TRABAJO DE GRADO

Opción Investigación o Proyecto de Grado

Algoritmo genético para resolver el problema de programación de producción tipo Job shop con bloqueo

Corporación Universitaria Remington.
Facultad de Ingeniería.
Tecnología en desarrollo de software.

Juan Camilo Rendon Naranjo.
John Andersson Valencia Palacio.

Proyecto de grado tipo tesis.

2024

Agradecimientos

Agradezco este trabajo a profesores John Andersson Valencia, Porfirio de Jesus Alvarez, Jose Antonio Polo por su apoyo incondicional y motivación a lo largo de este proceso. Gracias por su mentoría.

Tabla de Contenidos

Contenido

Resumen

La programación de producción hace parte de las decisiones operativas de la administración de operaciones que consiste en la asignación y utilización de recursos para la ejecución de tareas con la finalidad de cumplir uno o más objetivos. En este proyecto de investigación se desarrolla un algoritmo genético para resolver el problema de programación de producción en configuraciones job shop con restricciones de bloqueo y con el objetivo de minimizar el tiempo de compleción máximo (makespan)

Para ello se realiza la caracterización y descripción del problema, el desarrollo conceptual y computacional del algoritmo genético para resolver el problema. Finalmente se realiza la validación del metaheurístico desarrollado a partir de corridas de prueba bajo para un conjunto de datos específicos.

Palabras clave

Algoritmo genético, metaheurístico, programación de tareas, job shop, bloqueo

Introducción, Marco teórico o de referencia

Introducción

La gestión de operaciones consiste en gestionar los recursos utilizados para producir y entregar productos y servicios. La función de operaciones es el área dentro de la organización encargada de llevar a cabo esta gestión. Todas las organizaciones cuentan con una función de operaciones, ya que cada una produce algún tipo de bien o servicio. (Slack, 2010).

La estrategia de operaciones ofrece una guía para tomar decisiones dentro de la cadena de suministros, con el objetivo de crear una red de organizaciones cuyo trabajo y producto final estén orientados a satisfacer las necesidades de productos y servicios de los clientes. Las decisiones relacionadas con la capacidad se fundamentan en estimaciones de la demanda futura. (Schroeder Et Al, 2011) El diseño y ubicación de las instalaciones, la planificación agregada y la programación de la producción forman una jerarquía de decisiones sobre la capacidad dentro de una planificación de operaciones, la cual puede abarcar desde el largo plazo hasta el mediano y corto plazo. (Valencia, 2018) La programación es un proceso de toma de decisiones utilizado como base en numerosas compañías manufactureras y empresas de servicios

Consiste en asignar recursos a tareas específicas dentro de un período de tiempo determinado, con el objetivo de optimizar uno o más resultados (Pinedo, 2012). Los recursos y las tareas en una organización pueden presentarse de diversas maneras. Los recursos pueden ser máquinas en un taller, pistas en un aeropuerto, personal en una construcción o unidades de procesamiento en un entorno informático. De esta forma, las tareas pueden incluir operaciones en un proceso de producción, etapas en un proyecto de construcción, ejecución de programas informáticos, entre otros. Un problema de programación de producción se compone de la configuración de las máquinas o

recursos (como una única máquina, máquinas paralelas, flowshop, job shop y sistemas híbridos), las restricciones propias del sistema de producción o servicios (como interrupciones, bloqueos, esperas, tiempos de aprontamiento dependientes de la secuencia, entre otras) y los objetivos a cumplir, como tiempo de finalización, retrasos, costos, eficiencia energética, entre otros

En este trabajo se desarrolla un algoritmo genético que permite generar soluciones siempre factibles al problema de programación de producción en configuraciones job shop con restricciones de bloqueo y con el objetivo de minimizar el máximo tiempo de compleción de los trabajos.

Marco de referencia

El primer trabajo sobre el problema de programación Job Shop con restricciones de bloqueo fue publicado por Mascis & Pacciarelli (2000), en el cual describieron el problema a partir de grafos disyuntivos alternativos que hicieron parte de un heurístico de búsqueda local cuyo desempeño compararon con reglas de despacho.

Algunos autores han abordado el BJSP con características y restricciones adicionales considerando particularidades de algunos sistemas productivos. Lui y Kozan (2009) describen el modelo del problema BJS con máquinas paralelas aplicado en la programación de trenes, Zeng, Tang y Yang (2014) describen el modelo incorporando AGV con tiempos de transporte y estaciones de carga y descarga. Lui y Kozan (2016) trabaja un problema similar utilizando robots para el transporte de los trabajos entre máquinas. Louaqad y Kamach (2018) y Louaqad et Al (2018) trabajan un problema con restricciones adicionales de no espera y tiempos de transporte entre máquinas.

Respecto a métodos de resolución del problema, la mayoría de los trabajos publicados describen y aplican heurísticos para encontrar soluciones con el objetivo de minimizar el tiempo de compleción de los trabajos a ser realizados. Mascis y Pacciarelli (2002) proponen un heurístico Greedy para encontrar soluciones factibles al problema con el objetivo de minimizar el makespan. Por otra parte, Mejía y Montoya (2009) desarrollan un heurístico de búsqueda local integrado con redes Petri para generar soluciones. Hayasaka y Hino (2012) plantean heurístico de búsqueda local con intercambio de trabajos adyacentes evaluando y recuperando la factibilidad de las soluciones a partir de grafos disyuntivos. Louaqad et Al (2018); Gholami y Tornquist (2018), Lange y

Werner (2018) proponen heurísticos constructivos con estrategias de reparación a las soluciones que no son factibles para el problema.

Algunos autores como Lange y Werner (2019), Groflin y Klinkert (2007) y Hayasaka y Hino (2012) proponen estrategias de intercambio pareado de trabajos adyacentes y utilización de reglas de despacho en la generación de soluciones al problema.

De los artículos que han propuesto estrategias metaheurísticas para resolver el problema con restricciones de bloqueo, Brucker y Kampmeyer (2008), Mati, Rezg y Xie (2001) utilizan la Búsqueda Tabú con reparación de soluciones no factibles generadas por vecindarios, Lakehal et Al (2015) proponen dos tipos de búsqueda Tabú diferenciados en la estructura de generación de vecindarios a partir de reglas de despacho. Bendyoudi y AitZai (2017) describen un enfoque de solución a partir de reparación de soluciones no factibles utilizando los arcos de los grafos disyuntivos. Sólo dos artículos han propuesto algoritmos genéticos para resolver el problema con restricciones de bloqueo. AitZai, Benmedjdoub y Boudhar (2012) utilizan la teoría de grafos para generar soluciones a partir de codificación binaria y reglas de despacho, mostrando en sus resultados que la estrategia no generó mejores resultados comparados con la búsqueda tabú de Groflin y Klinkert (2007); mientras que Suavey et Al. (2020) describen el algoritmo genético a partir de vectores de Bierwirth y diversos tipos de bloqueo dependientes de cada trabajo. Estos mismos autores proponen un PSO para resolver dicho problema y realizan comparaciones con los resultados obtenidos con el algoritmo genético. Finalmente, Hou et Al. (2012) proponen un algoritmo Harmony para encontrar soluciones a partir de la reparación de soluciones no factibles, la eliminación de ciclos en grafos disyuntivos y la aplicación de reglas de despacho. AitZai y Boudhar (2013) incorporan un PSO secuencial y uno paralelo para instancias de muchos trabajos y muchas máquinas; y un ramal y límite en instancias pequeñas, para resolver el problema con bloqueo utilizando notación binaria y reglas de despacho para construcción de soluciones.

