Desarrollo de un algoritmo de búsqueda tabú para la programación de pedidos en manufactura aditiva

Paula Andrea Chacón Santamaría

Trabajo de Grado para optar al título de Ingeniera industrial

Director

Néstor Raúl Ortiz Pimiento

PhD en Ingeniería

Codirectora

Lina Mayerly Lozano Suarez

Magister en Ingeniería Industrial

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Estudios Industriales y Empresariales

Ingeniería Industrial

Bucaramanga

2023

Dedicatoria

A mi nonita Herminda

Agradecimientos

A mis padres por brindarme todo y acompañarme en este proceso.

A la profesora Lina por su gran ayuda y guía en el desarrollo del proyecto.

Al profesor Néstor, al grupo OPALO, a la Universidad Industrial de Santander.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Los apéndices están adjuntos y puede visualizarlos en la base de datos de la biblioteca UIS.

Resumen

Título: Desarrollo de un algoritmo de búsqueda tabú para la programación de pedidos en manufactura aditiva[*]

Autor: Paula Andrea Chacón Santamaría[**]

Palabras Clave: Manufactura aditiva, programación de la producción, búsqueda tabú, makespan

Descripción: La fabricación o manufactura aditiva (AM), también conocida como impresión 3D, es un proceso de producción en el que se construyen objetos capa por capa a partir de sus modelos digitales tridimensionales. En esta industria, los tiempos de impresión suelen ser muy elevados, razón por la cual, las decisiones para programar de la producción son de vital importancia. En esta investigación se utilizó un algoritmo de búsqueda tabú (TS) para resolver el problema de programación de pedidos en un ambiente de múltiples máquinas AM diferentes, el cual es de tipo NP-hard. Por tanto, las técnicas metaheurísticas son apropiadas para su solución. La función objetivo fue la minimización del makespan. El algoritmo TS fue probado con instancias de la literatura. En las instancias de menor tamaño mostró ser una buena opción para la resolución de este problema. Al ser comparado con el método exacto, logró una disminución del tiempo de cómputo promedio de 1020,69 a 0,48 segundos, mientras que los resultados del makespan distaron en promedio un 2,45% de los obtenidos mediante CPLEX. Además, al compararlo con otras metaheurísticas como Búsqueda Local Iterada de Aprendizaje por Refuerzo (ILS+Q-Learning) y un algoritmo evolutivo (EA) de la literatura, la diferencia relativa promedio fue de 0,70% y 0,75% en favor del TS, respectivamente. En instancias de mayor tamaño el algoritmo TS mostró un menor rendimiento en comparación con la ILS+Q-Learning y el EA, ya que estas diferencias fueron de 3,02% y 3,01%, respectivamente a favor de estos últimos.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Estudios Industriales y Empresariales. Ingeniería Industrial. Director: Néstor Raúl Ortiz Pimiento. PhD en Ingeniería. Codirectora: Lina Mayerly Lozano Suarez. Magister en Ingeniería Industrial.

Abstract

Title: A tabu search algorithm for additive manufacturing scheduling[*]

Author(s): Paula Andrea Chacón Santamaría[**]

Keywords: Additive manufacturing, scheduling, tabu search, makespan

Description: Additive manufacturing (AM), also known as 3D printing, is a production process in which objects are built layer by layer from their three-dimensional digital models. In this industry, printing times are usually very high, which means decisions to schedule production are of vital importance. In this research, a tabu search (TS) algorithm was used to solve the scheduling problem in an environment of multiple different AM machines, which is NP-hard. Therefore, metaheuristic techniques are appropriate for its solution. The objective function was the minimization of the makespan. The TS algorithm was tested with instances from the literature. In smaller instances, it proved to be a good option for solving this problem. When compared with the exact method, it achieved a decrease in average computation time from 1020.69 to 0.48 seconds, while the makespan results were on average 2.45% different from those obtained using CPLEX. Furthermore, when compared with other metaheuristics such as Reinforcement Learning Iterated Local Search (ILS+Q-Learning) and an Evolutionary Algorithm (EA) from literature, the average relative difference was 0.70% and 0.75% in favor of the TS, respectively. In larger instances, the TS algorithm showed lower performance compared to ILS+Q-Learning and EA, since these differences were 3.02% and 3.01%, respectively, in favor of the latter.

[*] Degree Work

[**]Faculty of Physical-Mechanical Engineering. School of Industrial and Business Studies. Industrial engineering. Director: Nestor Raúl Ortiz Pimiento. PhD in Engineering. Co-director: Lina Mayerly Lozano Suarez. Master in Industrial Engineering.

Introducción

En la presente investigación se realizó el diseño un algoritmo de búsqueda tabú (TS) para la programación de pedidos en manufactura aditiva (AM) o impresión 3D con el objetivo de minimizar el makespan. La AM es una tecnología cuyo uso ha ido creciendo en los últimos años, y se ha pronosticado que la cuota de mercado de los artículos producidos mediante AM frente a los artículos producidos de forma convencional será significativa en todas las industrias en pocos años (Jiang et al., 2017). El consecuente aumento del número de pedidos que un proveedor de servicios de impresión debe producir debido a esta situación, genera complicaciones al momento de determinar cómo organizar la producción.

Usualmente, la programación se hace por lotes, los cuales pueden contener piezas iguales o diferentes. Generalmente se le denomina *job* (o *build*) a este conjunto de piezas, las cuales se procesan simultáneamente en una máquina AM. Como el tiempo de procesamiento de un *job* depende de las piezas que fueron asignadas, estas decisiones tienen importantes repercusiones en los tiempos de entrega y los costos de los pedidos.

En los problemas de programación en AM (*Additive Manufacturing Scheduling Problems*, AMSP) se busca la óptima organización de la producción, es decir, de los *jobs*. Existen diferentes configuraciones de las máquinas para este problema como: una sola máquina, múltiples máquinas iguales en paralelo o múltiples máquinas diferentes en paralelo.

Para este problema ya han sido propuestos diversos procedimientos heurísticos y metaheurísticos como optimización de enjambre de partículas (PSO), algoritmo competitivo imperialista (ICA), algoritmo genético (Kim & Kim, 2022), Recocido simulado (Che et al., 2021), Búsqueda Local Iterada de Aprendizaje por Refuerzo (Alicastro et al., 2021), entre otros.

Asimismo, la TS se ha usado para dar solución al problema en un ambiente de una sola máquina (Fera et al., 2020).

La TS es una de las metaheurísticas más utilizadas por su destacable éxito para resolver problemas de optimización duros. Sin embargo, este algoritmo ha sido poco explorado para dar solución a este problema para varias máquinas. En consecuencia, esta investigación permitirá observar y comparar el desempeño del algoritmo TS para resolver el problema de programación para un entorno de múltiples máquinas AM diferentes, el cual es un problema NP-hard según Kucukkoc, (2019).

Por consiguiente, en este trabajo se desarrolló un algoritmo TS para dar solución a esta problemática, el cual se validó mediante pruebas con instancias de la literatura. Este documento está estructurado de la siguiente manera: En las primeras dos secciones se describe el problema y se plantean los objetivos del estudio. En la sección 3, se realiza un análisis bibliométrico y una revisión de literatura del problema en cuestión. En la sección 4, se presentan antecedentes, es decir, otros trabajos de grado relacionados con el tema, y se expone la fundamentación teórica. En la sección 5, se introduce el modelo matemático base usado y se describe el procedimiento utilizado para construir y validar el algoritmo. En la sección 6, se exponen los resultados del makespan obtenido con el algoritmo. Finalmente, se presentan conclusiones y recomendaciones.

Tabla 1.

Cumplimiento de objetivos del proyecto.

Objetivo	**Cumplimiento**
1. Realizar una revisión de literatura sobre el problema de programación de pedidos en manufactura aditiva.	Sección 3.
2. Establecer un modelo matemático que represente el problema de programación de pedidos en manufactura aditiva para minimización del makespan.	Sección 5.1.

Objetivo	Cumplimiento
3. Diseñar un algoritmo de búsqueda tabú que dé solución al modelo definido.	Apéndice A.
4. Validar el algoritmo propuesto.	Sección 6.
5. Escribir un artículo de carácter publicable.	Apéndice F.

1. Planteamiento del problema

La fabricación o manufactura aditiva (AM), también conocida como impresión 3D, es un proceso de producción en el que se construyen objetos capa por capa a partir de sus modelos digitales tridimensionales. Principalmente se utiliza para fabricar piezas en plástico o metal, y emplea diferentes tecnologías para esto. Algunas de ellas son: estereolitografía (SLA), modelado por deposición fundida (FDM), sinterización selectiva por láser (SLS), fusión por haz de electrones (EBM), entre otras. Cada una de estas tecnologías utiliza diferentes métodos para la creación de la pieza, y los materiales que se pueden utilizar dependen de la tecnología escogida, e incluso también se puede imprimir en materiales compuestos. La versatilidad y flexibilidad de diseño que posibilita la AM, la han puesto como uno de los pilares de la industria 4.0. Además, se ha pronosticado que, en 2030, la cuota de mercado de los artículos producidos mediante AM (productos, componentes) frente a los artículos producidos de forma convencional será significativa (> 10%) en todas las industrias (Jiang et al., 2017).

Recientemente, ha aumentado el interés en la planificación de la producción específicamente para AM, dadas sus particulares condiciones. Usualmente, la programación se hace por lotes, los cuales pueden contener piezas iguales o diferentes. Generalmente se le denomina *job* (o *build*) a este conjunto de piezas, las cuales se procesan simultáneamente en una

máquina AM. Una vez que se inicia el *job*, no se puede agregar o quitar ninguna pieza de la máquina hasta que se complete todo. El tiempo de procesamiento de un *job* puede verse como la composición de dos secciones: el tiempo dedicado a la preparación del *job* y la recolección de piezas, que es relativamente fijo, y el tiempo dedicado a la impresión de las piezas asignadas al *job*, que es una función de los atributos de las piezas (como su volumen y dimensiones), así como de las especificaciones de la máquina (Li et al., 2019). La diferencia del costo de producción por volumen de material puede ser superior al 40% dependiendo del *job* al que se asigne la pieza (Li et al., 2017).

A estos problemas en los que se busca organizar de la mejor manera la producción, es decir, los *jobs* se les denomina problemas de programación en AM (Additive Manufacturing Scheduling Problems, AMSP). Según Kucukkoc (2019) este problema es de tipo NP-hard ya que el tiempo de procesamiento de un *job* es una función y, adicionalmente, en máquinas paralelas no idénticas, las impresoras soportan alturas diferentes, lo que afecta mucho la combinación de piezas asignadas a las máquinas. Por tanto, dada la dificultad del problema, las técnicas metaheurísticas son apropiadas para su solución.

Los algoritmos de búsqueda tabú (TS) han sido poco explorados para dar solución a este problema. Una característica distintiva de este procedimiento metaheurístico es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas. Además, el destacable éxito de la búsqueda tabú para resolver problemas de optimización duros (especialmente aquellos que surgen en aplicaciones del mundo real) ha causado una explosión de nuevas aplicaciones TS (Glover & Melián, 2003). En consecuencia, esta investigación permite observar y comparar el desempeño del algoritmo TS para resolver el problema de programación de pedidos para un entorno de múltiples maquinas AM diferentes.

2. Objetivos

2.1. Objetivo General

Diseñar un algoritmo de Búsqueda Tabú para la programación de pedidos en manufactura aditiva para minimización del makespan.

2.2. Objetivos Específicos

Realizar una revisión de literatura sobre el problema de programación de pedidos en manufactura aditiva.

Establecer un modelo matemático que represente el problema de programación de pedidos en manufactura aditiva para minimización del makespan.

Diseñar un algoritmo de búsqueda tabú que dé solución al modelo definido.

Validar el algoritmo propuesto.

Escribir un artículo de carácter publicable.

3. Revisión de literatura

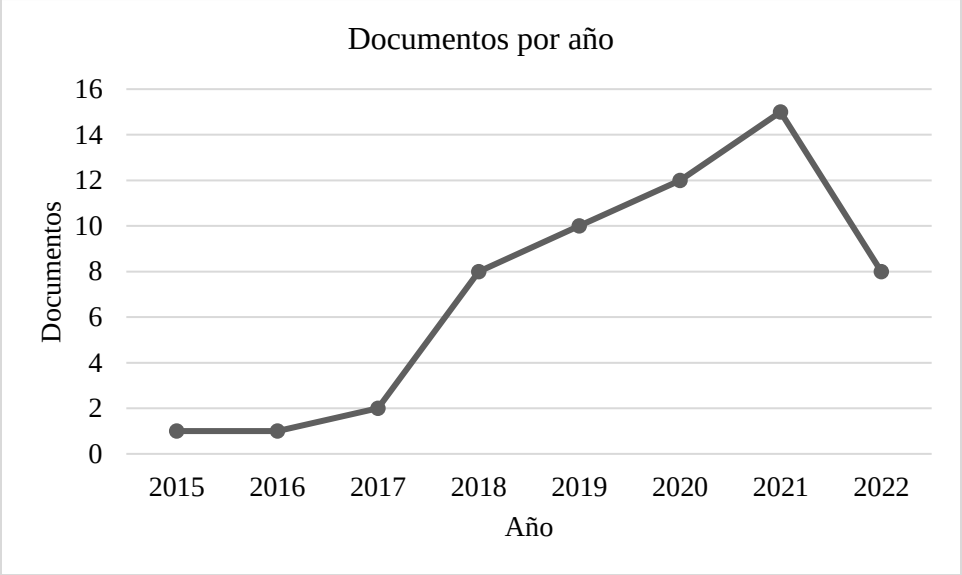
3.1. Análisis bibliométrico

Para la realización del análisis bibliométrico, se inició definiendo la ecuación de búsqueda que fue utilizada en la base de datos Scopus. La ecuación fue la siguiente: TITLE((scheduling OR "production planning") AND ("additive manufacturing" OR "3d print*")). Como resultado de esta búsqueda se obtuvieron 57 resultados (fecha: 10/07/2022).

