UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Estadística e Investigación Operativa Aplicadas y Calidad

Heurísticas para el problema de taller de flujo de permutación distribuido heterogénea

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Análisis de Datos, Mejora de Procesos y Toma de Decisiones

AUTOR/A: Cózar Terrón, Hugo de

Tutor/a: Vallada Regalado, Eva

Cotutor/a: Villa Juliá, María Fulgencia

CURSO ACADÉMICO: 2022/2023

Resumen

En este trabajo se estudia el problema de permutación distribuida heterogénea o DHPFSP (distributed heterogeneous permutation flowshop scheduling problem). Este problema pertenece al ámbito de la secuenciación de la producción, rama de la Investigación Operativa. Esta configuración de la producción es especialmente relevante en la economía actual, en la que una empresa puede disponer de varios centros de producción. Se caracteriza el DHPFSP para posteriormente proponer cinco heurísticas, cuyo rendimiento se analiza estadísticamente mediante un ANOVA, teniendo en cuenta distintos factores como el número de trabajos, y las interacciones de primer orden. Sobre todas las heurísticas se aplica una búsqueda local con el fin de mejorar la calidad de la solución, y se mide el rendimiento de la misma. Adicionalmente, se realiza un estudio sobre el coste computacional de las distintas heurísticas, y se analiza cómo cambia el valor esperado de la función objetivo según lo heterogéneo que sea la instancia del problema. Finalmente se sugiere la mejor heurística para resolver el problema teniendo en cuenta todos los factores estudiados.

Palabras clave: Optimización, Heurísticas, Secuenciación distribuida, Taller de flujo de permutación

Resum

En aquest treball s'estudia el problema de permutació distribuïda heterogènia o DHPFSP (distributed heterogeneous permutation flowshop scheduling problem). Aquest problema pertany a l'àmbit de la seqüenciació de la producció, branca de la Investigació Operativa. Aquesta configuració de la producció és especialment rellevant en l'economia actual, en la qual una empresa pot disposar de diversos centres de producció. Es caracteritza el DHPFSP per a posteriorment proposar cinc heurístiques, el rendiment de les quals s'analitza estadísticament mitjançant un ANOVA, tenint en compte diferents factors com el nombre de treballs, i les interaccions de primer ordre. Sobre totes les heurístiques s'aplica una cerca local amb la finalitat de millorar la qualitat de la solució, i es mesura el rendiment d'aquesta. Addicionalment, es realitza un estudi sobre el cost computacional de les diferents heurístiques, i s'analitza com canvia el valor esperat de la funció objectiu segons l'heterogeni que siga la instància del problema. Finalment se suggereix la millor heurística per a resoldre el problema tenint en compte tots els factors estudiats.

Paraules clau: Optimització, Heurístiques, Seqüenciació distribuïda, Taller de flux de permutació

Abstract

In this paper we study the distributed heterogeneous permutation flowshop scheduling problem (DHPFSP). This problem belongs to the field of production sequencing, a branch of Operations Research. This production configuration is particularly relevant in today's economy, where a company may have several production centers. The DHPFSP is characterized and then five heuristics are proposed, whose performance is statistically analyzed by means of an ANOVA, taking into account different factors such as the number of jobs, and first-order interactions. On all heuristics a local search is applied in order to improve the quality of the solution, and the performance of the solution is measured. Additionally, a study on the computational cost of the different heuristics is performed, and it is analyzed how the expected value of the objective function changes according to the heterogeneity of the problem instance. Finally, we suggest the best heuristic to solve the problem taking into account all the factors studied.

Key words: Optimization, Heuristics, Distributed Scheduling, Permutation Flowshop

Agradecimientos

En primer lugar, quiero expresar mi agradecimiento a mis tutoras María Fulgencia Villa y Eva Vallada, sin las cuales no habría sido posible realizar este Trabajo Fin de Máster.