En cuanto a las funciones objetivo-abordadas en los trabajos previos del BJSP, sólo 2 artículos han trabajado un objetivo distinto al tiempo máximo de compleción. Lange & Werner (2018) y Lange & Werner (2019) proponen heurísticos constructivos a partir de permutaciones y una estrategia de recuperación de factibilidad con el fin de minimizar la tardanza total.

Planteamiento del problema.

En la configuración job shop se tiene un conjunto de pedidos o trabajos a ser procesados en una serie de máquinas. Un pedido consiste en un conjunto de O operaciones que deben ser realizadas según una ruta de procesamiento definida, la cual puede ser diferente para cada pedido. Cada operación puede realizarse sólo en una máquina y cada máquina sólo ejecuta una operación a la vez. Una vez se inicia una operación en una máquina, no puede ser interrumpida. Al considerarse la restricción de bloqueo, un trabajo que es completado en una máquina no podrá ser entregado a la siguiente máquina hasta que esta se encuentre libre o disponible, generando así el bloqueo de la máquina previa.

En términos de bloqueo, existen dos tipos de variantes a considerar asociadas con permitir o no el intercambio de trabajos, el cual se refiere a que, si en algún momento en un programa existe un conjunto de dos o más trabajos, cada uno esperando una máquina que está bloqueada por otro trabajo del conjunto, los trabajos deben moverse simultáneamente a la siguiente máquina. Por el contrario, no se permite el intercambio cuando los trabajos puedan pasar a la siguiente máquina sólo cuando esta se encuentre disponible.

Por lo general, los problemas de programación de producción son NP-Completos y, para resolverlos, se necesitan métodos complejos que consumen mucho tiempo. En este contexto, las técnicas de optimización combinatoria se presentan como una alternativa prometedora, ya que permiten encontrar soluciones de alta calidad en tiempos de cómputo manejables (López, 2013). Los métodos de resolución existentes en la literatura pueden agruparse en dos grandes áreas: Métodos exactos a partir de programación matemática, lineal o no lineal; y los métodos aproximados como heurísticos y metaheurísticos (Yalaoui, 2011). De estos últimos, en los problemas de programación de

producción se han utilizado métodos constructivos, búsqueda local, enfriamiento o recocido simulado (Simulated Annealing-SA), Búsqueda Tabú (TS), Algoritmos Genéticos (AG), Colonias de Hormigas (ACO), Harmony, Enjambre de partículas (PSO), entre otros.

El problema de programación job shop con restricciones de bloqueo (BJSP) se considera en términos de complejidad computacional como Strongly NP Hard considerando únicamente 2 máquinas, según lo establece Mogali (2021), por lo que los métodos de resolución existentes en la literatura para resolver este problema han sido principalmente heurísticos y algunos metaheurísticos con el objetivo de minimizar el tiempo de compleción máximo o Cmax, sólo un par de artículos han abordado el problema para minimizar objetivos distintos como tardanza o demora.

El BJSP se aplica en diversas áreas, como la programación de transporte y la gestión de materiales (Lange & Werner, 2019; Mati y Xie, 2011), la gestión de actividades en almacenes automatizados (Klinkert, 2001), la asignación de trenes a las diferentes rutas (Lange & Werner, 2018; Liu & Kozan, 2009) y la organización de rutas de camiones (Van den Bossche et al., 2020).

Objetivos

Objetivo General

Desarrollar un algoritmo genético para encontrar soluciones eficientes al problema de programación job shop con restricciones de bloqueo y con el objetivo de minimizar el makespan

Objetivos específicos

- Caracterizar el problema de programación de producción en configuraciones job shop con restricciones de bloqueo
- Definir conceptual y computacionalmente el algoritmo genético para resolver el problema de programación.
- Validar el funcionamiento del algoritmo con instancias de prueba

Metodología

Para alcanzar los objetivos planteados en el proyecto, se desarrolla una metodología fundamentada en la información científica a partir de las siguientes fases:

- Modelamiento conceptual y matemático del problema de programación de producción en configuraciones job shop con restricciones de bloqueo; describiendo parámetros, variables, objetivos y restricciones.
- Desarrollo conceptual del algoritmo genético para resolver el problema de programación de producción en configuraciones job shop con restricciones de bloqueo
- Desarrollo computacional del algoritmo genético a partir de la selección de la plataforma y lenguaje y construcción del algoritmo.
- Ejecución de pruebas para validar la funcionalidad para encontrar soluciones al problema de programación de producción en configuraciones job shop con restricciones de bloqueo

Resultados y Discusión

Descripción del problema

El problema de programación *job shop* con bloqueo es una variante del problema clásico en la que se procesan n trabajos en m máquinas, pero con restricciones adicionales. Cada trabajo consta de una serie de operaciones que deben ejecutarse en un orden específico, que puede variar entre los diferentes trabajos, y cada operación tiene un tiempo de procesamiento fijo y conocido. Una vez que una operación comienza en una máquina, esta debe completarse sin interrupciones, y solo puede procesarse un trabajo a la vez en cada máquina. Además, dado que no hay almacenamiento intermedio entre máquinas, un trabajo se queda en la máquina actual hasta que la siguiente esté disponible, lo que añade complejidad y limita la flexibilidad del flujo de trabajo.

