas.

2.2. Antecedentes

Desde principios del aprendizaje automático (Machine Learning) hasta la actualidad han habido numerosos avances importantes la mayoría en el campo de los videojuegos, sin embargo su uso no es nulo en el campo de la optimización.

Chen *et al.* (2022) usan el aprendizaje por refuerzo (Q-learning) en un problema de entrega el mismo día (Same-day delivery) utilizando vehículos y drones, también detallan una solución aproximada usando el aprendizaje por refuerzo (Q-learning). Los resultados computacionales mostrados demuestran que el algoritmo es capaz de tomar decisiones de servicio y asignación de drones permitiendo un número esperado mayor de clientes atendidos a lo largo del día. Además se menciona que pueden mejorar los resultados de métodos heurísticos al permitir la reasignación de pedidos aceptados; sin embargo se advierte que esto puede derivar en una explosión de tiempo computacional, lo cual es una advertencia importante a tomar en cuenta.

Huang *et al.* (2022) presentan el método clasificación de corte (Cut Ranking) para seleccionar los cortes en un problema de ramificación y corte para programación entera mixta (mixed-integer programming). Los MIP (por sus siglas en inglés) sintéticos planteados en este artículo demuestran que la selección de cortes basada en una clasificación aprendida es más competitiva con otras heurísticas manuales, y también con capacidad de generalización en problemas con diferentes escalas, rangos de coeficientes o estructuras.

Qiu *et al.* (2022) en lugar de utilizar algoritmos de optimización para resolver el problema de entrega e instalación a domicilio, desarrollan una optimización neuronal basada en un mecanismo de aprendizaje por refuerzo profundo. Los resultados de sus experimentos confirman la calidad de solución y eficiencia computacional ante algunos algoritmos clásicos heurísticos. Qiu *et al.* (2022) dejan en claro que pretenden que su trabajo motive el estudio de aprendizaje por refuerzo en otros problemas de optimización combinatoria.

Pezzella *et al.* (2008) presentan un algoritmo genético (Genetic Algorithm) para la solución al problema de programación de taller de trabajo flexible y demuestran que la propuesta GA (por sus siglas en inglés) es de propósito general y se puede adaptar para cualquier función objetivo sin cambiar la rutina de GA. El enfoque GA es comparado con catorce metaheurísticas y por último los resultados muestran que el enfoque encuentra soluciones que son iguales o superiores a los enfoques anteriores.

Meng *et al.* (2020) presentan resultados a los modelos de programación lineal entera mixta (mixed-integer linear programming) y programación de restricciones (constraint programming), MILP y CP por sus siglas en inglés respectivamente, en el escrito se demuestra que los modelos MILP y CP siempre pueden obtener la misma solución para todas las instancias con el mismo tiempo, mientras que los algoritmos metaheurísticos no pueden asegurar la misma solución en cada repetición incluso para instancias pequeñas, el modelo CP supera a todos los demás algoritmos existentes en términos de calidad en solución y eficiencia.

Palacio *et al.* (2022) presentan un enfoque de aprendizaje para un escenario real de programación de taller de trabajo flexible mundial, los experimentos muestran que el algoritmo es capaz de producir mejores resultados que los existentes en términos de tiempo máximo de terminación de todas las tareas (makespan). También se demuestra como el agente puede tomar decisiones cuando por alguna razón una acción se convierte en horario inválido.

Zhao *et al.* (2019) proponen la implementación de un algoritmo de doble capa de aprendizaje por refuerzo (Q-learning) así como acciones para la solución del problema dinámico de programación de talleres flexibles con fallas en máquinas. Los resultados indican que el modelo de agentes basados con enfoque pueden seleccionar una mejor estrategia bajo diferentes fallas de máquinas, lo cual prueba que el aprendizaje por refuerzo (Q-learning) propuesto es eficiente para el problema dinámico de programación de talleres flexibles con fallas en máquinas.