En segundo lugar, le agradezco a ValgrAI su labor por democratizar el acceso al sector de las nuevas tecnologías, especialmente al apartado de análisis de datos e inteligencia artificial. Gracias a ellos me ha sido posible centrarme en estos estudios, aparte de asistir a eventos y congresos que han sido muy inspiradores.

También me gustaría agradecer a Rafael Martí y Rubén Ruiz las clases recibidas durante la realización del máster. Gracias a ellas descubrí la motivación para profundizar en las heurísticas y metaheurísticas, eligiendo este tema para realizar mi Trabajo Fin de Máster.

Por último, le agradezco a mi padre Jose Manuel de Cózar su apoyo, el cual he recibido en todo momento. Sus consejos, paciencia y aliento han sido una parte fundamental de que este trabajo haya podido salir adelante.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y Objetivos

1.1. Introducción

Un sistema productivo se entiende como un conjunto de máquinas, en una configuración determinada, que procesa un conjunto de órdenes. Las configuraciones más usuales son en serie, en paralelo o una mezcla de ambas. Por otro lado, una orden se compone de un número de piezas a fabricar, que deben pasar por una serie de máquinas determinadas para recibir operaciones (es decir, en cada máquina sufren una transformación). Según el número de piezas, la velocidad de la máquina y el tipo de operación, cada orden tendrá un tiempo de procesamiento en cada máquina.

Un ejemplo real de uno de estos sistemas lo encontramos en el ámbito del automóvil. En una línea de montaje de este tipo se debe procesar órdenes de un número determinado de coches de distintos tipos. Tenemos varias máquinas (o estaciones) en serie (montaje, soldadura, pintado, etc.), con lo que las órdenes irán pasando por cada máquina, donde tendrá que esperar un tiempo determinado antes de pasar a la siguiente.

Una buena planificación en estos sistemas es muy importante, ya que a igualdad de recursos, es posible tener mejores o peores resultados según la secuencia que se escoja para las órdenes. Esto es lo que se conoce como optimización de la producción. El objetivo es obtener la mejor secuencia posible para lograr los mejores valores de la función objetivo. Esta función es escogida por adelantado y está relacionada con mejorar los tiempos de entrega, disminuir retrasos o el uso de recursos, según convenga (entre otros posibles objetivos).

La optimización de la producción pertenece al ámbito de la Investigación Operativa, que surge durante la segunda guerra mundial para mejorar la fabricación y logística en el ámbito bélico. Algunos de sus primeros proponentes fueron S. M. Johnson, creador de la regla homónima para optimizar un sistema de dos máquinas en paralelo, y H. L. Gantt, creador del gráfico Gantt para la visualización de secuencias en los trabajos.

Hoy en día, existe numerosa literatura científica sobre la optimización de la producción, dedicada a las distintas variantes de sistemas productivos. Una de estas configuraciones es el Flowshop de Permutación. Este es un sistema de máquinas en serie en el que el orden de los trabajos es el mismo en todas las máquinas. Además, cada orden debe ser procesada por completo en una máquina antes de pasar a la siguiente. Este trabajo presenta y optimiza una variante de este problema.

El problema DPFS (*Distributed Permutation Flowshop Scheduling*) ha recibido considerable atención desde que en 2010 B. Naderi y R. Ruiz publicaran el artículo original en el que se aborda este problema. Esta secuenciación distribuida es de gran relevancia en la actualidad, ya que las operaciones productivas de muchas grandes empresas se desarrollan en fábricas situadas en localizaciones diferentes. Además, estas fábricas trabajan con un sistema de lotes, que facilita una mayor personalización y variedad de producto frente a la producción en serie o continua. Por último, la configuración de las fábricas suele ser de máquinas en serie, o flowshop, que van aplicando sucesivas operaciones de transformación a la materia prima y semi elaborados.

Hasta la fecha, la investigación sobre el DPFS se ha centrado en las variantes homogéneas del mismo. Esto significa, que los trabajos tienen el mismo tiempo de procesamiento en las distintas fábricas. Esta simplificación resta realismo al problema, ya que en la realidad cada taller tiene diversas características que hacen que un mismo trabajo sea procesado a distintas velocidades. Por lo tanto, en este trabajo nos hemos centrado en la variante heterogénea del problema, que no ha sido muy estudiada hasta ahora.