Formulación Matemática

A continuación, se presenta la formulación matemática para el problema, incluyendo la notación, los parámetros, la función objetivo, las variables y las restricciones

Parámetros:

$n: número\ de\ trabajos$

$m: número\ de\ máquinas$

$O_{i,u}: U - ésima\ operación\ del\ trabajo\ J_i$

$M(i,u): máquina\ donde\ es\ procesada\ la\ operación\ o_{i,u}$

$p_{i,u}: Tiempo\ de\ procesamiento\ de\ la\ operación\ o_{i,u}$

Conjuntos:

$J = \{j_1, j_2, j_3, .., j_n\}$ *Conjunto de trabajos*

$M = \{M_1, M_2, M_3, .., M_m\}$ *Conjunto de máquinas*

$I = \{O_{1,1}, O_{1,2}, O_{1,3}, .., O_{n,m}\}$ *Conjunto de de todas las operaciones*

Variables:

$s_{i,u}$: *Tiempo de inicio de la operación* $o_{i,u}$

$c_{i,u}$: *Tiempo de terminación de la operación* $o_{i,u}$

c_{max}: *Makespan*

Función objetivo

$$Min\ c_{max} = \max\{c_{j,m}\}, j\ \in \{1, \dots, n\} \quad (1)$$

Restricciones

$$s_{j,v} \geq 0 \quad \forall\ j\ \in \{1, \dots, n\} \quad (2)$$

$$s_{j,v} + p_{j,v} \leq s_{j,v+1} \quad \forall\ j\ \in \{1, \dots, n\} \quad (3)$$

$$\forall\ i, j\ \in \{1, \dots, n\}, i \neq j, M(i,u) = M(j,v), \qquad s_{j,v} + p_{j,v} \leq s_{i,u}\ o\ s_{i,u} + p_{i,u} \leq s_{j,v} \quad (4)$$

$$\forall s_{j,v} > s_{i,u},\ i, j\ \in \{1, \dots, n\}, i \neq j, M(i,u) = M(j,v),\ entonces\ s_{j,v} \geq s_{i,u+1}$$

(5)

La ecuación (1) define la función objetivo. La restricción (2) asegura que los tiempos de inicio de cada oj,v sean positivos. La restricción (3) establece que cada trabajo solo puede ser procesado en una máquina a la vez, mientras que la restricción (4) garantiza que cada máquina solo puede atender un trabajo a la vez. Por último, la restricción (5) corresponde a la condición de bloqueo, que implica que un trabajo permanece en una máquina hasta que la siguiente esté disponible.

Descripción del Algoritmo genético

El algoritmo genético comienza creando una población inicial de individuos, los cuales son evaluados usando una medida de desempeño definida. Luego, los individuos se ordenan del mejor al peor según el valor obtenido en el desempeño, y se selecciona una parte de los mejores. A partir de los individuos seleccionados, se generan nuevas soluciones mediante una operación de cruzamiento. La población se completa añadiendo soluciones generadas de manera aleatoria. El algoritmo se repite hasta que se alcanza un criterio de parada específico. A continuación, se presenta el seudocódigo del algoritmo propuesto.

Seudo-Código

Leer Trabajos, máquinas, rutas y tiempos de procesamiento
Para cada máquina
Listar trabajos que inician en la máquina
Ordenar de manera aleatoria los trabajos en la lista
Programar la operación inicial del primer trabajo en la lista
Actualizar tiempo programable tprog=tprog +Tiempo de operación del trabajo programado
Fin Para

Hasta Finalizar todos los trabajos
Para cada Etapa
Para cada máquina
Definir el cambio de máquina del trabajo programado (M_i a M_j)
Si la Operación programada es la última,
Definir a la máquina como libre
Si en la máquina no hay operaciones por programar,
Definir a la máquina como libre
Si se evidencian intercambios entre trabajos programados
Programar operaciones de los trabajos
Actualizar el tiempo de programación de cada máquina
Si la máquina se encuentra libre
Si hay algún trabajo pendiente de iniciar en la máquina

Programar el primer trabajo en la lista
Actualizar el tiempo de programación de la máquina
De lo contrario
Si hay algún trabajo en proceso para esa máquina
Programar el trabajo en la máquina
Actualizar el tiempo de programación de la máquina
De lo contrario
Definir la máquina como libre
De lo contrario
Si se presentan empates en la máquina destino
Si la máquina destino está libre,
Programar aleatoriamente alguno de los trabajos
Actualizar el tiempo de programación de la máquina
Definir en el estado bloqueada a la máquina donde se encuentra el trabajo no seleccionado
De lo contrario
Definir como bloqueadas a las máquinas donde se encuentran los trabajos
De lo contrario
Asignar cada trabajo a la máquina correspondiente
Actualizar tiempo de programación de cada máquina
Fin de etapa
Calcular tiempo programable como el máximo tiempo de compleción de las operaciones que se encuentran programadas
Actualizar el el tiempo programable de las máquinas que se encuentran libres o bloqueadas
Fin Hasta
imprimir el Cmax, los tiempos de cada operación en cada máquina y el diagrama de Gantt asociado

Representación de una solución

Para ejecutar el algoritmo, se define que una solución factible estará representada por una permutación de trabajos para cada máquina, teniendo en cuenta únicamente aquellos trabajos que inician en esa máquina. Por ejemplo, consideremos los trabajos y las rutas que se presentan en la tabla 1

Trabajo	Ruta	Tiempo de Procesamiento
1	4-1-3-2-5	5,3,2,4,1
2	1-3-5-2-4	7,4,1,3,2
3	5-2-1-3-4	3,4,2,1,2
4	3-2-4-5-1	3,7.5,4,2
5	1-4-2-3-5	4,2,5,1,3
6	1-4-2-5-3	3,2,1,4,3

Tabla 1. Problema con 6 trabajos y 5 máquinas

Para este ejemplo, la representación inicial sería:

M1: {2,6,5}

M2: {}

M3: {4}

M4: {1}

M5:{3}

De esta manera, sobre la máquina 1 inician tanto el trabajo 2 como el trabajo 5 y el 6, sobre la máquina 2 no inicia ningún trabajo, en la máquina 3 el trabajo 4 y en las máquinas 4 el trabajo 1, en la maquina 5 el trabajo 3.

Operador de selección

Después de generar todas las soluciones de una etapa, se evalúa el desempeño una por una, se ordenan de la mejor a la peor y se eligen las *S* mejores soluciones, que continuarán en la siguiente etapa

Operador de cruce

Con los mejores individuos seleccionados, se realiza la operación de cruce, la cual consiste en elegir de a dos soluciones, y generar un nuevo individuo considerando un solo punto de cruce, el cual se encuentra en el punto medio de la representación de cada solución, de tal manera que por torneo binario se elige la permutación que tendrá el nuevo individuo en cada una de las máquinas del problema. A continuación, se muestra un ejemplo de dicho proceso:

Solución 1:

M1: {3,7,1,4}, M2: {9,2,8}, M3: {6,5,10}

Solución 2:

M1: {1,4,7,3}, M2: {8,2,9}, M3: {10,5,6}

Nueva solución:

M1: {3,7,1,4}, M2: {8,2,9}, M3: {6,5,10}

Operador de mutación

Finalmente, se completa la población de individuos con soluciones nuevas generadas de manera aleatoria con el fin de introducir diversidad en el espacio de búsqueda de soluciones.