3. Metodología

El aprendizaje por refuerzo (Q-learning), propuesto por Watkins y Dayan (1992) contiene una de los más populares algoritmos de aprendizaje por reforzamiento de los últimos días. En este algoritmo el Agente intenta una acción en un determinado estado y evalúa sobre la base de una recompensa o castigo al interactuar con el entorno. El Agente al intentar todas las acciones en todos los estados, es capaz de juzgar cuál es la mejor decisión. El diagrama se muestra en la figura 1.

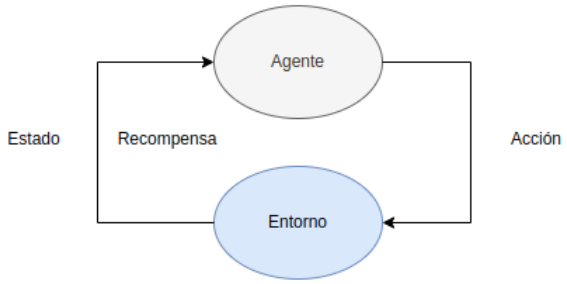


Figura 1: Interacción entre el Agente y el entorno.

Antes del proceso de aprendizaje, Q se inicializa a un posible valor arbitrario. Después, el Agente selecciona una acción en cada tiempo t, recibe una recompensa y llega a un nuevo estado, luego el valor de Q se actualiza con base en la siguiente fórmula:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\, max(Q(s_{t+1}, a) - Q(s_t, a_t))] \quad (1)$$

Donde r_t representa la recompensa recibida cuando el Agente es transferido del estado s_t al estado s_{t+1}. Y α representa la tasa de aprendizaje ($\alpha \in (0, 1]$).

El objetivo del algoritmo aprendizaje por refuerzo (Q-learning) se actualiza siguiendo la fórmula:

$$r_{t+1} + \gamma\, max Q(s_{t+1}, a) \quad (2)$$

Y un episodio del algoritmo termina cuando el estado s_{t+1} es terminal.

Los pasos específicos están detallados a continuación:

Paso 1 Inicializar $Q(s, a)$ arbitrariamente

Paso 2 Establecer el parámetro γ y α

Paso 3 Repetir (para cada episodio):

1 Inicializar *s*

2 Repetir (para cada paso del episodio):

2.1 Elegir *a* desde *s* usando una de las políticas de Q.

2.2 Tomar la acción *a*, observar r, s'

2.3 $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma\, max(Q(s', a') - Q(s, a))]$

2.4 s $\leftarrow s'$

2.5 Hasta que s' sea terminal.

La programación de taller de trabajo se divide en dos categorías funcionales (Zhao *et al.*, 2019):

1. Selección de trabajo
2. Selección de máquina.

De acuerdo con los objetivos, las reglas de selección de trabajo y las reglas de selección de máquina se juntan; por lo tanto se necesita una regla de despacho de doble capa que incluya una capa para la selección de trabajo y una para la selección de máquina.

En la primer capa el conjunto de acciones son las siguientes (Zhao *et al.*, 2019):

1. SPT: representa el mínimo de tiempo de procesamiento de las operaciones que serán seleccionados

2. EDD: representa el mínimo de tiempo de entrega en las operaciones que serán seleccionadas

3. FIFO: representa el primer trabajo en llegar que será seleccionado.

La segunda capa se utiliza para seleccionar la máquina óptima de las máquinas alternativas según sus estados de procesamiento. Para simplificar el problema solo se seleccionará una regla de despacho definida como M, como la acción de la segunda capa que representa la máquina disponible más cercana que será seleccionada (Zhao *et al.*, 2019).

La doble capa en grupo de acción incluye: SPT+M, EDD+M, FIFO+M, Ninguna.

Si se observa, el Agente realiza la selección en función de los estados; por lo tanto es muy importante determinar el número de estados, esto conlleva a dos resultados, por un lado si el número de estados es demasiado grande aumentarán en gran medida la carga calculada y por otro lado si es demasiada pequeña puede que no se adquiera el óptimo.