Un ejemplo de este sistema productivo, al que se le podrían aplicar la metodología propuesta en este trabajo, es la empresa de automóviles americana Local Motors. Esta empresa utiliza una plataforma en línea donde la clientela pueden presentar ideas de diseño para automóviles. La comunidad vota por los mejores diseños, y luego la sección de ingeniería trabaja en el desarrollo del prototipo utilizando tecnologías como la impresión en 3D y el corte láser. Una vez que se finaliza el diseño y se aprueba, los componentes se fabrican en distintas microfactorías y se ensamblan en los puntos de venta de Local Motors. Este enfoque de fabricación distribuida permite a Local Motors reducir los costos de producción, acelerar el tiempo de desarrollo y adaptar rápidamente los diseños a las demandas del mercado local. Además, fomenta la participación de la comunidad y promueve la innovación colaborativa en la industria automotriz.

A lo largo de este trabajo, se describirá en profundidad el problema al que se enfrentan las personas encargadas de la gestión de este tipo de sistema productivo. Después se cubrirán los trabajos realizados por otros investigadores hasta la fecha, se propondrán cinco heurísticas (dos de ellas originales) y se probarán mediante un set de instancias.

1.2. Objetivos

Los objetivos de este trabajo son los siguientes:

- Introducir el problema de DPFS heterogénea (conocido en inglés como DHPFS).
- Desarrollar heurísticas eficientes y eficaces para resolver el problema.
- Estudiar el impacto de la heterogeneidad del problema en la calidad de la solución.

1.3. Estructura del documento

Este Trabajo Final de Máster está formado por seis capítulos cuya identificación y breve descripción se muestra a continuación:

- **Capítulo 1. Introducción**: en este apartado se introduce de manera general el problema objeto de este trabajo, se detallan los conceptos básicos para el entendimiento de dicho problema y se detallan los objetivos que se espera alcanzar.
- **Capítulo 2. Descripción del problema**: se ofrece una descripción detallada tanto del problema general de la DPFS, como la variante particular abordada en este trabajo, la DPFS heterogénea.
- **Capítulo 3. Revisión bibliográfica**: se muestran las publicaciones más relevantes y recientes respecto a la secuenciación de una DPFS, abordando sus distintas variantes y métodos empleados para resolverlas. También se expondrá la aportación que hace este trabajo a la línea actual de investigación.
- **Capítulo 4. Metodología**: en este capítulo se muestran las diferentes heurísticas diseñadas para resolver el problema.
- **Capítulo 5. Análisis de resultados**: se analiza el rendimiento de las distintas heurísticas empleadas. Para ello, se propone un set de instancias de diferentes tamaños y características.

- **Capítulo 6. Conclusiones y líneas futuras**: se exponen las conclusiones del trabajo y los objetivos alcanzados. Por último, se comentan algunos aspectos en los cuales se podrá mejorar en un futuro.

Capítulo 2

Descripción del problema

2.1. El problema de la secuenciación

El problema de la secuenciación se da en muchos ámbitos prácticos, como el reparto del trabajo en plantas de produccion, el transporte y la logística, o las operaciones portuarias, entre muchos otros. Los problemas de asignación consisten en determinar qué recursos (máquinas, operarios o herramientas, por ejemplo) deben ser asignados a cada tarea para su realización, con el fin de optimizar un cierto objetivo. Por otro lado, los problemas de secuenciación consisten en determinar el orden en el que se realizan las tareas de tal forma que se optimice la función objetivo escogida. Generalmente, esta guarda relación con maximizar la eficiencia y minimizar los tiempos de producción, cumplir con los plazos requeridos, optimizar el uso de los recursos disponibles o reducir los costos asociados.