Desarrollo computacional

Para el desarrollo de este proyecto de grado, se utilizaron diversas tecnologías, herramientas, y buenas prácticas que permitieron abordar los retos técnicos con eficiencia y mantener altos estándares de calidad. A continuación, se describen los principales elementos utilizados:

Tecnologías y Herramientas

- **Lenguaje de programación**: Java fue elegido como lenguaje principal por su robustez, versatilidad y enfoque en la Programación Orientada a Objetos (POO). La POO permitió estructurar el proyecto de manera modular, fomentando la reutilización del código, el encapsulamiento y el polimorfismo, lo que facilitó la creación de componentes flexibles y fáciles de mantener.

  Su característica multiplataforma garantizó que el sistema pudiera ejecutarse en diferentes entornos sin necesidad de modificar el código, gracias a la Máquina Virtual de Java (JVM). Además, el manejo automático de memoria mediante el Garbage Collector fue esencial para procesar eficientemente millones e incluso billones de registros, evitando problemas de rendimiento. Java destaca por su seguridad integrada y su gran comunidad, lo que asegura un entorno confiable y soporte a largo plazo para futuras mejoras.

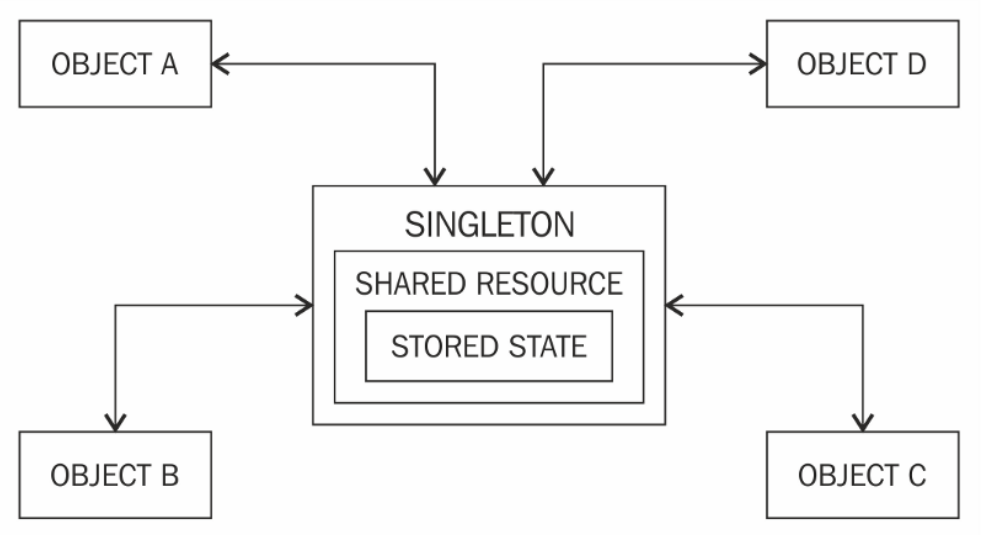
- **IntelliJ IDEA**: IntelliJ IDEA fue seleccionado como el Entorno de Desarrollo Integrado (IDE) por su potente conjunto de herramientas y su capacidad para optimizar el flujo de trabajo. Su autocompletado inteligente y análisis en tiempo real permitieron detectar errores y sugerir mejoras desde etapas tempranas, garantizando un desarrollo más ágil y eficiente.

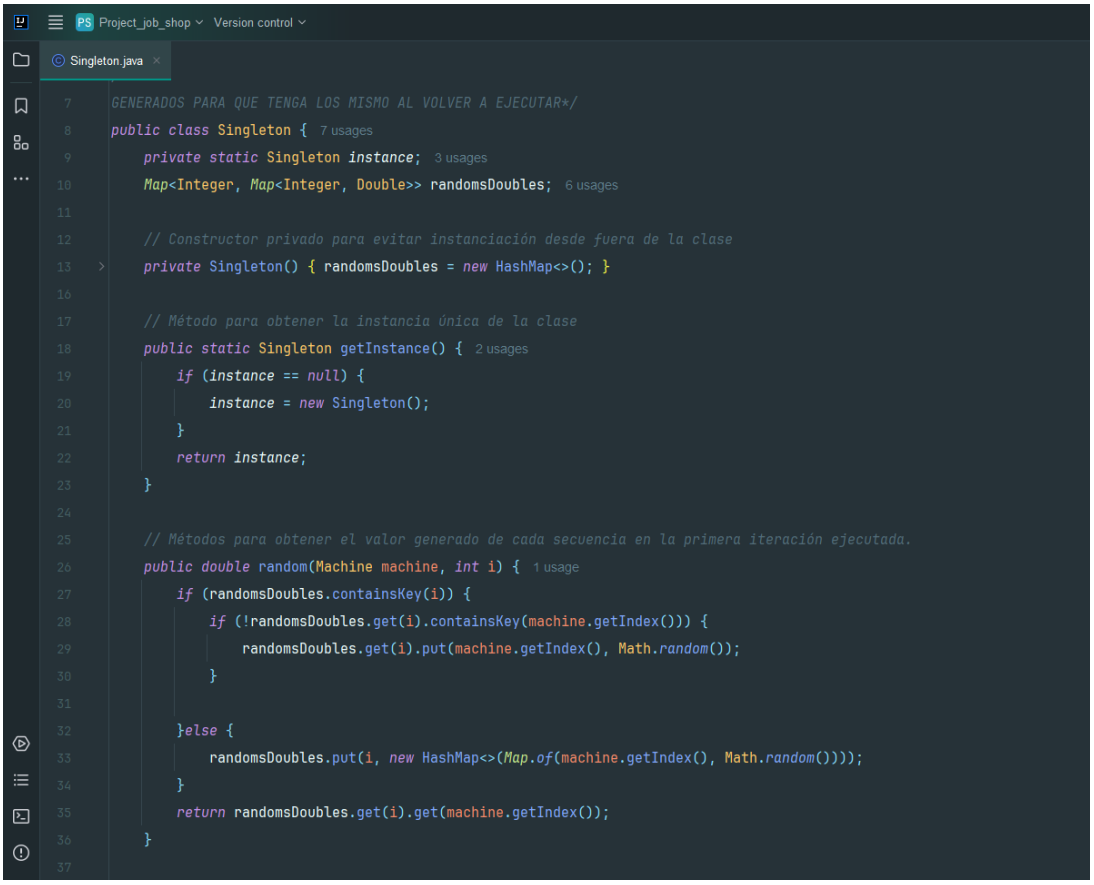
El IDE facilitó la gestión del proyecto mediante integración nativa con sistemas de control de versiones, permitiendo llevar un registro detallado de los cambios. Además, las funcionalidades avanzadas para pruebas y depuración permitieron identificar y resolver errores rápidamente, mejorando la calidad del código.

Por otro lado, IntelliJ ofreció una excelente compatibilidad con Java y soporte para plugins, lo que potenció el desarrollo al añadir funcionalidades personalizadas según las necesidades del proyecto.