Una vez mencionado lo anterior se define SD como un indicador para establecer estados. SD es la relación de duración en que una máquina falla al tiempo total de procesamiento de las operaciones restantes en máquina fallida. La fórmula se describe a continuación:

$$SD = \frac{100 \times T}{RT} \tag{3}$$

Donde T es la duración en que la máquina falla y RT representa el tiempo total de procesamiento del resto de operaciones a partir de que la máquina falla.

El Agente de aprendizaje por refuerzo (Q-learning) selecciona la mejor acción a juzgar de la recompensa (reward). Esto implica que la definición de recompensa es muy importante para el algoritmo. En el problema de programación de taller de trabajo flexible el tiempo de espera representa uno de los factores más importantes a evaluar; por lo tanto se define la relación siguiente: "entre más tiempo de espera menos es la recompensa" (Zhao *et al.*, 2019).

Los valores contemplados hasta el día de hoy si la acción seleccionada en SPT+M, EDD+M, FIFO+M y ninguna son: 2, 1, -1 y -2 respectivamente (Zhao *et al.*, 2019).

La estructura del método propuesto se basa en la figura 2 (Zhao *et al.*, 2019).

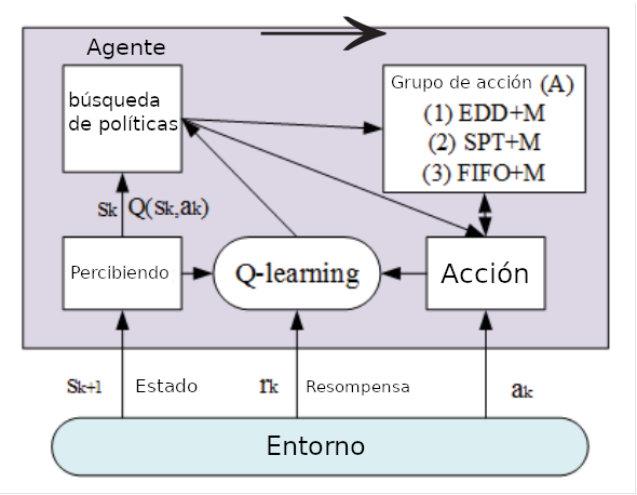


Figura 2: Estructura de la interacción del Agente dentro del entorno.

4. Resultados

4.1. Esquema de planificación inicial basada en GA

Se inicia con una planificación basada en GA programada en python del problema Mk03 que es un caso típico de programación de taller de trabajo flexible presentado por Brandimarte (1993) citado por Zhao *et al.* (2019). El parámetro de la población es de 50, el número de iteraciones es de 500, y las probabilidades transversal y de mutación son 0.8 y 0.2 respectivamente.

A continuación se muestra el diagrama de Gantt del esquema de programación inicial en la figura 3. Y se presenta la curva de variación de la función de aptitud en la figura 4 (Zhao *et al.*, 2019).

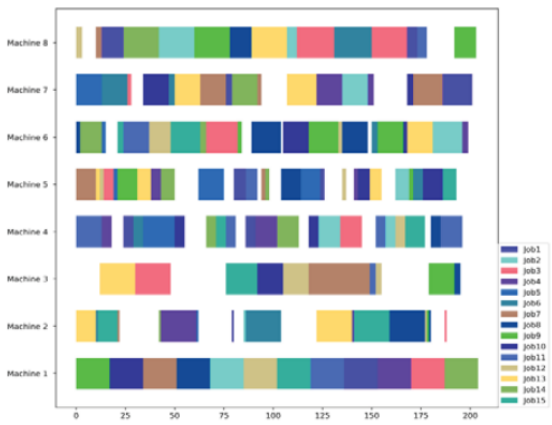


Figura 3: Esquema inicial basado en GA.