La secuenciación de órdenes es un concepto utilizado en la gestión de la producción y la planificación de operaciones para determinar el orden en el que se deben ejecutar las tareas o actividades en un proceso de producción. Consiste en establecer una secuencia lógica y eficiente para llevar a cabo las diferentes órdenes de trabajo o pedidos de la mejor manera posible.

En la secuenciación de órdenes se consideran diferentes factores y criterios para determinar el orden de ejecución de las tareas. Algunos de los factores comunes que se tienen en cuenta incluyen:

- Restricciones de dependencia: algunas tareas pueden depender del resultado de otras, por lo que es necesario secuenciarlas de manera que las tareas precedentes se completen antes de comenzar las siguientes.
- Prioridades: algunas órdenes o tareas pueden tener mayor prioridad debido a la importancia del cliente, la urgencia del pedido o la rentabilidad asociada. Estas prioridades se consideran al establecer la secuencia de ejecución.
- Fechas de entrega: si existen plazos específicos para la entrega de las órdenes, es necesario tener en cuenta estos plazos al secuenciar las tareas, asegurando que se completen a tiempo.
- Capacidad y disponibilidad de recursos: se debe considerar la capacidad y disponibilidad de los recursos necesarios para llevar a cabo las tareas, como la maquinaria, el personal y los materiales. Esto implica programar las tareas de acuerdo con la capacidad de los recursos para evitar cuellos de botella o retrasos.

Existen diversos métodos y algoritmos utilizados para la secuenciación de órdenes, desde reglas de despacho hasta complejas metaheurísticas. Los modelos matemáticos de programación lineal entera mixta (MILP) son una opción para obtener soluciones óptimas en problemas con tamaños de muestra pequeños.

2.2. Secuenciación en talleres de flujo

La secuenciación en talleres de flujo, también conocida como FSP (Flowshop Scheduling Problem), es un problema de optimización de la producción que involucra el ordenamiento eficiente de las tareas en un proceso de producción secuencial. Esta área de investigación surge con Johnson (1954) y ha sido muy prolífica desde entonces.

En un taller de flujo, hay una serie de máquinas en las que se deben realizar las operaciones de producción. Cada tarea o trabajo pasa a través de todas las máquinas, y una vez que comienza una tarea en una máquina, debe completarse antes de pasar a la siguiente (máquinas en serie). El objetivo es encontrar la secuencia óptima que minimice una métrica de rendimiento, como el makespan (tiempo máximo de finalización de todos los trabajos).

El FSP plantea desafíos debido a la combinación de restricciones de ordenamiento y de asignación de tareas a las máquinas. Existen diferentes variantes del problema PFSP, como el FSP clásico (donde todas las tareas tienen la misma secuencia en todas las máquinas) y el FSP flexible (donde las tareas pueden tener diferentes secuencias en diferentes máquinas).

La secuenciación en talleres de flujo es un problema NP-difícil, lo que implica que no existen algoritmos eficientes para resolverlo de manera exacta en tiempos razonables para problemas grandes. Por lo tanto, se utilizan enfoques heurísticos y algoritmos de optimización para encontrar soluciones aproximadas o subóptimas.

Los N trabajos deben pasar por las M máquinas, por lo que cada trabajo $j, j \in N$ está compuesto por m tareas. Los tiempos de procesamiento son conocidos por adelantado, no negativos y deterministas. Su notación es $p_{i,j}, j \in N, i \in M$

El FSP conlleva una serie de supuestos:

- Todos los trabajos son independientes y están disponibles para ser procesados en el tiempo 0.
- Las máquinas están disponibles continuamente (sin averías).
- Cada máquina sólo puede procesar un trabajo a la vez.
- Una vez iniciado el procesamiento de un trabajo en una máquina determinada no puede interrumpirse y el procesamiento continúa hasta su finalización.
- Los tiempos de preparación son independientes de la secuencia y se incluyen en los tiempos de proceso o se ignoran.
- Se permite el almacenamiento infinito durante el proceso.
- Los trabajos no pueden pasar a la máquina siguiente hasta haber sido finalizados por completo en la anterior.