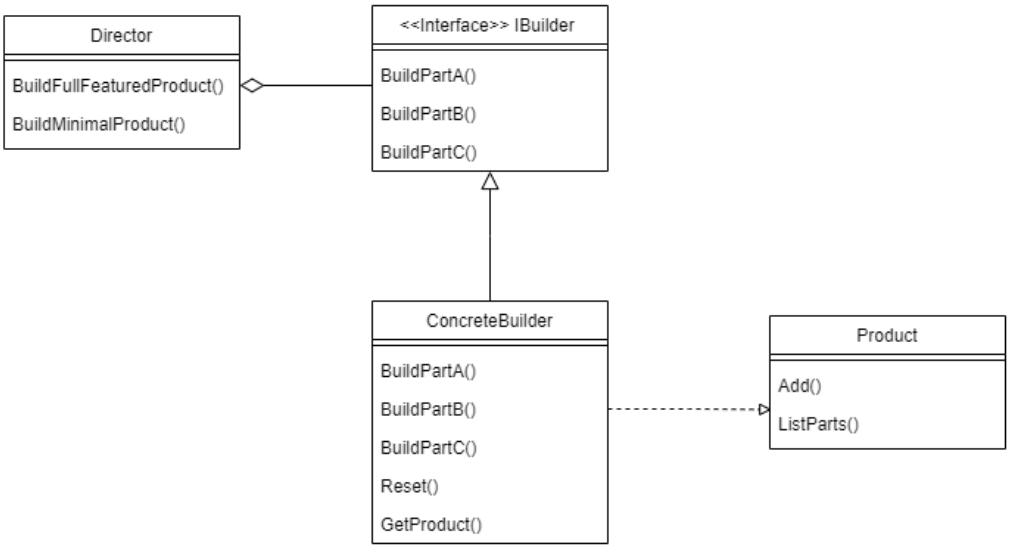
Patrones de Diseño

- **Singleton**: Implementado para asegurar que ciertas clases solo tengan una instancia global accesible, evitando problemas de concurrencia.



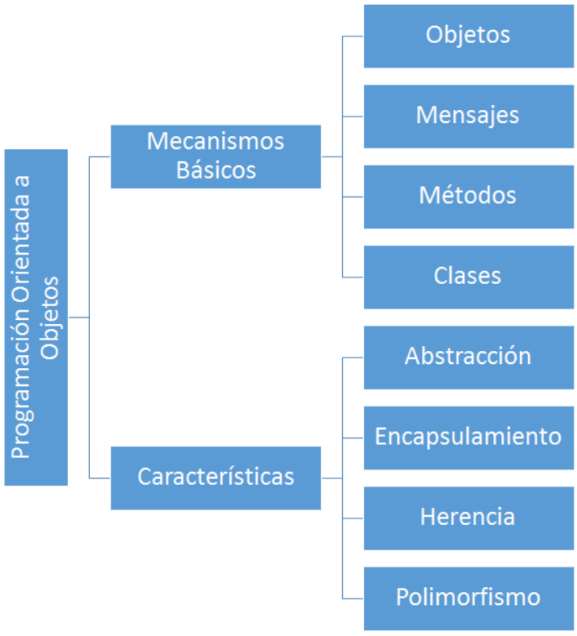

Project_job_shop
Version control
Singleton.java
GENERADOS PARA QUE TENGA LOS MISMO AL VOLVER A EJECUTAR*/
public class Singleton { 7 usages
private static Singleton instance; 3 usages
Map<Integer, Map<Integer, Double>> randomsDoubles; 6 usages
// Constructor privado para evitar instanciación desde fuera de la clase
private Singleton() { randomsDoubles = new HashMap<>(); }
// Método para obtener la instancia única de la clase
public static Singleton getInstance() { 2 usages
if (instance == null) {
instance = new Singleton();
}
return instance;
}
// Métodos para obtener el valor generado de cada secuencia en la primera iteración ejecutada.
public double random(Machine machine, int i) { 1 usage
if (randomsDoubles.containsKey(i)) {
if (!randomsDoubles.get(i).containsKey(machine.getIndex())) {
randomsDoubles.get(i).put(machine.getIndex(), Math.random());
}
}else {
randomsDoubles.put(i, new HashMap<>(Map.of(machine.getIndex(), Math.random())));
}
return randomsDoubles.get(i).get(machine.getIndex());
}

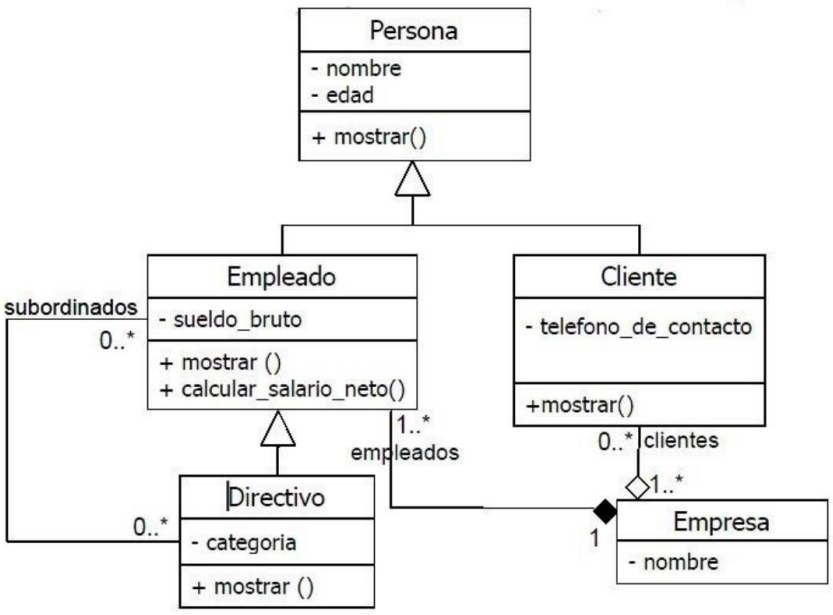
- **Builder**: Utilizado para la creación controlada de objetos complejos, garantizando la inmutabilidad y facilitando la construcción paso a paso de instancias con múltiples parámetros.



Buenas Prácticas de Desarrollo

- **Programación orientada a objetos**: Favoreciendo la reutilización de código, encapsulamiento y abstracción.



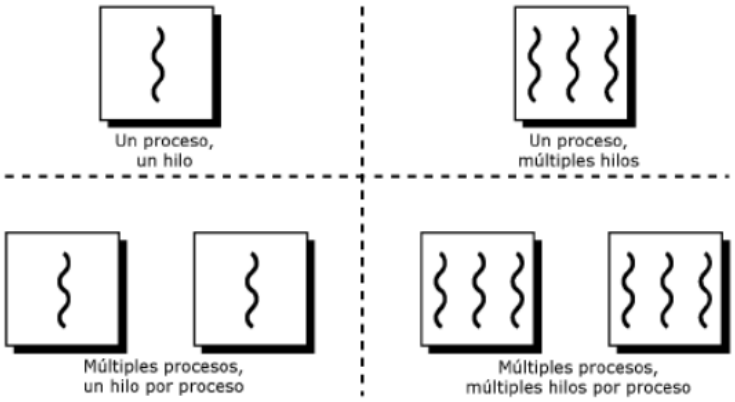


- **Control de versiones**: Uso de buenas prácticas para gestionar versiones del proyecto, facilitando el trabajo colaborativo y la trazabilidad de cambios.