En la Figura 1 se muestra la evolución de la cantidad de documentos publicados en los últimos 8 años (2015-2022). Se observa que el tema de programación de la producción en manufactura aditiva es relativamente nuevo y la producción científica sobre este tópico va en aumento (teniendo en cuenta que el dato de 2022 corresponde solo a aproximadamente medio año).

Figura 1.

Documentos publicados por año según el análisis bibliométrico.



Por otro lado, en la Figura 2 y Figura 3 se presentan los datos correspondientes a los documentos publicados por autor y por país, respectivamente. El autor más representativo de esta temática es Kucukkoc con 6 publicaciones, siendo además pionero, como se menciona más adelante en la revisión de literatura. Otros autores de relevancia son Tavakkoli y Li (quien posee varias publicaciones junto a Kucukkoc). En cuanto a los países, se percibe que en donde existe mayor interés por el tema es en China, seguida por Estados Unidos, como es de esperarse. También sobresale Turquía (país donde reside Kucukkoc).

Figura 2.

Documentos publicados por autor según el análisis bibliométrico.

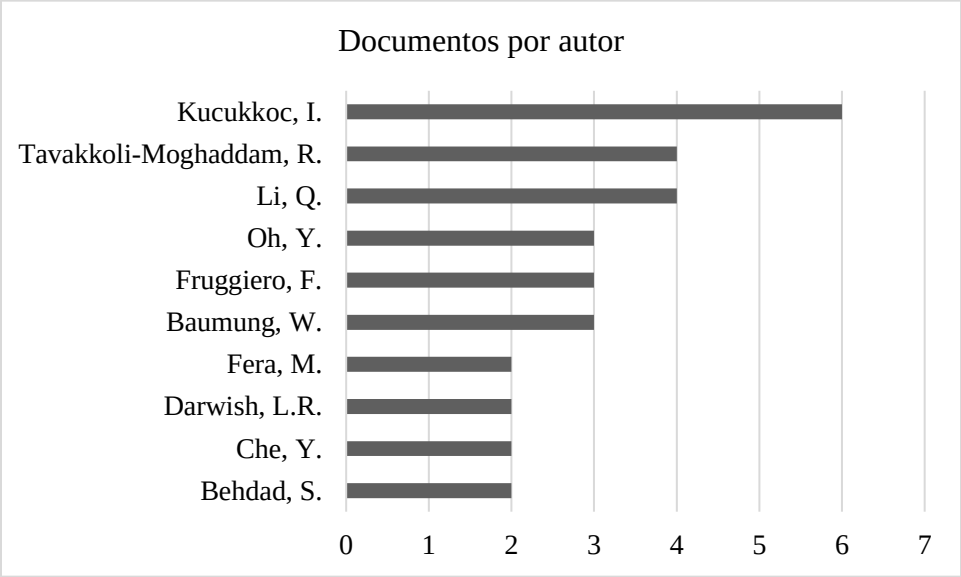
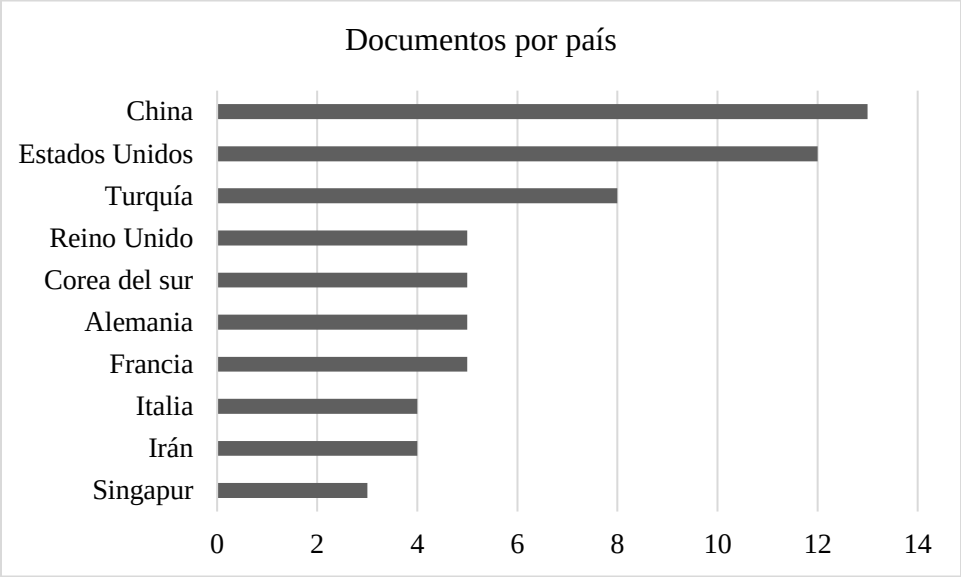


Figura 3.

Documentos publicados por país según el análisis bibliométrico.



Las áreas del conocimiento de mayor relevancia son Ciencias de la computación (29.4%), Ingeniería (28.6%), Ciencias de la decisión (15.9%) y Matemáticas (11.9%) como se ilustra en la Figura 4. Esto debido a que la resolución del problema exige la utilización de técnicas matemáticas como la programación lineal o algoritmos metaheurísticos (para mayores instancias). Adicionalmente, en la Figura 5 se muestra el mapa de conexión de palabras clave, donde las palabras más relevantes fueron scheduling, 3D printers y additive manufacturing.

Figura 4.

Documentos publicados por área del conocimiento según el análisis bibliométrico.

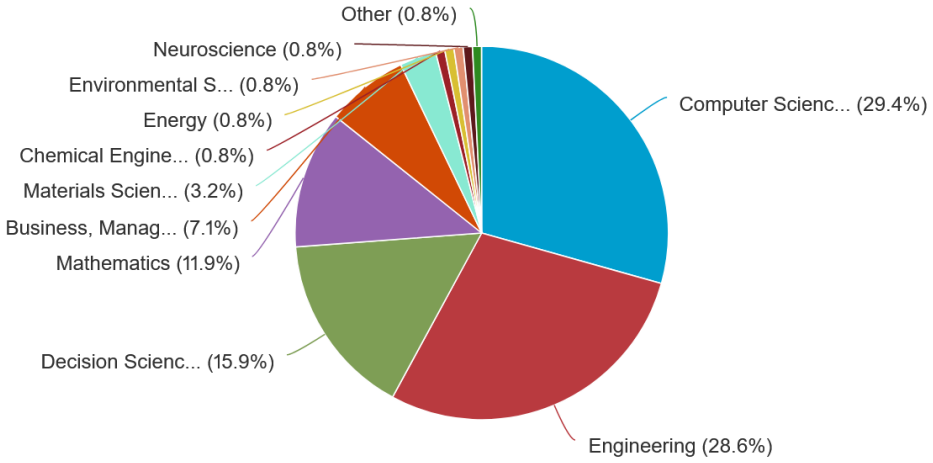
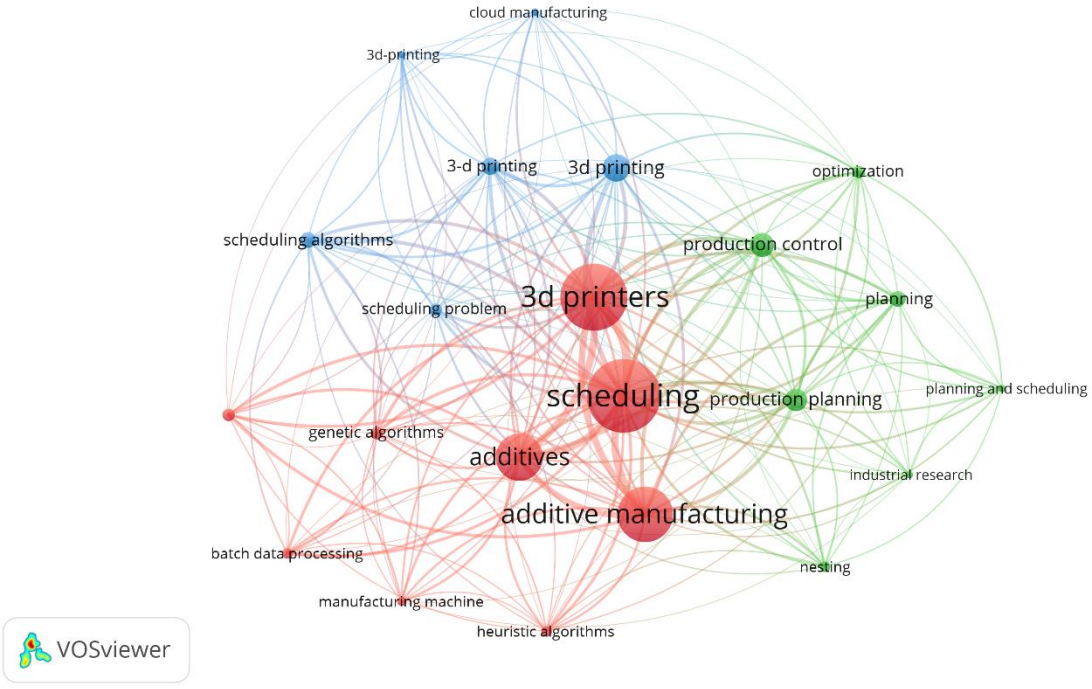


Figura 5.

Mapa de conexión de palabras clave.



Los artículos más citados se muestran en la Tabla 2. La mayoría de estos artículos fueron pioneros en el tema, ya sea porque introdujeron el problema como tal, agregaron alguna nueva consideración, o un nuevo método de solución. Además, destacan algunos de los autores mostrados en la Figura 2.

Tabla 2.

Documentos más citados según el análisis bibliométrico.

Título del documento	Autores	Año	Citaciones
Production planning in additive manufacturing and 3D printing	Li, Q., Kucukkoc, I., Zhang, D.Z.	2017	88
Production scheduling and nesting in additive manufacturing	Chergui, A., Hadj-Hamou, K., Vignat, F.	2018	63
Multi-task scheduling of distributed 3D printing services in cloud manufacturing	Zhou, L., Zhang, L., Laili, Y., Zhao, C., Xiao, Y.	2018	62
MILP models to minimise makespan in additive manufacturing machine scheduling problems	Kucukkoc, I.	2019	42
A modified genetic algorithm for time and cost optimization of an additive manufacturing single-machine scheduling	Fera, M., Fruggiero, F., Lambiase, A., Macchiaroli, R., Todisco, V.	2018	39

3.2. Análisis de literatura

La manufactura aditiva, conocida coloquialmente como impresión 3D, ha ganado progresivamente importancia no solo en diversos campos de los negocios, sino también en la vida diaria de los consumidores. La tecnología es capaz de unir varios materiales y crear objetos a partir de datos 3D, generalmente capa sobre capa en contraste con las tecnologías tradicionales de fabricación sustractiva (Jiang et al., 2017). Esto posibilita que los propios consumidores puedan diseñar sus productos y el aumento de la impresión bajo demanda impulsada digitalmente,

combinado con la flexibilidad de los pedidos online, ha aumentado en gran medida el número de pedidos que un proveedor de servicios de impresión debe producir, generando complicaciones al momento de determinar cómo organizar la producción.

A los problemas en los que se busca organizar de la mejor manera la producción, es decir, los *jobs* se les denomina problemas de programación en AM (*Additive Manufacturing Scheduling Problems*, AMSP). Un problema muy relacionado a los AMSP es el de anidamiento o agrupación (*nesting*) que consiste en organizar las diferentes piezas de un *job* dentro del espacio de construcción de la máquina. Los problemas de planificación de la producción en AM se han analizado para diferentes configuraciones de las máquinas como: una sola máquina, múltiples máquinas iguales en paralelo o múltiples máquinas diferentes en paralelo. Asimismo, se han desarrollado modelos de uno o varios objetivos.

En la literatura han planteado modelos mono-objetivo con diversos objetivos, pero cabe destacar que este problema fue abordado por primera vez por Li et al. (2017) para un ambiente de múltiples máquinas diferentes, con el objetivo de minimizar el costo de producción promedio por volumen de material para todo el sistema (incluidos todos los *jobs* en todas las máquinas). Su modelo matemático fue codificado en CPLEX y dos procedimientos heurísticos diferentes llamados reglas de "mejor ajuste" y "mejor ajuste adaptado" fueron desarrolladas en JavaScript. En otro estudio, un enfoque dinámico para la aceptación de pedidos y programación en AM para sistemas de fusión de lecho de polvo (Powder Bed Fusion, PBF) fue llevado a cabo con el objetivo de maximizar la ganancia promedio por unidad de tiempo durante todo el makespan, en este trabajo se desarrollaron dos procedimientos heurísticos basados en estrategias de toma de decisiones con el lenguaje Python para múltiples máquinas (Li et al., 2019). Karimi et al. (2021) construyeron un modelo MILP cuya función objetivo fue minimizar el costo total de electricidad para la fabricación

de piezas idénticas utilizando máquinas de modelado por deposición fundida (Fused Deposition Modeling, FDM) idénticas.