El objetivo es obtener una secuencia de producción de los *N* trabajos en las *M* máquinas de forma que se optimice un criterio dado. Normalmente se define una secuencia de producción para cada máquina. Dado que existen *n!* permutaciones posibles de trabajos por máquina, el número total de secuencias posibles es $(n!)^m$. Sin embargo, una simplificación común en la literatura es suponer que se mantiene el mismo orden de los trabajos (permutación) en todas las máquinas. Esto prohíbe el paso de trabajos entre máquinas y reduce el espacio de soluciones a *n!*. Si el problema presenta esta simplificación, se denomina problema de programación de flujos de permutación o PFSP.

Los tiempos de finalización de los trabajos en las máquinas tienen la notación $C_{j,i}, j \in N, i \in M$. Una vez que se ha determinado una secuencia de producción o permutación de trabajos (sea esta permutación π , y sea el trabajo que ocupa la posición *j*, $\pi_{(j)}$), los tiempos de finalización se calculan fácilmente con la siguiente expresión recursiva:

$C_{\pi(j),i} = max\{C_{\pi(j),i-1}, C_{\pi(j-1),i}\} + p_{i,j}$

donde $C_{j,0} = 0$.

Una de las funciones objetivo más habituales para este problema se trata de la minimización del máximo tiempo de finalización (también denotado como makespan o C_{max}).

La figura 2.1 representa una secuencia de dos trabajos en un flowshop de permutación con tres máquinas. El primer trabajo ha sido colocado antes que el segundo, por lo que irá antes en todas las máquinas. Se puede observar que, para que un trabajo comience a ser procesado en una máquina, se deben cumplir dos condiciones:

- La tarea en la máquina anterior debe haber sido terminada por completo. Por ejemplo, en la máquina $M2$, el segundo trabajo no comienza hasta que no ha acabado su primera tarea en la máquina $M1$
- La máquina debe estar libre. Es decir, no debe estar procesando ningún trabajo actualmente. Por ejemplo, en la máquina $M1$, el trabajo dos comienza cuando termina el trabajo uno.

C_{max} es el tiempo de finalización del último trabajo en la última máquina (en este caso, el trabajo número dos). Ejemplos más complejos incluyen un mayor número de máquinas o de trabajos, pero siguen una estructura similar.