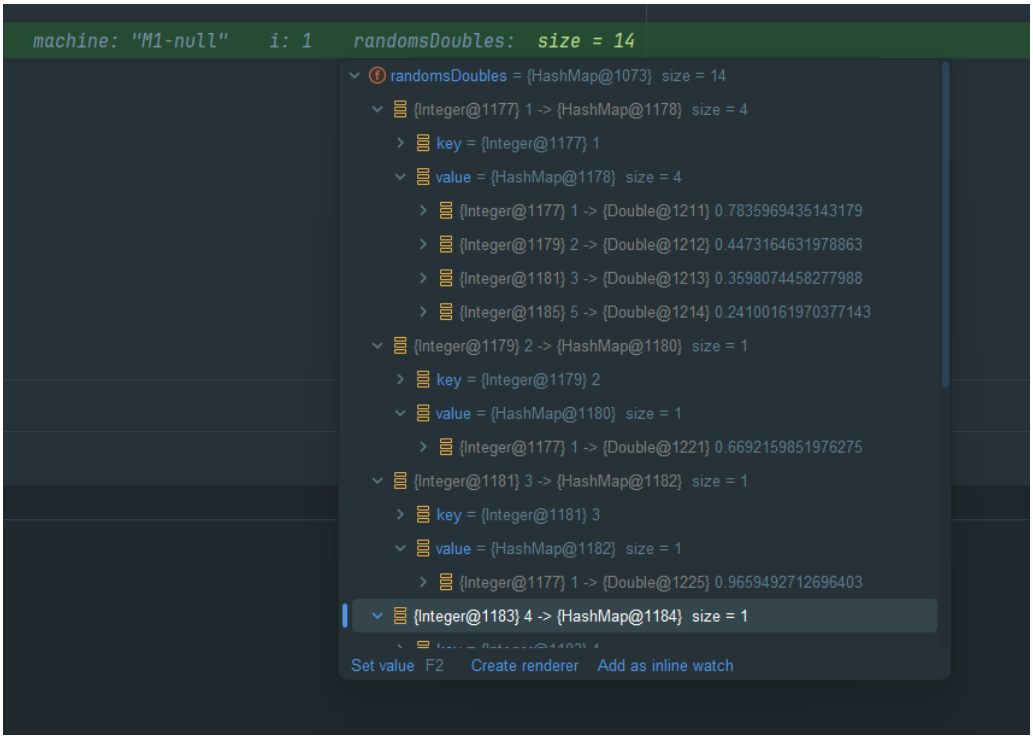
- **Pruebas unitarias y de integración**: Asegurando que cada módulo funcione correctamente de manera individual y en conjunto.

Conocimientos Avanzados de Programación

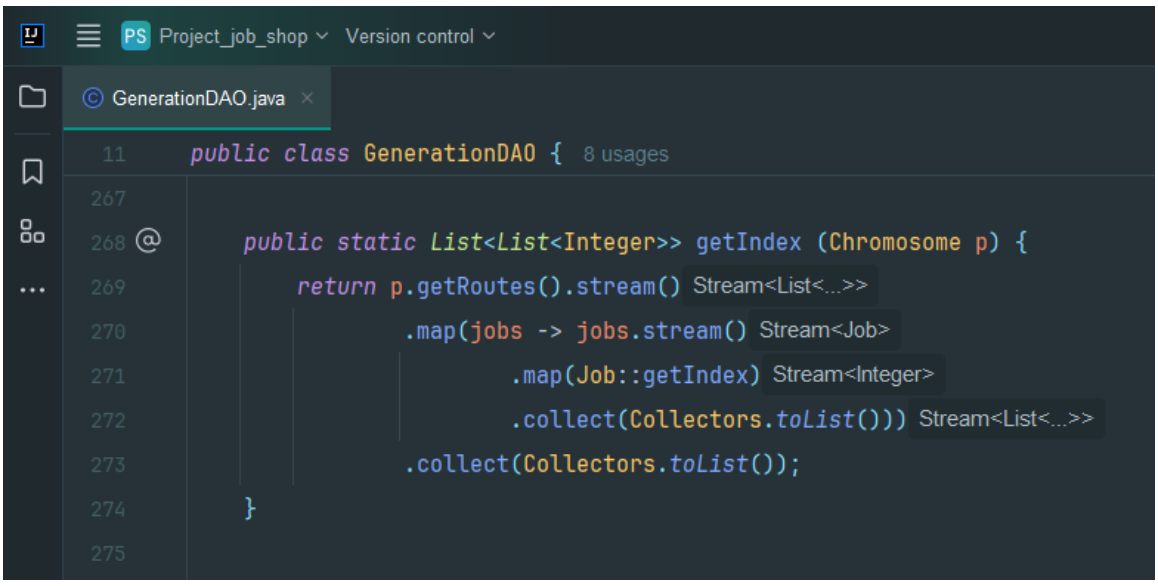
- **Manejo de hilos**: Implementación de programación concurrente para maximizar el rendimiento en operaciones paralelas y evitar bloqueos.



- **Diccionarios (HashMaps)**: Utilizados para gestionar eficientemente grandes volúmenes de datos clave-valor.



- **Streams**: Aplicación de streams en Java para procesar flujos de datos de forma eficiente, facilitando la manipulación y agregación de datos a gran escala.



Procesamiento de Datos Masivos

El proyecto también requirió trabajar tanto con miles de datos como cientos de millones de datos. Se aplicaron técnicas avanzadas de programación y optimización, como la paralelización del procesamiento mediante hilos y el uso de streams, garantizando así un rendimiento adecuado para el manejo de este volumen de información.

Con la combinación de estas tecnologías, patrones y prácticas, se logró un proyecto sólido y eficiente, alineado con los estándares actuales del desarrollo de software.