En cuanto a los trabajos que se han hecho específicamente con el objetivo de minimizar el makespan se encuentra el realizado por Kucukkoc (2019), en el que por primera vez fueron formulados modelos MILP para los casos de una sola impresora e impresoras en paralelo iguales y diferentes. También está el realizado por Alicastro et al. (2021) en el que se propone una metaheurística de Búsqueda Local Iterada de Aprendizaje por Refuerzo, basada en la implementación de una Búsqueda de Vecindario Variable de Q-Learning para múltiples máquinas de AM. Che et al. (2021) presentaron un modelo MILP y un algoritmo de recocido simulado (Simulated annealing, SA) para simultáneamente asignar piezas a *jobs*, determinar la orientación de las piezas, agrupar las piezas en la superficie bidimensional (problema de anidamiento) y asignar los *jobs* a las máquinas. Kim & Kim (2022) abordaron el problema de anidamiento de piezas y programación en AM con tiempos de alistamiento dependientes de la secuencia para máquinas en paralelo no idénticas comparando el rendimiento de tres metaheurísticas: optimización de enjambre de partículas (PSO), algoritmo competitivo imperialista (ICA) y algoritmo genético (GA).

También se han abordado enfoques multi-objetivo donde se consideran dos o tres objetivos al mismo tiempo. Fera et al. (2018) desarrollaron un Algoritmo Genético Modificado para la optimización de la anticipación/tardanza y los costos de producción de una sola máquina de fusión selectiva por láser (Selective Laser Melting, SLM). Con estos mismos objetivos, Ying et al. (2022) propusieron un Algoritmo Ajustado Iterado Codicioso (Adjusted Iterated Greedy, AIG), igualmente, para un entorno de producción de una sola máquina. Fera et al. (2020) plantearon un algoritmo de búsqueda tabú modificado para una sola máquina con un modelo bi-objetivo para

minimizar el tiempo de finalización y el costo total de los pedidos. Tafakkori et al. (2022) diseñaron, por primera vez, un modelo de optimización de tres objetivos que interpreta las ganancias, la utilización de energía de las máquinas y las pérdidas de *goodwill* (es decir, tardanza, negociación y aumento de la fecha de entrega) como tres criterios de sostenibilidad, y desarrollaron tres algoritmos: método robusto mejorado de restricción ε, algoritmo genético de clasificación no dominado (NSGA-II) y optimizador de lobo gris multiobjetivo (MOGWO) para el problema de anidamiento y programación de la producción. Rohaninejad et al. (2021) construyeron un modelo matemático bi-objetivo para un sistema AM con máquinas SLM paralelas no idénticas, considerando el makespan y la penalización por tardanza total como dos funciones objetivo. Desarrollaron un algoritmo metaheurístico híbrido mediante la combinación del algoritmo genético de clasificación no dominada (NSGA-II) con una nueva búsqueda local basada en aprendizaje apoyada en el algoritmo de agrupamiento k-means y una red neuronal de regresión. Finalmente, Toksarı & Toğa (2022) consideraron tiempos de alistamiento dependientes de la secuencia y piezas de múltiples materiales, y desarrollaron un modelo matemático MILP para una sola máquina AM para minimizar directamente el makespan e indirectamente minimizar los cambios de material.

4. Marco de referencia

4.1. Marco de antecedentes

Morera Mora (2021) analizó los factores que intervienen en el tiempo total de fabricación en una sola máquina AM y diseñó varios algoritmos para evaluar este tiempo, es decir, el makespan. Fueron propuestos ocho reglas de despacho: por altura (piezas ordenadas de mayor a

menor altura), FIFO, First Fit (similar al FIFO, con la salvedad que busca hasta encontrar una pieza que quepa en la máquina), por volumen (ordenadas de mayor a menor volumen), ponderado (ponderar en un 10% el volumen de las piezas y 90% en la altura), por cociente (ordenar de menor a mayor cociente superficie/altura), por tiempo de fabricación y por superficie. Para realizar la comparación, lo hizo en términos de diferencia porcentual relativa (Relative Percentaje Deviation, RPD), encontrando que el mejor fue por altura. Adicionalmente a la proposición de las reglas constructivos anteriores, propuso un algoritmo de búsqueda local a partir de la regla por superficie. En este trabajo se utilizó una base de datos que contiene información asociada a 1000 piezas, 100 máquinas y 42 instancias provenientes de Li et al. (2017).

Senovilla Minguela (2019) presentó un método para incrementar la productividad de las empresas productoras mediante tecnologías AM. El método constó de dos etapas. En la primera propuso un algoritmo que permitió optimizar la distribución de las piezas en la superficie de fabricación, programado con Python. En la segunda eligió la mejor solución desde el punto de vista del retorno y de la productividad en cada orden de fabricación para permitir a las empresas organizar de forma más eficiente la producción.

En cuanto a trabajos relacionados con programación de operaciones, se encuentra el realizado por Lozano & Torres (2018) quienes abordaron el problema del taller de trabajo flexible con tiempos de configuración dependientes de la secuencia (SDST-FJSP), que se considera un problema NP-hard, con el objetivo de minimizar el makespan. Para resolver este problema, diseñaron un algoritmo híbrido genético donde parte de la población inicial se generó utilizando la metaheurística de recocido simulado. Además, realizaron un diseño de experimentos 2^3 para determinar cómo los factores influyen en la variable respuesta makespan, resultando que el factor tamaño de la población tiene un efecto significativo. Finalmente, el algoritmo que propusieron fue

validado con 20 instancias de la literatura y comparado con diferentes metodologías propuestas por varios autores, dando como resultado buenas soluciones en las instancias más pequeñas. Sin embargo, para instancias más grandes, el algoritmo tuvo dificultades para encontrar buenas soluciones en comparación con la mejor solución encontrada.

Con respecto a investigaciones del algoritmo de búsqueda tabú, está la hecha por Merlano Canoles & Castellanos Pico (2021) quienes aplicaron el algoritmo para el problema de enrutamiento de un recolector (SPRP) en un almacén de comercio electrónico con almacenamiento disperso y múltiples depósitos. Este problema es considerado NP-hard, debido a la variedad de productos almacenados ubicados en un número variable de estantes y en posiciones distintas dentro de ellos, provocando así, múltiples rutas posibles por las que puede optar el selector, así como múltiples alternativas de depósito. También realizaron una comparación entre un método exacto y el algoritmo en términos de tiempo computacional y distancia (la variable a optimizar). Para esto, compararon el desempeño en instancias pequeñas, medianas y grandes, encontrando que los tiempos de cómputo son excesivamente grandes en el método exacto, e incluso no pudieron encontrar soluciones factibles para el conjunto de grandes instancias. Otro aspecto relevante fue que realizaron un diseño experimental 2^2 con cinco replicas para determinar la influencia que tienen dos parámetros de entrada del algoritmo en la variable de respuesta distancia. El factor más representativo fue el número de iteraciones, mientras que el número tabú resultó no tener significancia en dicha variable.

4.2. Marco teórico

A continuación, se describen las teorías, técnicas y herramientas en las que se soporta la presente investigación. Esta información está dividida en tres ejes temáticos: manufactura aditiva,

problemas de optimización combinatoria y finalmente se detallan los conceptos más relevantes de la metaheurística de búsqueda tabú.

4.2.1. Manufactura aditiva

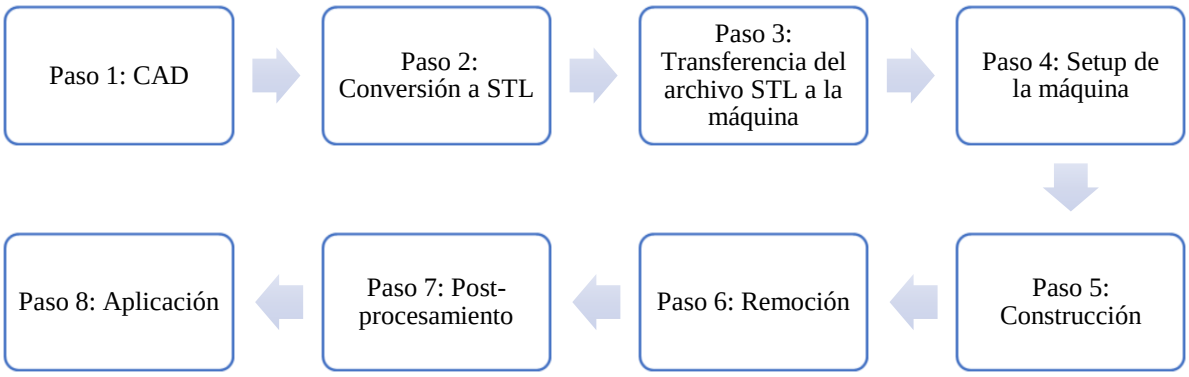
Esta tecnología fue desarrollada por Charles Hull en 1986 en un proceso conocido como estereolitografía (SLA), al que siguieron desarrollos posteriores como la fusión de lecho de polvo (PBF), el modelado por deposición fundida (FDM), entre otros (Ngo et al., 2018). AM implica una serie de pasos que van desde la descripción CAD virtual hasta la pieza física resultante. Pero, para resumir, la mayoría de los procesos de AM involucran, al menos hasta cierto punto, los ocho pasos que se muestran en la Figura 6. A continuación, se explican un poco más estos pasos (Gibson et al., 2015):

- Paso 1: Implica el uso de casi cualquier software profesional de modelado de sólidos CAD, el resultado debe ser una representación sólida o de superficie en 3D.
- Paso 2: Generar el formato de archivo STL, este archivo describe las superficies cerradas externas del modelo CAD original y forma la base para el cálculo de los cortes.
- Paso 3: El archivo STL que describe la pieza debe transferirse a la máquina AM.
- Paso 4: La máquina AM debe configurarse correctamente antes del proceso de construcción, tales configuraciones se relacionan con los parámetros de construcción.
- Paso 5: Construir la pieza, este es un proceso automatizado y la máquina puede continuar en gran medida sin supervisión.
- Paso 6: Una vez que la máquina AM haya completado la construcción, se deben quitar las piezas.

- Paso 7: Es posible que las piezas requieran cierta limpieza adicional antes de que estén listas para usar.
- Paso 8: Ahora las piezas pueden estar listas para usarse. Sin embargo, también pueden requerir un tratamiento adicional antes de que sean aceptables para su uso.

Figura 6.

Proceso genérico AM.



4.2.1.1 Tecnologías AM

Adicionalmente, existen diferentes tecnologías AM y cada una varía en la forma en que forman piezas (de plástico o metal) y pueden diferir en la selección de materiales, el acabado de la superficie, la durabilidad y la velocidad y el costo de fabricación. Las tecnologías AM generalmente se clasifican en siete categorías o grupos (Bourell et al., 2017):

1. *Binder jetting*: Los procesos de chorro de aglutinante depositan líquido en forma de gotitas para unir el material en polvo. A menudo, el aglutinante tiene cualidades adhesivas y se inyecta con tinta sobre la superficie de un lecho de polvo.

2. *Directed energy deposition* (DED): La deposición de energía dirigida es un conjunto de procesos que utilizan energía térmica enfocada para fundir y unir materiales alimentados en forma de polvo o alambre. La fuente térmica suele ser un láser o un haz de electrones.
3. *Material extrusion*: La extrusión de material es la tecnología AM más popular según la cantidad de fabricantes y el bajo costo del hardware. La materia prima se fuerza a través de una boquilla que define el tamaño de las capas.
4. *Material jetting*: El chorro de material se logra depositando selectivamente gotas de material en una plataforma de construcción. En general, las tecnologías comerciales actuales de inyección de material utilizan polímeros termo-endurecibles fotosensibles que se curan tras la deposición.
5. *Powder bed fusion* (PBF): Los procesos de fusión de lecho de polvo consisten en capas delgadas de polvos muy finos, que se esparcen y empaquetan en una plataforma. Los polvos de cada capa se fusionan con un rayo láser o un aglutinante. Las capas posteriores de polvos se colocan sobre las capas anteriores y se unen hasta que se construye la pieza final en 3D (Ngo et al., 2018).
6. *Sheet lamination*: La laminación incluye procesos en los que la materia prima tiene la forma de una lámina, que constituye una capa en la construcción. La hoja se corta a la forma y luego es apilada/adherida a las capas anteriores, o la hoja se une a la capa anterior y luego se corta para formar la geometría de la capa, así como también se sombrea en el área que no es parte para facilitar la eliminación de la parte en el final de la construcción.
7. *Vat polymerization*: La polimerización en cuba, el más antiguo de los procesos AM comerciales, implica la foto polimerización (mediante rayos UV) de resinas termoestables líquidas para formar un sólido.