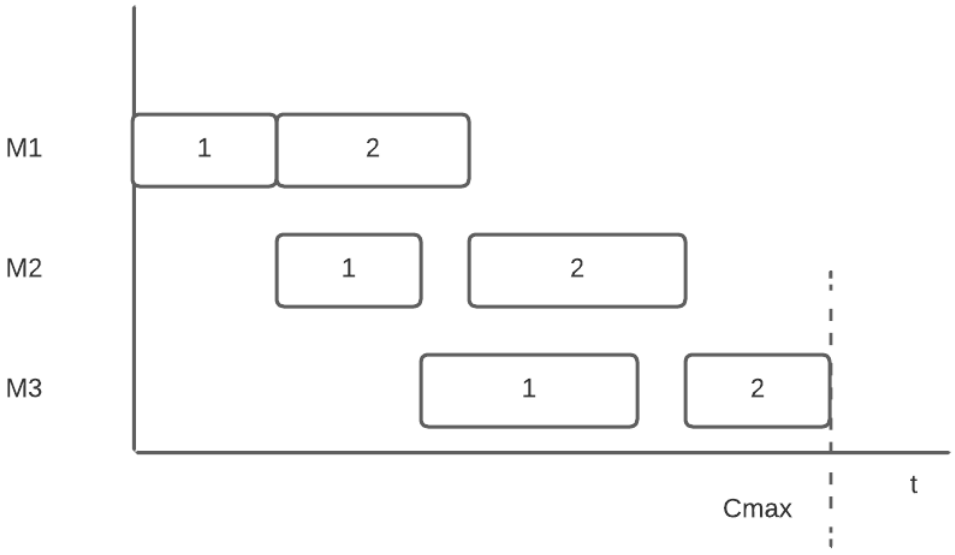

Figura 2.1: Ejemplo de diagrama de Gantt para un flowshop de permutación

2.3. DPFS

Los Flowshops de Permutación Distribuidos (DPFS, por sus siglas en inglés) son una variante del problema clásico de programación de talleres de flujo de permutación (PFSP) en el cual las tareas se distribuyen en múltiples ubicaciones o entidades en lugar de estar centralizadas en un único taller. Además, estas ubicaciones se organizan en un flowshop, o taller de máquinas en serie. De nuevo el objetivo es optimizar la secuenciación, lo que se traduce en mejores valores de la función objetivo escogida.

En otras palabras, un conjunto de N trabajos debe ser secuenciado en F fábricas, cada una formada por M máquinas en una configuración en serie (flowshop o PFSP). Resolver este problema implica adoptar dos

decisiones: la asignación de los trabajos a las fábricas, y la secuenciación del conjunto de trabajos resultante, en cada fábrica. Al ser los flowshops de permutación, los trabajos siempre visitan las máquinas en el mismo orden.

Para resolver el problema de manera óptima sería necesario generar todas las soluciones y escoger la mejor. El número de soluciones posible puede calcularse de la siguiente manera: deben repartirse N trabajos en F fábricas, trabajos que luego pueden adoptar cualquier permutación. Multiplicamos el número de soluciones del problema de reparto (número combinatorio) con el de ordenación (n factorial), obteniendo la siguiente expresión: $\binom{n+f-1}{n} \cdot n! = \frac{(n+f-1)!}{(f-1)!}$. Esta fórmula fue desarrollada por Ruiz y Naderi (2010). Como vemos, el número de soluciones posibles crece de manera explosiva a medida que aumenta N, lo que hace imposible obtener la solución óptima para instancias que no sean muy pequeñas.

Las suposiciones aceptadas en este trabajo son las que se suelen asumir en los problemas de secuenciación de flowshops. El único cambio consiste en el tiempo de procesamiento de los trabajos, a los que se debe añadir un nuevo subíndice para indicar la fábrica. Así, el trabajo *j* de una máquina *i* en la fábrica f tiene tiempo de procesamiento p_{ijf}. Este puede ser igual en todas las fábricas (variante homogénea) o no (variante heterogénea).

En la figura 2.2 podemos ver un ejemplo de una DPFS sencilla. Los trabajos a secuenciar se encuentran en la parte izquierda. En este caso, se trata de un set J de cinco trabajos, que deben ser repartidos en un set F de tres fábricas. Cada fábrica sigue una configuración en serie de tres máquinas en este caso. Después de asignar los trabajos a cada fábrica, será necesario determinar el orden en el que son procesados.