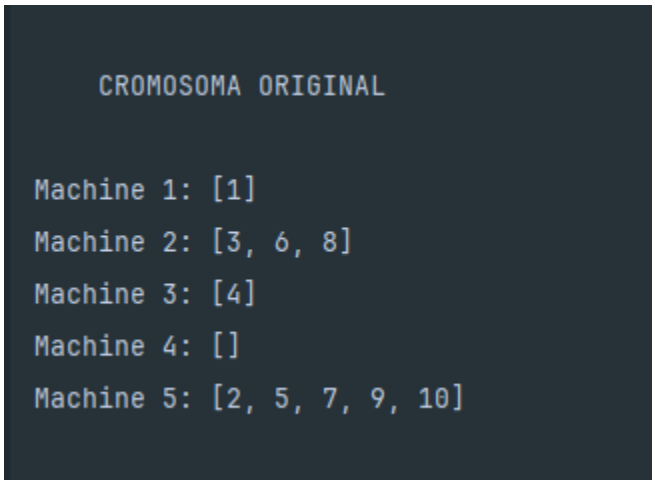
Conclusiones

El desarrollo de este proyecto de grado permitió alcanzar los objetivos propuestos y brindar soluciones eficientes a los desafíos planteados.

Las pruebas realizadas demostraron que la solución desarrollada supera significativamente las alternativas existentes, tanto en términos de rendimiento como de eficiencia.

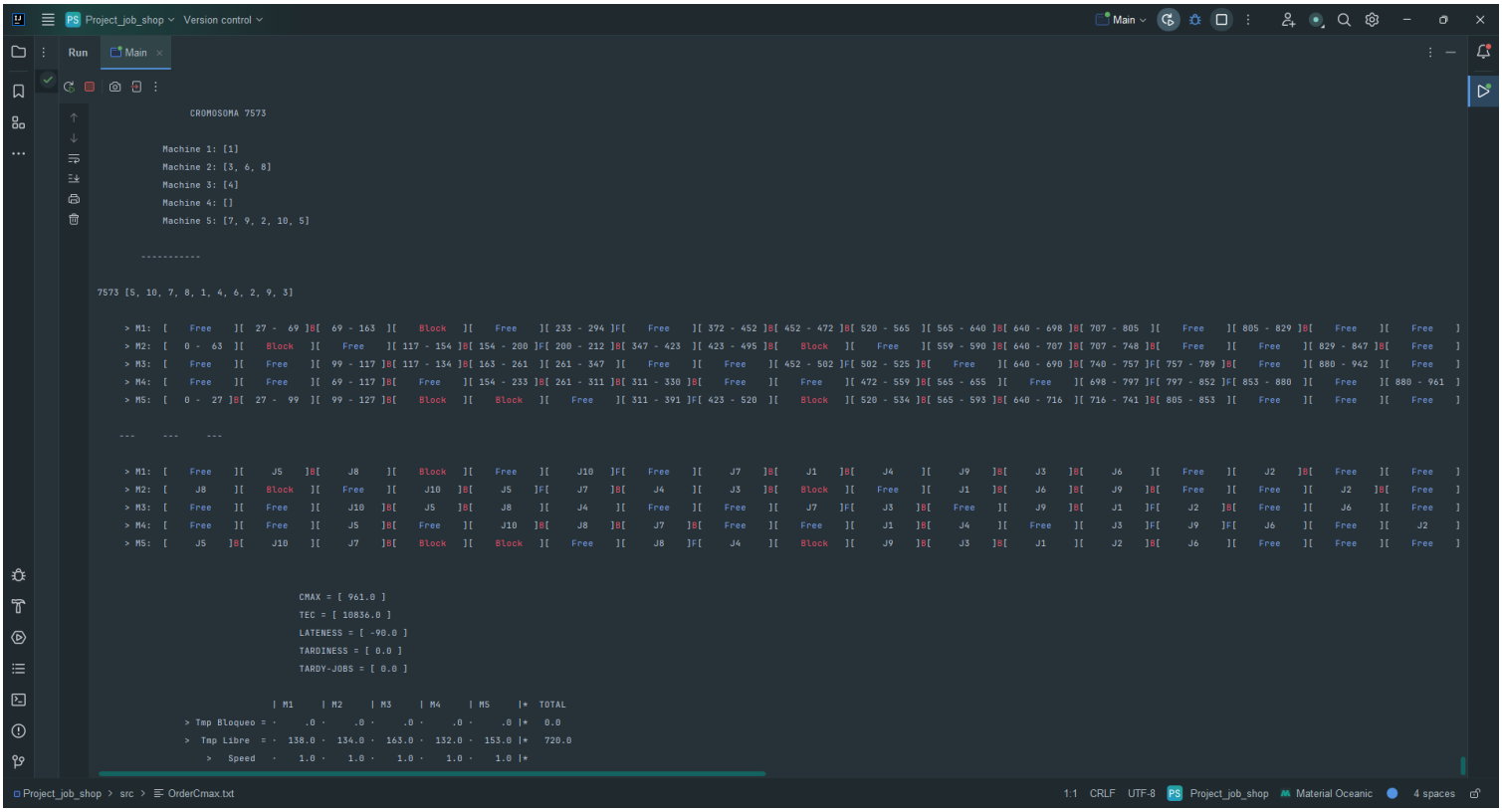
Se evidenció una clara optimización en procesos que anteriormente presentaban soluciones menos favorables, mejorando tiempos de ejecución y reduciendo el consumo de recursos.

Problema de ejemplo.



Soluciones encontradas.

Visualización de Cromosoma #7573



Referencias

Aitzai, A., Benmedjdoub, B., & Boudhar, M. (2012). A branch and bound and parallel genetic algorithm for the job shop scheduling problem with blocking. In *Int. J. Operational Research* (Vol. 14, Issue 3).

Alvarez, O. D. G., Larrea, N. P. L., & Valencia, M. V. R. (2022). Análisis comparativo de Patrones de Diseño de Software. Polo del Conocimiento: Revista científico-profesional, 7(7), 2146-2165.

Argente Villaplana, E. (2017). CREACIÓN DE HILOS EN JAVA.

Bastidas, L.R. (2007). El inicio del siglo XXI. Planeta. Sitio web: http://www.rbastidasl.com/libro-inicio-del-sigloxxi.

Borges, J.L. (2013). Ficciones. Buenos Aires, Argentina: Debolsillo.

Brizuela, C. A., Zhao, Y., & Sannomiya, N. (2001). No-wait and blocking job-shops: Challenging problems for GA's. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, *4*, 2349–2354. https://doi.org/10.1109/ICSMC.2001.972908

Brucker, P., Heitmann, S., Hurink, J., & Nieberg, T. (2005). *Job-shop scheduling with limited capacity buffers*.

Brucker, P., & Kampmeyer, T. (2008). Cyclic job shop scheduling problems with blocking. *Annals of Operations Research*, *159*(1), 161–181. https://doi.org/10.1007/s10479-007-0276-z

Dabah, A., Bendjoudi, A., & Aitzai, A. (2017). An efficient tabu search neighborhood based on reconstruction strategy to solve the blocking job shop scheduling problem. *Journal of Industrial and Management Optimization*, *13*(4), 2015–2031. https://doi.org/10.3934/jimo.2017029

Dabah, A., Bendjoudi, A., AitZai, A., El-Baz, D., & Taboudjemat, N. N. (2018). Hybrid multi-core CPU and GPU-based B&B approaches for the blocking job shop scheduling problem. *Journal of Parallel and Distributed Computing*, *117*, 73–86. https://doi.org/10.1016/j.jpdc.2018.02.005

Gholami, O., & Törnquist Krasemann, J. (2018). A heuristic approach to solving the train traffic re-scheduling problem in real time. Algorithms, 11(4), 55.