4.2.1.2 Materiales

En cuanto a materiales, existe una amplia gama, desde chocolate hasta materiales multifuncionales avanzados. Materiales en forma de filamentos, alambre, polvo, pasta, láminas y tintas se pueden utilizar para la impresión 3D. Los principales materiales usados en AM son los siguientes (Ngo et al., 2018):

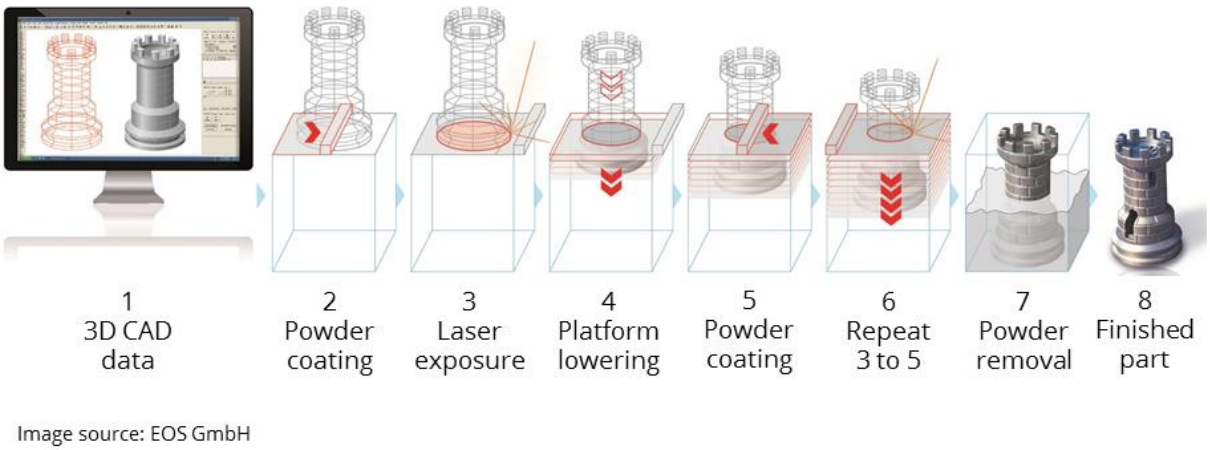
- Polímeros y materiales compuestos: Los polímeros se consideran los materiales más comunes que se han desarrollado para las industrias aeroespacial, automotriz, deportiva, médica, arquitectónica y de juguetes. Los polímeros utilizados en AM se encuentran principalmente en forma de filamentos para FDM (el método más común), polvos, aglutinantes auxiliares en el método de lecho de polvo o resinas en estereolitografía.
- Metales y aleaciones: Muchos materiales metálicos, como aceros inoxidables y para herramientas, algunas aleaciones de aluminio, titanio y sus aleaciones, y aleaciones a base de níquel se pueden fabricar mediante procesos AM basados en PBF.
- Cerámicos: AM se ha convertido en un método esencial para la fabricación de cerámicas avanzadas para biomateriales e ingeniería de tejidos. La impresión 3D de formas complejas seguida de sinterización para producir cerámicas con formas complejas se ha desarrollado cada vez más. Además, la impresión 3D de cerámicas porosas o celosías introdujo numerosos beneficios al desarrollar materiales livianos avanzados que se adaptan a diferentes aplicaciones.
- Concreto: La tecnología AM se ha expandido a la industria de la construcción para la fabricación aditiva de estructuras de edificios. Los compuestos de hormigón reforzado con fibras impresos en 3D ofrecen la ventaja de controlar la orientación de las fibras en comparación con el hormigón reforzado tradicional.

4.2.1.3 Manufactura aditiva de metales

La presente investigación se centrará principalmente en manufactura aditiva de metales (MAM), de modo que se profundizará más en los procesos para este material. Los dos principales procesos MAM, la fusión selectiva por láser (SLM) y la fusión por haz de electrones (EBM) que se basan en lecho de polvo, donde el polvo metálico se deposita en la plataforma de construcción y se funde con un láser de alta energía o un haz de electrones capa por capa para formar piezas sólidas de metal (Li et al., 2020). Este proceso se ilustra en la Figura 7.

Figura 7.

Proceso MAM.



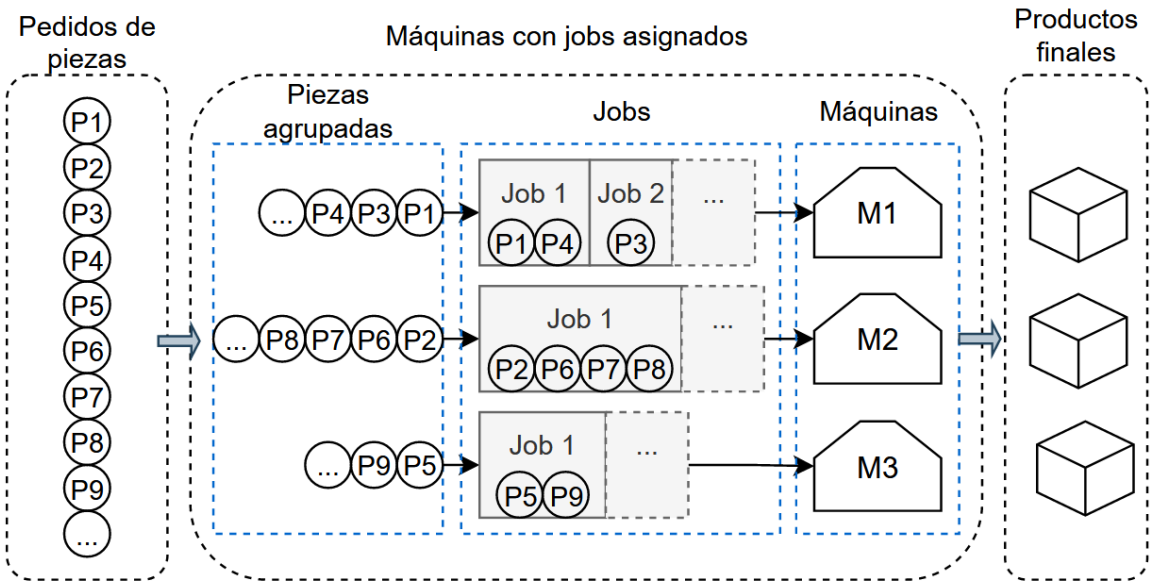
Tomado de: 3D-Printing Using Micro Laser Sintering (3D Micromac) por EOS. https://3d-micromac.com/laser-micromachining/applications/3d-printing/

Una máquina MAM basada en lecho de polvo es un tipo de máquina de procesamiento por lotes en la que un lote de piezas idénticas o no idénticas se puede procesar simultáneamente según su capacidad. En MAM, la producción de un lote de piezas se denomina *job* de producción. Las piezas pueden formar un *job* solo cuando pueden adaptarse a la capacidad de la máquina MAM,

que generalmente está limitada por el espacio paralelepípedo de la cámara de construcción de la máquina. Para cada *job* se requiere un tiempo específico de preparación/limpieza, mientras que el tiempo de procesamiento y el costo de cada *job* varían según el volumen total del material y la altura máxima de las piezas incluidas en este *job*, así como la eficiencia de la máquina MAM para realizar este *job* (Li et al., 2020). En la Figura 8 se ilustra el problema de programación de pedidos en AM.

Figura 8.

Representación del problema de programación de múltiples máquinas AM.



Adaptado de: Metal Additive Manufacturing: Nesting vs. Scheduling (p. 172) por Kucukkoc, 2021.

Este problema es diferente del problema de programación por lotes en varios aspectos. Una de las características más peculiares del problema de programación de máquinas AM es que el tiempo de producción de un *job* se calcula a través de una función basada en algunas características de las piezas producidas en ese *job* (mientras que, en un problema clásico de programación por

lotes, el tiempo de producción del trabajo se conoce en ventaja). Eso hace que sea imposible calcular el tiempo de procesamiento de cualquier parte individual antes de la programación. Además, las diferentes combinaciones de piezas en los *jobs* conducen a diferentes medidas de desempeño y costos en el problema de programación de máquinas AM (Kucukkoc, 2021).

4.2.2. Problemas de optimización

En matemáticas e informática, un problema de optimización es el problema de encontrar la mejor solución entre todas las soluciones factibles. En los términos más simples, un problema de optimización consiste en maximizar o minimizar una función real eligiendo sistemáticamente valores de entrada dentro de un conjunto permitido y calculando el valor de la función (Sadrehaghighi, 2022). Cabe señalar que los problemas de Optimización en los que las variables de decisión son enteras, es decir, donde el espacio de soluciones está formado por ordenaciones o subconjuntos de números naturales, reciben el nombre de Optimización Combinatoria. En este caso, se trata de hallar el mejor valor de entre un número finito o numerable de soluciones viables (López, 2010).

4.2.2.1 Complejidad computacional

La teoría de la complejidad estudia la manera de clasificar problemas de acuerdo con la dificultad propia para resolverlos, basándose en los recursos necesarios y requeridos para establecer su grado de complejidad (Cruz et al., 2014).

- NP es el conjunto de todos los problemas de decisión que se pueden resolver mediante algoritmos no deterministas en tiempo polinomial. Incluye los P y NP-complete (intersección entre NP y NP-hard).

- Problemas P, para los cuales existe una máquina de Turing determinista que los puede resolver en tiempo polinómico. Esto indica que existe un algoritmo determinista con complejidad polinomial que los puede resolver. Se consideran como la clase de problemas de reconocimiento relativamente sencillos, aquellos para los que existen algoritmos eficientes o exactos (Cruz et al., 2014).
- NP-complete, para los cuales no existe una máquina de Turing determinista que pueda resolverlos en tiempo polinómico. En su lugar, se puede encontrar un valor próximo a la solución del problema mediante una máquina de Turing no determinista acotando polinomialmente el tiempo. Estos problemas NP son aquellos cuya solución no se ha resuelto de manera exacta por medio de algoritmos deterministas en tiempo polinomial (Cruz et al., 2014).
- NP-hard: Cuando se prueba que un problema de optimización combinatoria en su versión problema de decisión, pertenece a la clase NP-complete, entonces la versión optimización es NP-hard (Riojas, 2005).

4.2.2.2 Problemas de programación de la producción

La programación de operaciones tiene como objetivo definir en forma concreta en que los recursos disponibles ejecutarán cada una de las operaciones necesarias para la realización de las órdenes de trabajo emitidas y los instantes en que tendrá lugar dicha ejecución. Los principales ambientes del problema de programación de la producción determinista incluyen (José & Delgado, 2012):

- Maquina única: Es el ambiente más simple y existe únicamente una máquina de producción.

- Maquinas paralelas: Consiste en resolver la programación de trabajos en un sistema de capacidad múltiple con m máquinas que realizan operaciones, dispuestas en paralelo y n trabajos a procesar en una, y sólo una, de las máquinas. Las maquinas pueden ser iguales o diferentes (Salazar-Hornig E & Medina-SJC, 2013).
- Flow shop: Hay m máquinas en serie, cada trabajo se procesa en cada una de ellas. Todos los trabajos tienen la misma trayectoria, es decir, primero se procesan sobre la máquina 1, luego sobre la máquina 2, y así sucesivamente. Cuando un trabajo deja de usar una máquina se agrega a la cola de la próxima máquina (José & Delgado, 2012).
- Job shop: Hay m máquinas y n trabajos. Cada trabajo tiene predeterminada su ruta. Se hace una distinción cuando los trabajos visitan las máquinas solo una vez y cuando los trabajos visitan las máquinas más de una vez (José & Delgado, 2012).
- Open shop: Hay m máquinas y n trabajos. Cada trabajo se debe procesar en cada una de las m máquinas. No hay restricción en relación a la ruta de cada trabajo a través del ambiente de máquinas. El plan determina la ruta de cada trabajo, diferentes trabajos pueden tener distintas rutas (José & Delgado, 2012).

Para estos problemas es usual realizar un diagrama de Gantt, el cual es un gráfico de barras apiladas para mostrar actividades (tareas o eventos) contra el tiempo. A la izquierda del gráfico hay una lista de actividades y en la parte superior hay una escala de tiempo adecuada. Cada actividad está representada por una barra; la posición y la longitud de la barra reflejan la fecha de inicio, la duración y la fecha de finalización de la actividad. El momento es que se finaliza la última actividad o pedido corresponde al makespan, es decir, es el tiempo que transcurre desde el inicio de la producción de los pedidos hasta su finalización.

4.2.2.3 Métodos de solución

Los métodos de solución que existen en la investigación de operaciones se pueden clasificar como:

- Exactos: Los métodos exactos evalúan todas las posibles soluciones del problema y seleccionan la que tenga un mejor valor de la variable a optimizar. Garantizan una solución óptima utilizando programación matemática y algoritmos exactos para la exploración de grafos.
- Heurísticos: Algunos problemas (y los correspondientes modelos de IO) pueden ser tan complicados que no es posible resolverlos para encontrar una solución óptima. En tales situaciones, aún es importante encontrar una buena solución factible que al menos esté razonablemente cerca de ser óptima. Por lo general, para buscar esa solución se utilizan métodos heurísticos (Hillier & Lieberman, 2010). Se califica de heurístico a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación. Se usa el calificativo heurístico en contraposición a exacto (Riojas, 2005).
- Metaheurísticos: Una metaheurística es un método de solución general que proporciona tanto una estructura general como criterios estratégicos para desarrollar un método heurístico específico que se ajuste a un tipo particular de problema. La metaheurística se ha convertido en una de las técnicas más importantes del paquete de herramientas que utilizan los profesionales de la IO (Hillier & Lieberman, 2010). Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para

> resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las metaheurísticas se sitúan conceptualmente “por encima” de las heurísticas en el sentido que guían el diseño de éstas, pueden estar compuestas por una combinación de algunas heurísticas (Riojas, 2005).

4.2.3. Algoritmo de búsqueda tabú

La búsqueda tabú es una metaheurística propuesta por primera vez por Fred Glover en un artículo publicado en 1986, y utiliza el concepto de "memoria" para forzar la búsqueda en nuevas áreas. El algoritmo es una variación del método *hill climbing* que incluye una lista tabú de longitud L, que almacena las soluciones más recientes que se convierten en "tabú" y, por lo tanto, no se pueden usar al seleccionar una nueva solución. La intención es mantener una memoria a corto plazo de los cambios recientes, evitando que movimientos futuros eliminen estos cambios. De manera similar al método *hill climbing*, la búsqueda de soluciones dentro de la vecindad de la solución actual (cambio de función) puede ser determinista, incluida toda la vecindad, o estocástica (por ejemplo, una pequeña perturbación aleatoria) (Cortez, 2014).