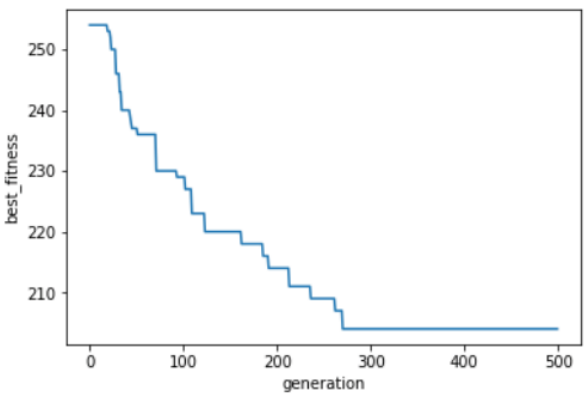


Figura 4: La curva de variación de la función de aptitud.

4.2. Planificación con base en aprendizaje por refuerzo (Q-learning)

Se planea la obtención de distintos resultados variando cada una de las cuatro acciones que se tienen (SPT+M, EDD+M, FIFO+M y ninguna). Después de haber variado las acciones, se tiene contemplado el análisis simultáneo de las decisiones tomadas en cada acción. De acuerdo a los principios antes descritos de aprendizaje por refuerzo (Q-learning), si el agente obtiene una recompensa (reward) positiva el valor de Q incrementa. Cuando el Agente selecciona su primer acción por ejemplo ninguna, el Agente archiva las recompensas positivas esto lo realiza de manera similar para las siguientes acciones. Teniendo los resultados en cada acción se evalúan los tiempos de las cuatro acciones, para concluir en un modelo de validación de nuestro algoritmo.

5. Conclusiones

Hasta el momento en que se escribe esta propuesta, la evidencia bibliográfica indica que el uso de un aprendizaje por refuerzo en un problema de programación de taller de trabajo flexible tiene resultados muy favorables; sin embargo esta propuesta también planea explorar el tiempo computacional de dicho aprendizaje el cual ya se mencionó puede explotar en ciertas circunstancias.

Referencias

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183.

Chen, X., Ulmer, M. W., y Thomas, B. W. (2022). Deep q-learning for same-day delivery with vehicles and drones. *European Journal of Operational Research*, 298(3):939–952.

Fernández-Vizcaíno, G. y Gallego-Durán, F. J. (2016). Ajustando q-learning para generar jugadores automáticos: un ejemplo basado en atari breakout. En *CoSECivi*, pp. 77–88.

Huang, Z., Wang, K., Liu, F., Zhen, H.-L., Zhang, W., Yuan, M., Hao, J., Yu, Y., y Wang, J. (2022). Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353.

Matich, D. J. (2001). Redes neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*, 41:12–16.

Meng, L., Zhang, C., Ren, Y., Zhang, B., y Lv, C. (2020). Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering*, 142:106347.

Palacio, J. C., Jiménez, Y. M., Schietgat, L., Van Doninck, B., y Nowé, A. (2022). A q-learning algorithm for flexible job shop scheduling in a real-world manufacturing scenario. *Procedia CIRP*, 106:227–232.

Pezzella, F., Morganti, G., y Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & operations research*, 35(10):3202–3212.

Qiu, H., Wang, S., Yin, Y., Wang, D., y Wang, Y. (2022). A deep reinforcement learning-based approach for the home delivery and installation routing problem. *International Journal of Production Economics*, 244:108362.

Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*.

Van Rossum, G. y Drake Jr, F. L. (1991). Guía de aprendizaje de python. *Release*, 2.

Watkins, C. J. y Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Zhao, M., Li, X., Gao, L., Wang, L., y Xiao, M. (2019). An improved q-learning based rescheduling method for flexible job-shops with machine failures. En *2019 IEEE 15th international conference on automation science and engineering (CASE)*, pp. 331–337. IEEE.