Gröflin, H., & Klinkert, A. (2006). Feasible insertions in job shop scheduling, short cycles and stable sets. *European Journal of Operational Research*, *177*(2), 763–785. https://doi.org/10.1016/j.ejor.2005.12.025

Gröflin, H., & Klinkert, A. (2009). A new neighborhood and tabu search for the Blocking Job Shop. *Discrete Applied Mathematics*, *157*(17), 3643–3655. https://doi.org/10.1016/j.dam.2009.02.020

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, *44*(3), 510-525.

Hayasaka, T., & Hino, R. (2012). Multiple exchanges of job orders for no-buffer job shop scheduling problem. *Journal of Advanced Mechanical Design, Systems and Manufacturing*, *6*(5), 661–671. https://doi.org/10.1299/jamdsm.6.661

Hayasaka, T., & Hino, R. (2016). Optimization problem for feasibility evaluation of schedules considering blocking. *Journal of Advanced Mechanical Design, Systems and Manufacturing*, *10*(2). https://doi.org/10.1299/jamdsm.2016jamdsm0016

Heger, J., & Voss, T. (2018). Optimal Scheduling of AGVs in a Reentrant Blocking Job-shop. *Procedia CIRP*, *67*, 41–45. https://doi.org/10.1016/j.procir.2017.12.173

Hou, P., Wang, D., & Li, X. (2012, May). An improved harmony search algorithm for blocking job shop to minimize makespan. In Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 763-768). IEEE.

Lakehal, M., Aissani, A., & Bouibede, K. (2015). New algorithm of scheduling based on prioriry rules on machines to solve bi-criteria blocking job shop scheduling problem. *12th International Symposium on Programming and Systems, ISPS 2015*, 149–156. https://doi.org/10.1109/ISPS.2015.7244978

Lange, J., & Werner, F. (2018a). *A Permutation-Based Neighborhood for the Blocking Job-Shop Problem with Total Tardiness Minimization* (pp. 581–586). https://doi.org/10.1007/978-3-319-89920-6_77

Lange, J., & Werner, F. (2018b). Approaches to modeling train scheduling problems as job-shop problems with blocking constraints. *Journal of Scheduling*, *21*(2), 191–207. https://doi.org/10.1007/s10951-017-0526-0

Lange, J., & Werner, F. (2019). On neighborhood structures and repair techniques for blocking job shop scheduling problems. *Algorithms*, *12*(11). https://doi.org/10.3390/a12110242

Liu, S. Q., & Kozan, E. (2009). Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers and Operations Research*, *36*(10), 2840–2852. https://doi.org/10.1016/j.cor.2008.12.012

Liu, S. Q., & Kozan, E. (2017). A hybrid metaheuristic algorithm to optimise a real-world robotic cell. *Computers and Operations Research*, *84*, 188–194. https://doi.org/10.1016/j.cor.2016.09.011

Liu, S. Q., Kozan, E., Masoud, M., Zhang, Y., & Chan, F. T. S. (2018). Job shop scheduling with a combination of four buffering constraints. *International Journal of Production Research*, *56*(9), 3274–3293. https://doi.org/10.1080/00207543.2017.1401240

López Vargas, J. C. (2013). Metodología de programación de producción en un flow shop híbrido flexible con el uso de algoritmos genéticos para reducir el makespan: aplicación en la industria textil. Departamento de Ingeniería Industrial.

López, L. (2013). Metodología de la programación orientada a objetos. Alpha Editorial.

Louaqad, S. (n.d.). *A disjunctive graph for the Job Shop Problem with Transport and blocking no wait constraints Oulaid KAMACH Laboratoire des technologies innovantes*.

Louaqad, S., Kamach, O., & Iguider, A. (2018). SCHEDULING FOR JOB SHOP PROBLEMS WITH TRANSPORTATION AND BLOCKING NO-WAIT CONSTRAINTS. *Journal of Theoretical and Applied Information Technology*, *31*(10). www.jatit.org

Mascis, A., & Pacciarelli, D. (2000). Machine scheduling via alternative graphs.

Mascis, A., & Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. European Journal of Operational Research, 143(3), 498-517.

Masoud, M., Kozan, E., & Kent, G. (2011). A job-shop scheduling approach for optimising sugarcane rail operations. *Flexible Services and Manufacturing Journal*, *23*(2), 181–206. https://doi.org/10.1007/s10696-011-9092-5

Mati, Y., Rezg, N., & Xie, X. (2001). *Scheduling Problem of Job-Shop with Blocking: A Taboo Search Approach*.

Mejía, G., & Montoya, C. (2009). Scheduling manufacturing systems with blocking: A Petri net approach. *International Journal of Production Research*, *47*(22), 6261–6277. https://doi.org/10.1080/00207540802225983

Mogali, J. K., Barbulescu, L., & Smith, S. F. (2021). Efficient primal heuristic updates for the blocking job shop problem. *European Journal of Operational Research*, *295*(1), 82–101. https://doi.org/10.1016/j.ejor.2021.02.051

Oddi, A., Rasconi, R., Cesta, A., & Smith, S. F. (2012, May). Iterative improvement algorithms for the blocking job shop. In Twenty-second international conference on automated planning and scheduling.

Pham, D. N., & Klinkert, A. (2008). Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, *185*(3), 1011–1025. https://doi.org/10.1016/j.ejor.2006.03.059

Pinedo, M. L. (2012). Scheduling (Vol. 29). New York: Springer.

Sauvey, C., Trabelsi, W., & Sauer, N. (2020). Mathematical model and evaluation function for conflict-free warranted makespan minimization of mixed blocking constraint job-shop problems. *Mathematics*, *8*(1). https://doi.org/10.3390/math8010121

Slack, N., Chambers, S., & Johnston, R. (2010). Operations management. Pearson education.

Yalaoui, N., Mahdi, H., Amodeo, L., & Yalaoui, F. (2011, March). A particle swarm optimization under fuzzy logic controller to solve a scheduling problem. In 2011 International Conference on Communications, Computing and Control Applications (CCCA) (pp. 1-6). IEEE.

Zeng, C., Tang, J., & Yan, C. (2014). Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles. *Applied Soft Computing Journal*, *24*, 1033–1046. https://doi.org/10.1016/j.asoc.2014.08.028

Zeng, C., & Tang, J. (2014, June). Blocking job shop cell scheduling with automated guided vehicles. In Proceeding of the 11th World Congress on Intelligent Control and Automation (pp. 438-442). IEEE.

Anexos