4.2.3.1 Intensificación y diversificación

Las estrategias de intensificación y diversificación constituyen dos elementos altamente importantes en un proceso de búsqueda tabú. Las estrategias de intensificación se basan en la modificación de reglas de selección para favorecer la elección de buenas combinaciones de movimientos y características de soluciones encontradas. Esto implica que es necesario identificar un conjunto de soluciones elite cuyos buenos atributos puedan ser incorporados a nuevas soluciones creadas. La pertenencia al conjunto de soluciones elite se determina generalmente atendiendo a los valores de la función objetivo comparados con la mejor solución obtenida hasta

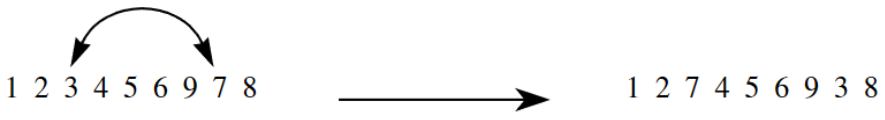
el momento. Por otro lado, las estrategias de diversificación tratan de conducir la búsqueda a zonas del espacio de soluciones no visitadas anteriormente y generar nuevas soluciones que difieran significativamente de las ya evaluadas (Glover & Melián, 2003).

4.2.3.2 Vecindario

La búsqueda tabú se centra principalmente en una exploración no trivial de todas las soluciones en un vecindario. El origen del nombre de este método se remonta a la idea de que una búsqueda local puede continuar más allá de un óptimo local, pero para asegurarse de que no volverá periódicamente al mismo óptimo local, es necesario prohibir volver a las soluciones ya visitadas, en otras palabras, algunas soluciones deben tener un estado tabú. En esencia, la búsqueda tabú no se centra en el azar, aunque se pueden introducir componentes aleatorios principalmente por razones técnicas. La idea básica de la búsqueda tabú es hacer uso de memorias durante la exploración de una parte de las soluciones del problema, que consiste en pasar repetidamente de una solución a una solución vecina. En la Figura 9 se ilustra una posibilidad de crear una solución vecina, para un problema donde se busca una permutación. Una modificación hecha a una solución se llama movimiento (Dreo et al., 2006).

Figura 9.

Ejemplo de creación de solución vecina.



Tomado de: Metaheuristics for Hard Optimization (p. 51) por Dreo et al., 2006.

4.2.3.3 Lista de candidatos

Para reducir el número de soluciones elegibles, algunos autores adoptan la política de elegir aleatoriamente un pequeño número de soluciones. Cuando una vecindad está dada por un conjunto estático M de movimientos, también se puede considerar dividir M en subconjuntos; en cada iteración, solo se examinará uno de estos subconjuntos. De esta manera, se puede hacer un examen parcial pero cíclico de la vecindad, lo que permitirá elegir un movimiento más rápido, con un deterioro asociado en la calidad ya que no se toman en cuenta todos los movimientos en cada iteración. Sin embargo, a nivel global, esta limitación podría no influir demasiado en la calidad de las soluciones producidas, ya que un examen parcial puede generar cierta diversidad en las soluciones visitadas, precisamente porque los movimientos que se eligieron no fueron las que habrían sido, si se llevara a cabo un examen completo del vecindario.

Finalmente, de acuerdo con la intuición de F. Glover cuando propuso el concepto de lista de candidatos, se puede suponer que un movimiento de buena calidad para una solución seguirá siendo bueno para soluciones no muy diferentes. En la práctica, esto se puede implementar ordenando, en una iteración dada, el conjunto completo de todos los movimientos factibles por calidad decreciente. Durante las iteraciones posteriores, solo se considerarán aquellos movimientos que se clasificaron entre los mejores. Esto se implementa en forma de una estructura de datos llamada lista de candidatos. Naturalmente, el orden de los movimientos se degradará durante la búsqueda, ya que las soluciones se vuelven cada vez más diferentes de la solución utilizada para construir la lista y es necesario evaluar periódicamente todo el vecindario para preservar una lista de candidatos adecuada (Dreo et al., 2006).

4.2.3.4 Lista tabú

La lista tabú es una lista en el contexto computacional, donde se registran aquellas soluciones o atributos de soluciones que no deben ser elegidas. Una forma sencilla de construir una lista tabú consiste en que cada vez que se realiza un movimiento, se introduce su inverso en una lista circular, de forma que los elementos de dicha lista están penalizados durante un cierto tiempo. Por tanto, si un movimiento está en la lista tabú no será aceptado, aunque aparentemente sea mejor solución que la solución actual. La lista tabú puede contener (Riojas, 2005):

- Soluciones visitadas recientemente
- Movimientos realizados recientemente
- Atributos o características que tenían las soluciones visitadas

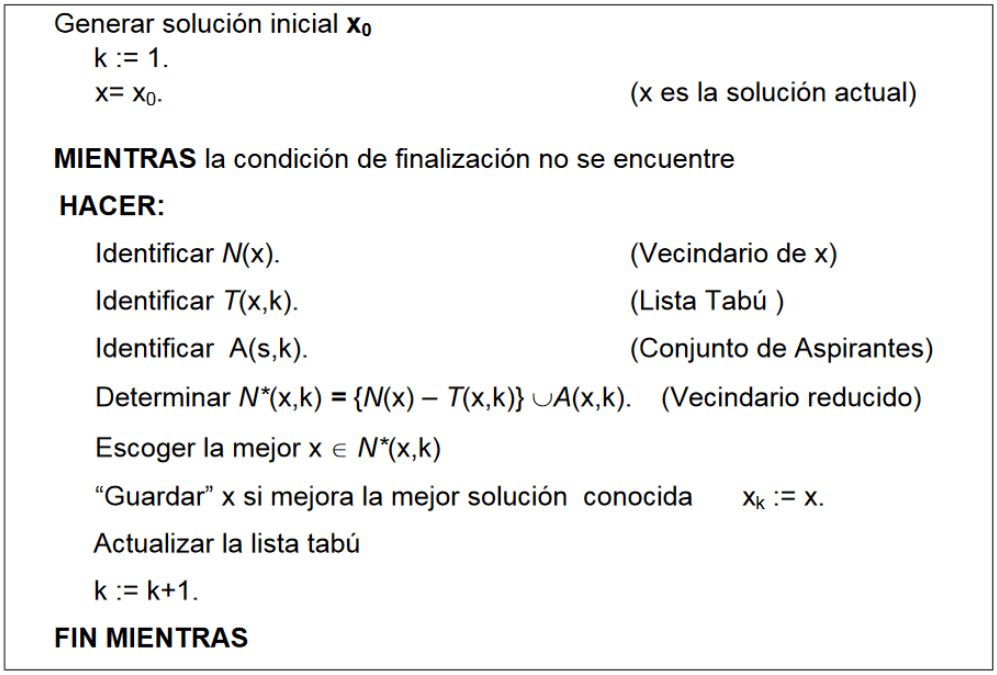
Para un número muy pequeño de movimientos tabú, la búsqueda iterativa tenderá a visitar las mismas soluciones una y otra vez. Si se aumenta este número, disminuye la probabilidad de quedar confinado a un número muy limitado de soluciones y, en consecuencia, aumenta la probabilidad de visitar varias soluciones buenas. Sin embargo, el número de movimientos tabú no debería ser muy grande, porque entonces se vuelve menos probable encontrar buenos óptimos locales, por falta de movimientos disponibles. Hasta cierto punto, la búsqueda está dirigida por los pocos movimientos permitidos más que por la función objetivo (Dreo et al., 2006).

4.2.3.5 Procedimiento básico del algoritmo

La búsqueda tabú procede como cualquier algoritmo de búsqueda: Dada una solución x se define un entorno o vecindario N(x), se evalúa y se “mueve” a una mejor solución, pero en lugar de considerar todo el entorno o vecindario la búsqueda tabú define el entorno reducido N*(x) como aquellas soluciones disponibles (no tabú) del entorno de x (Riojas, 2005). En la Figura 10 se presenta el Seudocódigo básico del algoritmo TS.

Figura 10.

Seudocódigo básico del algoritmo TS.



Tomado de: Conceptos, algoritmo y aplicación al problema de las N-reinas. Capítulo3. Búsqueda de tabú (p. 18) por (Riojas, 2005)

5. Metodología

5.1. Modelo matemático

Para la definición del modelo matemático, en primera instancia, se examinaron los modelos matemáticos expuestos en las publicaciones de la revisión de literatura hecha. Después, se escogió un modelo de los revisados como base, describiéndolo en términos de función objetivo, variables, parámetros y restricciones que se tendrán en cuenta.

El modelo matemático base establecido para la presente investigación es el presentado por Kucukkoc (2019) para un entorno de varias máquinas diferentes. El problema consiste en la programación de operaciones a realizar por un conjunto de máquinas AM ($m \in M = \{1, \ldots, m_n\}$). La máquina AM se caracteriza por un conjunto de especificaciones, las cuales son: el tiempo de

impresión por unidad de volumen de material (VT_m), el tiempo para la aplicación de capas de polvo (HT_m), y el tiempo necesario para el alistamiento (SET_m), como la inicialización y limpieza, que se repite para cada *job*. Además, se conocen las características físicas de cada máquina AM, es decir, el área de producción de la bandeja de la máquina (MA_m), y la altura máxima que soporta la máquina (MH_m). El objetivo del problema de programación de máquinas AM es procesar por lotes de manera óptima una colección de piezas ($i \in I = \{1, \ldots, i_n\}$) en *jobs* (lotes) ($j \in J = \{1, \ldots, j_n\}$) para programar en máquinas AM. Claramente, se cumple que $j_n \leqslant i_n$, porque podemos tener al menos una sola pieza en cada *job*. En cuanto a las dimensiones de las piezas, como se hace en Kucukkoc (2019), no se tiene en cuenta su forma real, y siempre se hace referencia al cuadro delimitador de la pieza. Cada pieza i solo se caracteriza por la altura (h_i), el área (a_i) y el volumen (v_i). Aunque el volumen sí representa el real de la pieza. De hecho, dada una instancia arbitraria, puede suceder que algunas piezas sean incompatibles con algunas máquinas. Una pieza con una altura superior a la altura máxima soportada por una determinada máquina no se puede asignar en ningún *job* de esa máquina. De manera similar, una pieza con un área mayor que el área máxima admitida por una máquina no se puede asignar en ningún *job* en esa máquina (Alicastro et al., 2021).

Notación:

Índices:

i	Índice utilizado para las órdenes o pedidos de las piezas ($i \in I = \{1, 2, \ldots, i_n\}$)
m	Índice utilizado para las máquinas AM ($m \in M = \{1, 2, \ldots, m_n\}$)
j	Índice utilizado para los *jobs* de cada máquina m ($j \in J = \{1, 2, \ldots, j_n\}$)

Parámetros:

h_i	Altura de la pieza de la orden i

a_i Área de la pieza de la orden i

v_i Volumen de material de la pieza de la orden i

VT_m Tiempo de impresión por unidad de volumen de material de la máquina m

HT_m Tiempo para la aplicación de capas de polvo de la máquina m

SET_m Tiempo de alistamiento de la máquina m

MA_m Área de impresión de la máquina m

MH_m Altura de impresión de la máquina m

Variables de decisión:

X_{mji} 1, si la pieza i es asignada al *job* j de la máquina m; 0, si no.

Z_{mj} 1, si el *job* j de la máquina m tiene alguna pieza asignada; 0, si no.

Variables auxiliares:

C_{mj} Tiempo de finalización del *job* j de la máquina m

PT_{mj} Tiempo de procesamiento o impresión del *job* j de la máquina m

Función objetivo:

$$Minimizar \quad \max_{m\in M, j\in J} C_{mj} \tag{1}$$

Restricciones:

$$\sum_{m\in M}\sum_{j\in J} X_{mji} = 1 \quad \forall i \in I \tag{2}$$

$$\sum_{i\in I} a_i\, X_{mji} \leq MA_m \quad \forall m \in M, \forall j \in J \tag{3}$$

$$h_i\, X_{mji} \leq MH_m \quad \forall m \in M, \forall j \in J, \forall i \in I \tag{4}$$

$$\sum_{i\in I} X_{m(j+1)i} \leq \varphi \sum_{i\in I} X_{mji} \quad \forall m \in M, \forall j \leq j_n - 1 \tag{5}$$

$$C_{m(j-1)} + PT_{mj} \leq C_{mj} \quad \forall m \in M, \forall j \in J \tag{6}$$

$$C_{m0} = 0 \quad \forall m \in M \tag{7}$$

$$PT_{mj} = SET_m \, Z_{mj} + VT_m \sum_{i\in I} v_i X_{mji} + HT_m \max_{i\in I}\{h_i X_{mji}\} \quad \forall m \in M, \forall j \in J \tag{8}$$

$$X_{mji}, Z_{mj} \in \{0,1\} \quad \forall m \in M, \forall j \in J, \forall i \in I \tag{9}$$

El objetivo es la minimización del tiempo máximo de finalización, es decir, del makespan. El modelo está compuesto por: restricciones de ocurrencia de partes (2) que restringen que todas las partes sean procesadas exactamente una vez; restricciones de capacidad de área (3) y altura (4), para garantizar que las piezas sean compatibles con la máquina asignada; restricciones en la utilización del *job* (5) para forzar la asignación de *jobs* con una estrategia incremental, lo que significa que si el *job* j de la máquina m está vacío, entonces las piezas no pueden asignarse al *job* j+1; restricciones de tiempo de finalización (6) y (7) y restricciones de tiempo de procesamiento (8) que especifican cómo calcular los tiempos de finalización y procesamiento de los *jobs*. Finalmente, hay restricciones de signo (9).

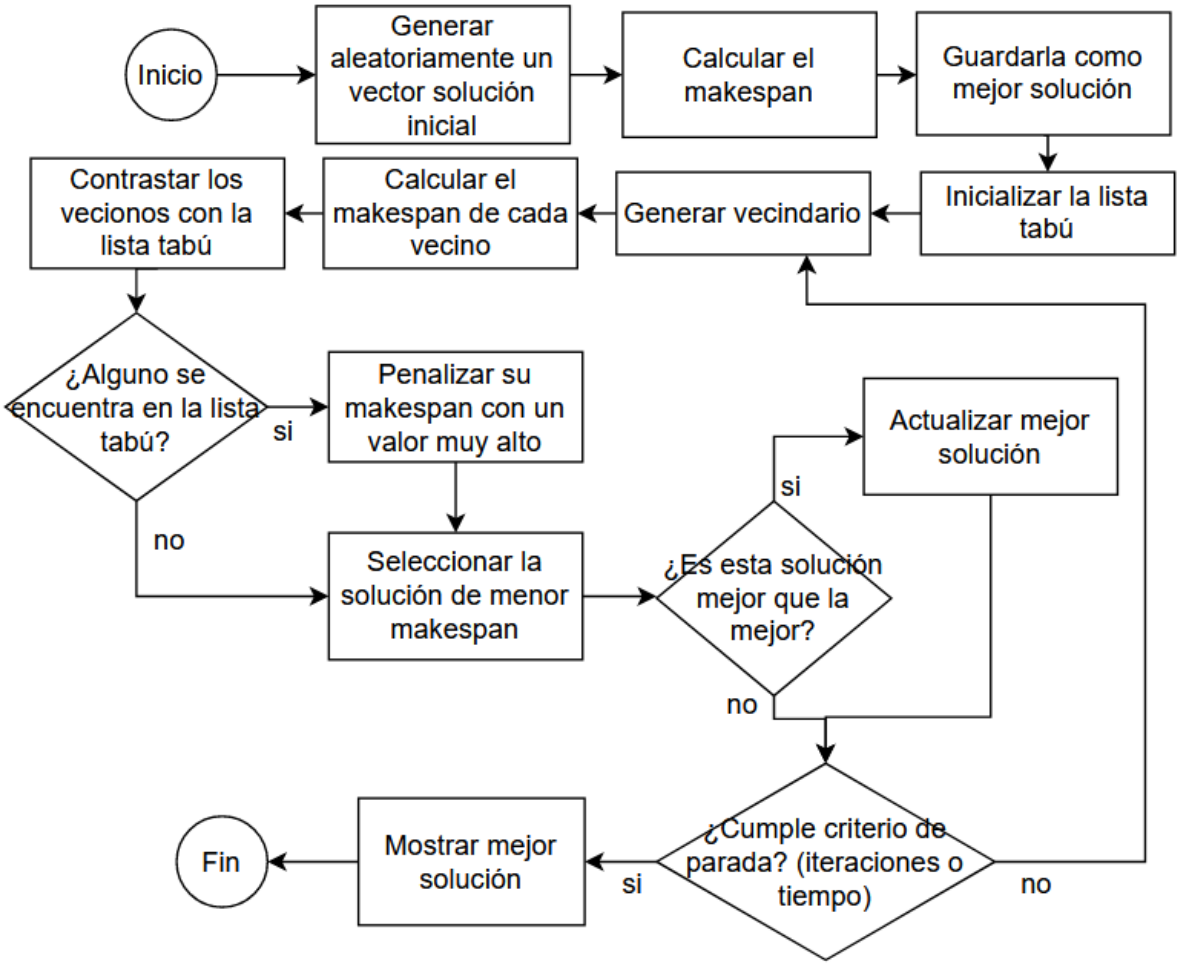
5.2. Algoritmo de búsqueda tabú

El algoritmo realizado cumple con todas las restricciones del modelo matemático expuesto anteriormente. En primera instancia, el algoritmo realiza la lectura de los datos de las piezas y máquinas del problema y genera aleatoriamente un vector solución inicial evaluando el makespan.

Posteriormente genera en cada iteración un vecindario y calcula para cada vecino la función objetivo. Si alguno de los vecinos generados está en la lista tabú, lo penaliza dando un valor muy grande a su respectivo makespan. Finalmente, escoge como mejor vecino al que posea menor makespan y continua con la siguiente iteración. El criterio de parada usado fue el número de iteraciones. En la Figura 11 se muestra el diagrama de flujo del algoritmo TS hecho.

Figura 11.

Diagrama de flujo del algoritmo.



El código del algoritmo TS fue hecho en Matlab R2022b, y se ejecutó en un computador con procesador Intel® Core™ i7-6500U CPU @ 2.50GHz 2.60 GHz y con 16 GB de RAM. Este código se encuentra en el Apéndice A. Adicionalmente, se impuso una restricción de tiempo de ejecución del algoritmo de 60 segundos.

5.2.1. Codificación

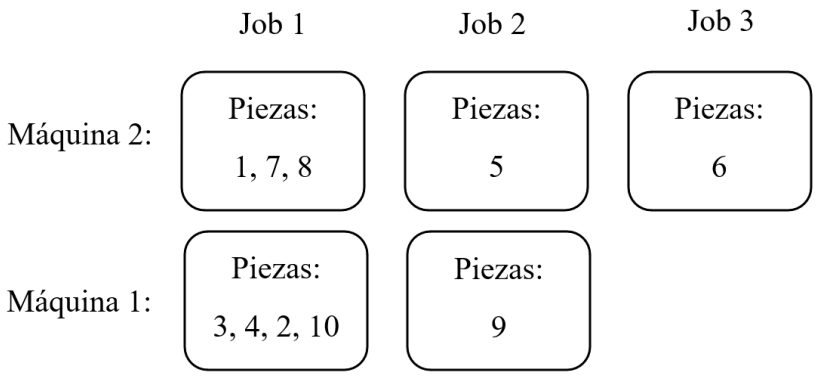
La solución fue codificada en forma de un vector (llamado sol) con números enteros del 1 a la cantidad de piezas ,i_n, de la instancia. Para una completa representación de la solución, se añadió, al vector de piezas, un vector de máquinas (llamado maq) y uno de *jobs* (llamado job). Cada uno también con una longitud igual a la cantidad de piezas. En la Figura 12a se muestran los 3 vectores para un ejemplo de 10 piezas y 2 máquinas, mientras que en la Figura 12b se ilustra gráficamente la solución.

Figura 12.

Ejemplo de la representación de la solución.

sol:	1	5	3	4	7	2	10	8	6	9
maq:	2	2	1	1	2	1	1	2	2	1
job:	1	2	1	1	1	1	1	1	3	2

(a)



(b)

5.2.2. Decodificación

El proceso de decodificación fue el mismo utilizado por Kucukkoc et al. (2018) en su algoritmo genético. En este proceso las piezas se van asignando en el orden que están en el vector solución a las máquinas. Se inicia con el primer *job* de la primera máquina y se van asignando las piezas que cumplan con la restricción de capacidad de la máquina en cuanto a área y altura (se asignan tantas piezas como sea posible). Se continua con el primer *job* de la segunda maquina y cuando todas las maquinas tengan su primer *job* completo, se pasa a los segundos *jobs*. Si alguna pieza no se puede asignar a un *job* de cierta máquina, se omite y se considera para el próximo *job* de la siguiente máquina. Este procedimiento continúa hasta que todas las piezas hayan sido asignadas a un *job* de alguna máquina.

Para realizar esta asignación, fue necesario introducir en el algoritmo un parámetro llamado número de *jobs*. Es necesario ajustar manualmente la cantidad de *jobs* que son suficientes para que todas las piezas queden asignadas. Esto no influye en la respuesta del makespan obtenida (si el número es lo suficientemente alto) pero sí en el tiempo de ejecución del algoritmo.

5.2.3. Cálculo del makespan

Una vez que todas las piezas estén asignadas se puede evaluar la función objetivo, en este caso el makespan. A continuación, se explicará con el ejemplo de la Figura 12 el cálculo del makespan. En la Tabla 3 se muestran los datos de las máquinas que corresponden al problema P1 de las instancias de Li et al. (2017). Primero se debe calcular el tiempo de procesamiento de cada *job*, como lo indica la ecuación (8), este tiempo incluye el tiempo de alistamiento y está dado por la suma de los volúmenes y la altura máxima de las piezas pertenecientes al *job*, así como de las especificaciones de la máquina SET, VT y HT. Los datos de sumatoria de volúmenes y altura máxima de cada *job* están en la Tabla 4.

Como ejemplo, el tiempo de procesamiento del *job* 1 de la máquina 1 se obtiene de la siguiente forma:

$$PT_{11} = SET_1 + VT_1 \sum_{i \in I_{11}} v_i + HT_1 \max_{i \in I_{11}} \{h_i\}$$

$$PT_{11} = 2 + 0{,}0308642 * 4345{,}56 + 0{,}7 * 18{,}09$$

$$PT_{11} = 148{,}79\ h$$

Tabla 3.

Especificaciones de las máquinas del ejemplo.

Máquina	VT [h/cm^3]	HT [h/cm]	SET [h]
1	0,0308642	0,7	2
2	0,0308642	0,7	1

Tabla 4.

Datos de las piezas del ejemplo.

	Máquina 1		Máquina 2	
	$\sum v_i$	Max h_i	$\sum v_i$	Max h_i
Job 1	4345,56	18,09	20583,41	33,23
Job 2	1786,36	12,53	3527,93	16,02
Job 3	-	-	3907,79	11,77

En la Tabla 5 se muestran los tiempos de procesamiento de todos los *jobs*. Dado que se supone que todas las máquinas empiezan a trabajar en el tiempo cero, y como el makespan es el tiempo que transcurre desde el inicio de la producción de los pedidos hasta su finalización, para obtener el makespan se suman todos los tiempos de cada máquina y se selecciona el mayor. De esta forma el makespan calculado es 910,5 horas. El diagrama de Gantt se ilustra en la Figura 13.

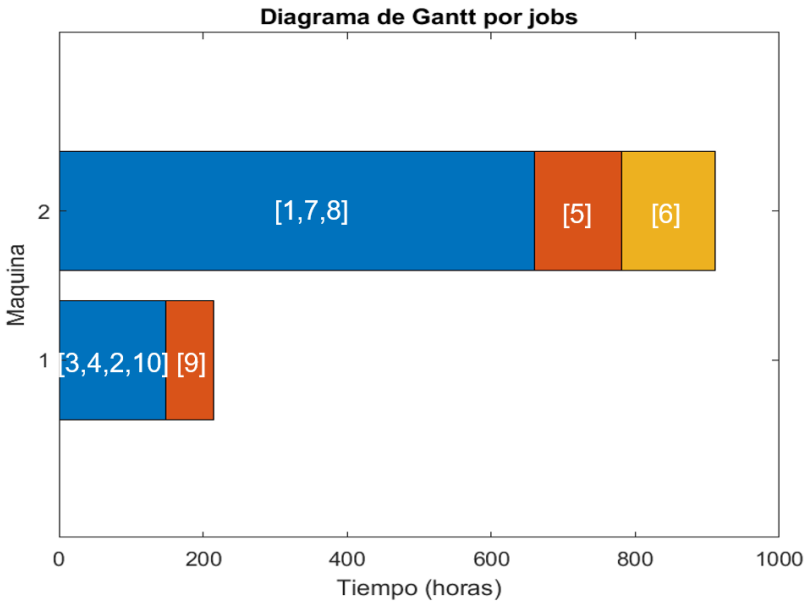
Tabla 5.

Tiempos de procesamiento del ejemplo.

	Tiempo de procesamiento	
	Máquina 1	Máquina 2
Job 1	148,79	659,55
Job 2	65,91	121,10
Job 3	-	129,85
Total	214,69	910,50

Figura 13.

Diagrama de Gantt ejemplo.



5.2.4. Generación del vecindario

Una vez que se tiene la solución inicial el algoritmo genera, a partir de esta solución, un vecindario de i_n-1 vecinos. Para esto se selecciona aleatoriamente una posición del vector y se intercambia con todas las demás posiciones como se ilustra en la Figura 14. Después de la primera

iteración, el algoritmo utiliza como partida para generar los vecinos a la mejor solución no tabú de la iteración anterior.

Figura 14.

Procedimiento para la generación del vecindario.

Solución inicial:	1	10	3	4	7	9	5	8	6	2
Vecino 1:	8	10	3	4	7	9	5	1	6	2
Vecino 2:	1	8	3	4	7	9	5	10	6	2
Vecino 3:	1	10	8	4	7	9	5	3	6	2
Vecino 4:	1	10	3	8	7	9	5	4	6	2
Vecino 5:	1	10	3	4	8	9	5	7	6	2
Vecino 6:	1	10	3	4	7	8	5	9	6	2
Vecino 7:	1	10	3	4	7	9	8	5	6	2
	0	0	0	0	0	0	0	0	0	0
Vecino 8:	1	10	3	4	7	9	5	6	8	2
Vecino 9:	1	10	3	4	7	9	5	2	6	8

Una vez generado el vecindario, se evalúa el makespan de cada vecino. Para esto, se lleva a cabo el mismo proceso de decodificación explicado anteriormente para cada vecino, así se ajustan los vectores de máquinas y *jobs* para poder obtener el nuevo valor de makespan. Finalmente, se selecciona el de menor makespan que no se encuentre en la lista tabú. Y si es mejor que la mejor solución guardada hasta el momento, esta se actualiza.

5.3. Validación del algoritmo

Para validar el algoritmo se utilizaron dos paquetes de instancias de la literatura las cuales son de diferentes tamaños, donde se varía el número de piezas y de máquinas. El primero proveniente de Kucukkoc (2019) del cual se seleccionaron las instancias P39 al P62, que

corresponden al problema con varias máquinas AM diferentes. Este paquete posee instancias que van desde 15 piezas con 2 máquinas hasta 46 piezas con 3 máquinas. El segundo paquete, de Li et al. (2017), consta de 42 problemas que van desde 10 piezas con 2 máquinas hasta 660 piezas con 6 máquinas. Las instancias se encuentran en el Apéndice B y proporcionan datos sobre las piezas y máquinas. Para las piezas, estos datos constan de: primero, la identificación numérica de la pieza, que consiste en un número entero positivo. En segundo lugar, la altura de la pieza, en cm. En tercer lugar, el volumen de la pieza, en cm^3. Por último, la superficie que ocupa la pieza en la máquina, en cm^2. Para las máquinas, los datos son, en primer lugar, número de identificación. En segundo lugar, dependiendo de la máquina, se tienen los valores del tiempo de preparación, SET, el tiempo de procesamiento por unidad de volumen, VT y el tiempo por unidad de altura, HT. Por último, la superficie disponible de la máquina y la altura máxima a la que puede imprimir.

Para calibrar la cantidad de iteraciones y la longitud de la lista tabú del algoritmo, se realizaron unas pruebas iniciales con cada instancia para poder definir los niveles bajo y alto de cada factor del diseño de experimentos. En estas pruebas se corrió el algoritmo con diferentes valores para el número de iteraciones y la longitud de lista tabú, además se pudo observar la cantidad de *jobs* que se estaban requiriendo en cada instancia. Los datos obtenidos a partir de estas pruebas se pueden visualizar en el Apéndice C.

Los niveles alto y bajo para las iteraciones y la longitud de lista tabú de cada instancia se muestran en el Apéndice D, así como los resultados del diseño de experimentos. Se llevó a cabo un diseño factorial 2^2 con 10 réplicas. Posteriormente, se escogió la combinación que dio mejores resultados de la función objetivo. Estos valores de los parámetros que fueron usados para generar los resultados finales se muestran en la Tabla 6 y la Tabla 7.

Los resultados del algoritmo TS hecho fueron comparados con los de Kucukkoc (2019) quien trabajó CPLEX y también se compararon con los obtenidos por la metaheurística de Búsqueda Local Iterada de Aprendizaje por Refuerzo (ILS+Q-Learning) de Alicastro et al. (2021) y con un algoritmo evolutivo (EA) que este autor adaptó de la literatura.

Las comparaciones fueron hechas en términos de GAP y DEV. Donde GAP es la diferencia relativa entre un valor de función objetivo (OFV) y el logrado por CPLEX (CPLEX OFV), mientras que DEV es la diferencia relativa entre el valor de una función objetivo y el mejor valor encontrado por uno de los algoritmos.

$$GAP = \frac{OFV - CPLEX\ OFV}{CPLEX\ OFV} * 100\% \tag{10}$$

$$DEV = \frac{OFV - mejor\ OFV}{mejor\ OFV} * 100\% \tag{11}$$

Tabla 6.

Parámetros del algoritmo usados en las instancias de Kucukkoc (2019).

Instancia	Iteraciones	Longitud lista tabú	Número de *jobs*
P39	200*	40	5
P40	150	10	5
P41	200*	40	5
P42	150	40	5
P43	150	80	5
P44	150	20*	5
P45	150	10	5
P46	150	20	5
P47	150	20	5
P48	200*	40	5
P49	150	40	5
P50	150	10	5
P51	150	20	5

Instancia	Iteraciones	Longitud lista tabú	Número de *jobs*
P52	150	10	5
P53	150	20	5
P54	150	20	5
P55	150	10	5
P56	100	40	5
P57	200*	10	5
P58	100	20	5
P59	150	40	5
P60	100	10	5
P61	200*	10	5
P62	150	20	5

Nota: los parámetros marcados con un asterisco (*) mostraron tener efectos significativos en el ANOVA.

Tabla 7.

Parámetros del algoritmo usados en las instancias de Li et al. (2017).

Instancia	Iteraciones	Longitud lista tabú	Número de *jobs*
P1	10	10	5
P2	10	10	5
P3	50*	20*	5
P4	50*	10	10
P5	50	10	10
P6	50	10	10
P7	10	10	10
P8	50*	10	10
P9	50*	10	10
P10	50	10	10
P11	100*	10	10
P12	100*	10*	10
P13	50	20*	10
P14	100	10	5
P15	100*	10	10
P16	100*	10	15
P17	150	40	10
P18	50	10	10
P19	100	20	10
P20	100	20	10
P21	50*	10	15

Instancia	Iteraciones	Longitud lista tabú	Número de *jobs*
P22	100	20	10
P23	100	20	10
P24	150	20	10
P25	150	40	10
P26	150	40	10
P27	200	20	15
P28	200*	20	15
P29	200*	10	20
P30	150	80*	25
P31	150	20	5
P32	200*	10	10
P33	150	10	10
P34	150	40	10
P35	200*	10	15
P36	200*	80	15
P37	200*	80	15
P38	200*	20	20
P39	150*	10	30
P40	100	80	50
P41	50*	10	70
P42	20	10	145

Nota: los parámetros marcados con un asterisco (*) mostraron tener efectos significativos en el ANOVA.

6. Resultados

En esta sección se presentan los resultados finales obtenidos por el algoritmo TS en la resolución de los problemas de programación de la producción de múltiples máquinas AM de las instancias provenientes de Kucukkoc (2019) y de Li et al. (2017). Se tomó la mejor respuesta después de 10 corridas. En promedio, se alcanzó un valor del makespan de 368,36 y 7918,09 horas, respectivamente para cada paquete.

Para el primer paquete de instancias, primero se compararon los valores del makespan y tiempo de cómputo contra las soluciones de Kucukkoc (2019) obtenidas mediante CPLEX (Tabla

8). Se obtuvo un GAP promedio de 2,45%. Los tiempos de cómputo promedio fueron de 0,48 segundos para el algoritmo TS, y de 1020,69 para el CPLEX. Es decir, se puede observar una disminución muy significativa de tiempo con resultados que no distan tanto de los arrojados par el método exacto. Este primer paquete también se comparó con los resultados de las metaheurísticas ILS+Q-Learning y EA provenientes de Alicastro et al. (2021) como se muestra en la Tabla 9. Se obtuvo una DEV promedio de -0,7% y -0,75%, respectivamente para cada algoritmo comparado.

En segundo paquete de instancias se comparó únicamente con los resultados de Alicastro et al. (2021) como se muestra en la Tabla 10. Se logró un DEV promedio de 3,02% contra el ILS+Q-Learning y de 3,01% contra el EA. El tiempo promedio de cómputo del algoritmo TS fue de 10,45 segundos. En estos problemas que son de mayor tamaño (principalmente en cuanto al número de piezas) se puede observar un decrecimiento del rendimiento del algoritmo TS en comparación con la ILS+Q-Learning y el EA.

Tabla 8.

Resultados del algoritmo en las instancias de Kucukkoc (2019) y comparación con CPLEX.

Instancia	Número de piezas	Número de máquinas	TS		CPLEX		TS/CPLEX
			Makespan [h]	Tiempo [s]	Makespan [h]	Tiempo [s]	GAP [%]
P39	15	2	196,60	0,48	195,44	5,8	0,60
P40	15	2	199,52	0,18	199,45	5,8	0,04
P41	17	2	388,51	0,59	385,59	8,4	0,76
P42	17	2	399,50	0,48	396,93	5,8	0,65
P43	18	2	376,05	0,60	372,58	7,4	0,93
P44	18	2	381,17	0,36	380,22	7,6	0,25
P45	21	2	287,95	0,23	286,53	21,4	0,50
P46	21	2	296,05	0,35	293,09	28,2	1,01
P47	22	2	426,01	0,38	425,93	26,2	0,02
P48	22	2	437,18	0,62	435,51	38,9	0,38
P49	23	2	447,97	0,70	447,48	67	0,11
P50	23	2	456,31	0,26	456,31	84	0,00

Instancia	Número de piezas	Número de máquinas	TS		CPLEX		TS/CPLEX
			Makespan [h]	Tiempo [s]	Makespan [h]	Tiempo [s]	GAP [%]
P51	25	3	329,32	0,47	296,05	48,9	11,24
P52	25	3	334,83	0,32	299,71	141,1	11,72
P53	28	3	360,36	0,48	351,67	2400	2,47
P54	28	3	371,62	0,53	357,76	2400	3,87
P55	30	3	355,93	0,47	342,3	2400	3,98
P56	30	3	359,81	0,48	345,04	2400	4,28
P57	36	3	382,83	0,59	374,05	2400	2,35
P58	36	3	392,08	0,45	377,12	2400	3,97
P59	38	3	373,26	0,84	364,62	2400	2,37
P60	38	3	382,94	0,32	368,94	2400	3,79
P61	46	3	450,12	0,72	443,71	2400	1,45
P62	46	3	454,81	0,68	445,38	2400	2,12

Nota: en las instancias en las que el CPLEX tuvo un tiempo de 2400, no son soluciones optimas (se cumplió el tiempo límite).

Tabla 9.

Resultados del algoritmo en las instancias de Kucukkoc (2019) y comparación con metaheurísticas.

Instancia	Número de piezas	Número de máquinas	TS	ILS+Q-Learning	EA	TS/ILS+Q-Learning	TS/EA
			Makespan [h]	Makespan [h]	Makespan [h]	DEV [%]	DEV [%]
P39	15	2	196,60	197,51	197,51	-0,46	-0,46
P40	15	2	199,52	203,89	204,49	-2,14	-2,43
P41	17	2	388,51	389,97	389,97	-0,37	-0,37
P42	17	2	399,50	397,78	397,78	0,43	0,43
P43	18	2	376,05	378,76	378,76	-0,72	-0,72
P44	18	2	381,17	385,09	385,68	-1,02	-1,17
P45	21	2	287,95	280,61	281,2	2,62	2,40
P46	21	2	296,05	294,95	294,95	0,37	0,37
P47	22	2	426,01	414,25	414,25	2,84	2,84
P48	22	2	437,18	433,1	433,1	0,94	0,94
P49	23	2	447,97	434,74	435,83	3,04	2,79
P50	23	2	456,31	454,85	454,85	0,32	0,32
P51	25	3	329,32	436,52	436,52	-24,56	-24,56
P52	25	3	334,83	456,55	456,55	-26,66	-26,66
P53	28	3	360,36	348,83	349,16	3,31	3,21
P54	28	3	371,62	359,01	359,66	3,51	3,33
P55	30	3	355,93	340,68	340,74	4,48	4,46

Instancia	Número de piezas	Número de máquinas	TS	ILS+Q-Learning	EA	TS/ILS+Q-Learning	TS/EA
			Makespan [h]	Makespan [h]	Makespan [h]	DEV [%]	DEV [%]
P56	30	3	359,81	350,61	350,34	2,62	2,70
P57	36	3	382,83	370,81	370,55	3,24	3,31
P58	36	3	392,08	381,19	381,67	2,86	2,73
P59	38	3	373,26	363,18	362,8	2,78	2,88
P60	38	3	382,94	374,07	374	2,37	2,39
P61	46	3	450,12	438,58	438,7	2,63	2,60
P62	46	3	454,81	451,82	451,85	0,66	0,66

Tabla 10.

Resultados del algoritmo en las instancias de Li et al. (2017) y comparación con metaheurísticas.

Instancia	Número de piezas	Número de máquinas	TS		ILS+Q-Learning	EA	TS/ILS+Q-Learning	TS/EA
			Makespan [h]	Tiempo [s]	Makespan [h]	Makespan [h]	DEV [%]	DEV [%]
P1	10	2	910,50	0,02	910,5	910,5	0,00	0,00
P2	10	2	1189,59	0,03	1168,22	1168,22	1,83	1,83
P3	14	2	1520,24	0,12	1516,24	1517,05	0,26	0,21
P4	16	2	3323,26	0,10	3282,19	3282,19	1,25	1,25
P5	18	2	2356,87	0,11	2043,57	2043,57	15,33	15,33
P6	20	2	6920,72	0,12	6918,48	6918,48	0,03	0,03
P7	15	3	1455,80	0,03	1457,57	1457,57	-0,12	-0,12
P8	18	3	1456,08	0,14	1417,39	1417,39	2,73	2,73
P9	21	3	3772,90	0,19	3777,21	3777,21	-0,11	-0,11
P10	24	3	5225,98	0,15	5225,98	5225,98	0,00	0,00
P11	27	3	5993,22	0,30	5993,22	5993,22	0,00	0,00
P12	30	3	6871,87	0,47	6871,53	6871,53	0,00	0,00
P13	20	4	1903,79	0,25	1912,28	1912,28	-0,44	-0,44
P14	24	4	1769,20	0,27	1729,32	1729,39	2,31	2,30
P15	28	4	5120,85	0,55	5120,85	5120,85	0,00	0,00
P16	32	4	6473,88	0,64	6473,88	6473,88	0,00	0,00
P17	36	4	3997,08	1,22	3970,98	3977,3	0,66	0,50
P18	40	4	8155,85	0,32	8198,11	8198,11	-0,52	-0,52
P19	25	5	5367,25	0,48	5358,14	5360,32	0,17	0,13
P20	30	5	1500,06	0,53	1487,82	1488,23	0,82	0,79
P21	35	5	4179,74	0,55	4179,74	4179,74	0,00	0,00
P22	40	5	3010,63	0,65	-	-	-	-
P23	60	5	4543,22	1,26	-	-	-	-
P24	80	5	3470,38	2,32	3267,4	3266,95	6,21	6,23

Instancia	Número de piezas	Número de máquinas	TS		ILS+Q-Learning	EA	TS/ILS+Q-Learning	TS/EA
			Makespan [h]	Tiempo [s]	Makespan [h]	Makespan [h]	DEV [%]	DEV [%]
P25	100	5	6370,53	4,30	5660,99	5654,69	12,53	12,66
P26	120	5	4584,78	4,87	4489,16	4490,58	2,13	2,10
P27	140	5	6738,18	8,82	6236,83	6223,87	8,04	8,26
P28	160	5	7571,79	11,12	7166,71	7149,92	5,65	5,90
P29	180	5	10259,36	13,73	9597,28	9583,41	6,90	7,05
P30	200	5	13816,04	22,62	13739,83	13731,18	0,55	0,62
P31	30	6	1464,58	0,62	1351,09	1378	8,40	6,28
P32	60	6	1874,41	2,22	1739,14	1736,73	7,78	7,93
P33	90	6	3212,76	2,77	2976,48	2973,83	7,94	8,03
P34	120	6	4900,24	6,34	4491,54	4476	9,10	9,48
P35	160	6	5201,95	11,91	5054,1	5050,85	2,93	2,99
P36	200	6	6661,95	23,25	6596,19	6599,1	1,00	0,95
P37	250	6	9599,05	33,43	9243,98	9241,21	3,84	3,87
P38	300	6	11194,30	46,94	-	-	-	-
P39	360	6	17598,89	59,24	16995,16	16933,64	3,55	3,93
P40	420	6	27955,89	58,32	27718,1	27647,83	0,86	1,11
P41	590	6	41179,12	58,40	-	-	-	-
P42	660	6	61886,93	59,31	-	-	-	-

Nota: en los espacios donde hay guiones (-) no se reportaron resultados por parte de Alicastro et al. (2021)

7. Conclusiones

En esta investigación se utilizó un algoritmo de búsqueda tabú (TS) para resolver el problema de programación de pedidos en un ambiente de múltiples máquinas de manufactura aditiva (AM) diferentes, el cual es de tipo NP-hard. El algoritmo fue probado con instancias de la literatura.

En las instancias de menor tamaño (desde 15 piezas con 2 máquinas hasta 46 piezas con 3 máquinas) el algoritmo TS mostró ser una buena opción para la resolución de este problema. Al ser comparado con el método exacto, logró una disminución del tiempo de cómputo promedio que pasó de 1020,69 a 0,48 segundos, mientras que los resultados de la función objetivo, el makespan,

distaron en promedio un 2,45% de los obtenidos mediante CPLEX. Es decir, con el algoritmo TS el valor promedio del makespan obtenido fue de 368,36 h, mientras que el de CPLEX fue 360,06. Además, al compararlo con otras metaheurísticas como Búsqueda Local Iterada de Aprendizaje por Refuerzo (ILS+Q-Learning) y un algoritmo evolutivo (EA), la DEV promedio fue de -0,7% y -0,75%, respectivamente. Esto significa que el algoritmo TS obtuvo mejores resultados de la función objetivo que estas, aunque bastante cercanos.

Por otra parte, en instancias de mayor tamaño (desde 10 piezas con 2 máquinas hasta 660 piezas con 6 máquinas) el algoritmo TS mostró un menor rendimiento en comparación con la ILS+Q-Learning y el EA, ya que la DEV promedio fue de 3,02% y 3,01%, respectivamente. Adicionalmente, el tiempo promedio de cómputo del algoritmo TS fue de 10,45 segundos. Este tiempo aumentó considerablemente debido al incremento en la cantidad de piezas de las instancias, aumentando el número de vecinos que se debían evaluar en cada iteración. En algunas de estas instancias (principalmente las últimas) se llegó al tiempo límite de 60 segundos establecido para el algoritmo TS.

Finalmente, se recalca que esta metaheurística mostró ser capaz de dar buenos resultados en tiempos relativamente cortos para organizar la producción en esta industria, donde cada pieza a fabricar tiene un tiempo de procesamiento diferente. Por consiguiente, con este trabajo se contribuye a mejorar la resolución de un problema relativamente nuevo, en el que no se han explorado tantas heurísticas ni metaheurísticas diferentes, aun cuando la industria AM tiene un gran potencial de crecimiento, pero suele tener tiempos de producción elevados, razón por la cual es de gran importancia saber ordenar los pedidos a fabricar.

8. Recomendaciones

En futuras investigaciones, se podrían probar otras metaheurísticas ya que es un problema relativamente nuevo, o algún algoritmo híbrido. También sería interesante utilizar modelos matemáticos diferentes, con variedad de materiales (tanto de las piezas como los aceptados por las máquinas), tiempos de alistamiento dependientes, o incluso buscar resolver el problema junto con el de anidamiento, ya que en el modelo trabajado en este estudio no se tuvo en cuenta la forma real de la pieza sino únicamente el rectángulo que delimita la forma real. Al incluir esto se puede trabajar con un modelo aún más cercano a la realidad y tener certeza que se pueden ubicar las piezas de un *job* en la impresora.

Ahora bien, si se quisiera complementar el algoritmo TS, se podría utilizar la memoria a largo plazo de este y realizar una lista tabú que contenga movimientos en vez de soluciones. Además, es posible probar otras estrategias para la generación del vecindario, u otros procedimientos de codificación y decodificación de la solución. Y en caso de usar una forma similar para generar el vecindario, sería recomendable usar un vecindario reducido, esto para reducir los tiempos computacionales en instancias con un número de piezas grande.

Referencias bibliográficas

Alicastro, M., Ferone, D., Festa, P., Fugaro, S., & Pastore, T. (2021). A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. *Computers and Operations Research*, *131*. https://doi.org/10.1016/j.cor.2021.105272

Bourell, D., Pierre, J., Leu, M., Levy, G., Rosen, D., Beese, A. M., & Clare, A. (2017). Materials for additive manufacturing. *CIRP Annals - Manufacturing Technology*, *66*(2), 659–681. https://doi.org/10.1016/j.cirp.2017.05.009

Che, Y., Hu, K., Zhang, Z., & Lim, A. (2021). Machine scheduling with orientation selection and two-dimensional packing for additive manufacturing. *Computers and Operations Research*, *130*. https://doi.org/10.1016/j.cor.2021.105245

Cortez, P. (2014). *Modern Optimization with R*. Springer. http://www.springer.com/series/6991

Cruz, M., Moreno, P., & Peralta, J. (2014). *Aplicación de la teoría de la complejidad en optimización combinatoria*. https://dialnet.unirioja.es/descarga/articulo/4733827.pdf

Dreo, J., Petrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristics for Hard Optimization*. Springer.

Fera, M., Fruggiero, F., Lambiase, A., Macchiaroli, R., & Todisco, V. (2018). A modified genetic algorithm for time and cost optimization of an additive manufacturing single-machine scheduling. *International Journal of Industrial Engineering Computations*, *9*(4), 423–438. https://doi.org/10.5267/j.ijiec.2018.1.001

Fera, M., Macchiaroli, R., Fruggiero, F., & Lambiase, A. (2020). A modified tabu search algorithm for the single-machine scheduling problem using additive manufacturing technology.

International Journal of Industrial Engineering Computations, *11*(3), 401–414. https://doi.org/10.5267/j.ijiec.2020.1.001

Gibson, I., Rosen, D., & Stucker, B. (2015). *Additive Manufacturing Technologies* (Second Edi). Springer. https://doi.org/10.1007/978-1-4939-2113-3

Glover, F., & Melián, B. (2003). Búsqueda tabú. *Revista Iberoamericana de Inteligencia Artificial. No*, *19*, 29–48. http://www.aepia.org/revista

Hillier, F., & Lieberman, G. (2010). *Introducción a la investigación de operaciones* (Novena edición). McGraw Hill.

Jiang, R., Kleer, R., & Piller, F. T. (2017). Predicting the future of additive manufacturing: A Delphi study on economic and societal implications of 3D printing for 2030. *Technological Forecasting and Social Change*, *117*, 84–97. https://doi.org/10.1016/j.techfore.2017.01.006

José, D., & Delgado, E. M. (2012). *Optimización de la programación (scheduling) en Talleres de Mecanizado*. Universidad Politécnica De Madrid.

Karimi, S., Kwon, S., & Ning, F. (2021). Energy-aware production scheduling for additive manufacturing. *Journal of Cleaner Production*, *278*. https://doi.org/10.1016/j.jclepro.2020.123183

Kim, Y. J., & Kim, B. S. (2022). Part-grouping and build-scheduling with sequence-dependent setup time to minimize the makespan for non-identical parallel additive manufacturing machines. *International Journal of Advanced Manufacturing Technology*, *119*(3–4), 2247–2258. https://doi.org/10.1007/s00170-021-08361-z

Kucukkoc, I. (2019). MILP models to minimise makespan in additive manufacturing machine scheduling problems. *Computers and Operations Research*, *105*, 58–67. https://doi.org/10.1016/j.cor.2019.01.006

Kucukkoc, I. (2021). Metal Additive Manufacturing: Nesting vs. Scheduling. In *Optimization and Data Science: Trends and Applications*. Springer Nature. http://www.springer.com/series/15947

Kucukkoc, I., Li, Q., He, N., & Zhang, D. (2018). Scheduling of Multiple Additive Manufacturing and 3D Printing Machines to Minimise Maximum Lateness. *Twentieth International Working Seminar on Production Economics, Innsbruck, Austria, 1*, 237–248.

Li, Q., Kucukkoc, I., & Zhang, D. Z. (2017). Production planning in additive manufacturing and 3D printing. *Computers and Operations Research*, *83*, 1339–1351. https://doi.org/10.1016/j.cor.2017.01.013

Li, Q., Zhang, D., Kucukkoc, I., & He, N. (2020). Heuristic Techniques for Real-Time Order Acceptance and Scheduling in Metal Additive Manufacturing. In *Mathematical Modelling and Optimization of Engineering Problems*. Springer Nature Switzerland. https://doi.org/10.1007/978-3-030-37062-6

Li, Q., Zhang, D., Wang, S., & Kucukkoc, I. (2019). A dynamic order acceptance and scheduling approach for additive manufacturing on-demand production. *International Journal of Advanced Manufacturing Technology*, *105*(9), 3711–3729. https://doi.org/10.1007/s00170-019-03796-x

López, R. (2010). *Algunos problemas clásicos de Optimización Combinatoria: una propuesta metodológica*.

Lozano, L., & Torres, F. (2018). *Solución al "Flexible Job Shop Problem" con Tiempos de Alistamiento Dependientes de la Secuencia Mediante un Algoritmo Híbrido Genético*. Universidad Industrial de Santander.

Merlano Canoles, K. L., & Castellanos Pico, K. L. (2021). *Un Algoritmo de Búsqueda Tabú para El Problema de Enrutamiento de un Recolector (SPRP) en un Almacén de Comercio Electrónico con Almacenamiento Disperso y Múltiples Depósitos* [Tesis de pregrado en Ingeniería Industrial]. Universidad Industrial de Santander.

Morera Mora, Á. (2021). *Programación de la producción en una máquina de fabricación aditiva* [Trabajo Fin de Grado en Ingeniería de Organización Industrial]. Universidad de Sevilla.

Ngo, T., Kashani, A., Imbalzano, G., Nguyen, K., & Hui, D. (2018). Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. *Composites Part B, 143*, 172–196. https://doi.org/10.1016/j.compositesb.2018.02.012

Riojas, A. (2005). *Conceptos, algoritmo y aplicación al problema de las N-reinas. Capítulo3. Búsqueda de tabú* [Universidad Nacional Mayor De San Marcos]. https://sisbib.unmsm.edu.pe/bibvirtualdata/monografias/basic/riojas_ca/cap3.pdf

Rohaninejad, M., Tavakkoli-Moghaddam, R., Vahedi-Nouri, B., Hanzálek, Z., & Shirazian, S. (2021). A hybrid learning-based meta-heuristic algorithm for scheduling of an additive manufacturing system consisting of parallel SLM machines. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2021.1987550

Sadrehaghighi, I. (2022). *Optimization Problem*. https://www.researchgate.net/publication/361306781_Optimization_Problem#pf4

Salazar-Hornig E, & Medina-SJC. (2013). Makespan Minimization for The Identical Machine Parallel Shop with Sequence Dependent Setup Times Using a Genetic Algorithm. *Ingeniería Investigación y Tecnología, Volumen XIV (Número1), Enero-Marzo 2013: 43-51* .

Senovilla Minguela, J. J. (2019). *Procesos de Optimización de Fabricación Aditiva* [Tesis de pregrado en Ingeniería en Organización Industrial]. Universidad de Valladolid.

Toksarı, M. D., & Toğa, G. (2022). Single batch processing machine scheduling with sequence-dependent setup times and multi-material parts in additive manufacturing. *CIRP Journal of Manufacturing Science and Technology*, *37*, 302–311. https://doi.org/10.1016/j.cirpj.2022.02.007

Ying, K. C., Fruggiero, F., Pourhejazy, P., & Lee, B. Y. (2022). Adjusted Iterated Greedy for the optimization of additive manufacturing scheduling problems. *Expert Systems with Applications*, *198*. https://doi.org/10.1016/j.eswa.2022.116908