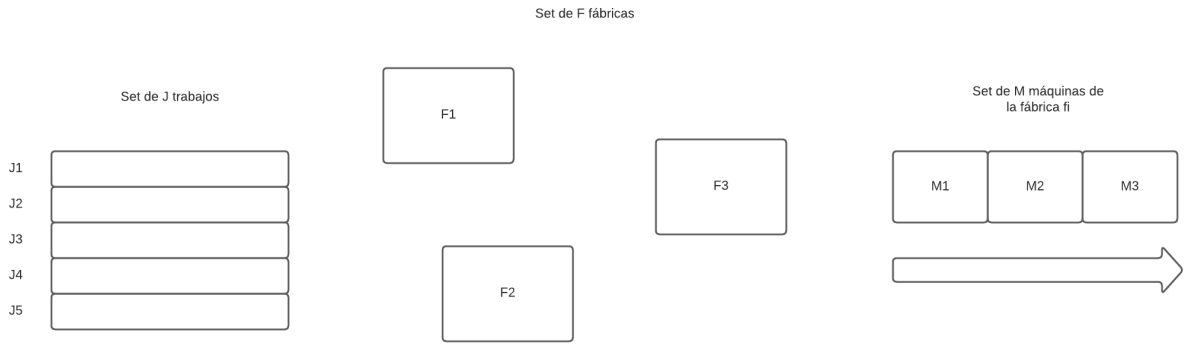

Figura 2.2: Esquema de una DPFS

2.4. DPFS heterogéneo

Finalmente, el DPFS heterogéneo es el problema que se aborda en este trabajo. Hasta la fecha, la mayoría de artículos que tratan el DPFS se centran en la variante homogénea (Framinan y Perez-Gonzalez 2022), que asume que todos los *N* trabajos tienen los mismos tiempos de procesamiento en cada una de las *F* fábricas. Esto es una simplificación del problema, ya que en la realidad cada fábrica cuenta con una serie de características que la hacen más o menos eficiente a la hora de procesar un trabajo en particular, o todos los trabajos en general:

- Habilidad y experiencia de operarios
- Estado del equipo
- Máquinas no idénticas: variaciones de antigüedad y modelos

Por lo tanto, es razonable asumir que los trabajos pueden tener diferentes tiempos de proceso en cada fábrica. Esto es lo que se conoce como el DPFS heterogéneo, denotado como (RF|prmu|γ) en la notación propuesta en Framinan y Perez Gonzalez (2022). *prmu* significa que los flowshop son de permutación mientras que γ es la función objetivo escogida en cada caso. Las funciones objetivo más comunes son minimizar la fecha de

finalización del último trabajo C_{max} o minimizar el tiempo total de procesamiento $\sum_{j=1}^{n} C_j$. Específicamente, en este trabajo se ha evaluado la calidad de las soluciones mediante el objetivo C_{max}.

Capítulo 3

Revisión bibliográfica

3.1. El problema de la secuenciación

En primer lugar, ha sido necesario sentar las bases de los problemas de secuenciación clásicos en el ámbito de la producción, especialmente el Flowshop de Permutación (PFSP por sus siglas en inglés). Este es un problema de secuenciación que involucra una configuración en serie de las máquinas. Además, los trabajos mantienen el orden que tienen en la secuencia, para todas las máquinas. En el DPFS, este es el sistema que opera en cada una de las fábricas, por lo que es relevante.

La principal referencia en este sentido ha sido el famoso libro de Michael L. Pinedo, publicado por primera vez en 2002. En esta obra se cubren todos los aspectos básicos de la secuenciación: los distintos tipos de problemas (una máquina, máquinas en paralelo, flowshops y jobshops), distintas funciones objetivo (makespan, weighted tardiness) y distintos métodos para obtener soluciones (heurísticas, reglas de despacho o métodos como algoritmos genéticos y búsquedas locales). Gracias a esta obra se han podido comprender los fundamentos de la secuenciación, haciendo posible la comprensión de lecturas posteriores y la implementación informática de los algoritmos. Por ejemplo, en este libro se da una fórmula que permite calcular el tiempo de finalización de cada trabajo en un flowshop. Esta fórmula ha sido necesaria para la construcción de todas las heurísticas presentadas en este trabajo. El concepto de búsqueda local también se explica en este libro, dando algunos ejemplos de implementaciones concretas (como movimientos swap, de inserción, etc.) que permiten mejorar la calidad de las soluciones. En este trabajo se ha implementado y evaluado una búsqueda local como las expuestas en el libro.

Para comprender en profundidad el problema del PFSP específicamente, se ha consultado Ruiz y Maroto (2005), un artículo en el que se exponen las principales heurísticas para resolver este tipo de taller. La heurística NEH, que tiene gran peso en este trabajo, es citada aquí, aunque proviene de un trabajo anterior de Nawaz et al (1983). En este artículo también se discuten otras heurísticas, metaheurísticas y métodos de búsqueda local.

3.2. El problema de la DPFS

El artículo germinal de la DPFS es realizado por Rubén Ruiz y B. Naderi en 2010, a partir del cual surge la línea de investigación actual. Este es el primer artículo que simula la configuración de la DPFS y aporta heurísticas sencillas para solucionarla, por lo que muchos de los artículos más recientes citan a este como su principal referencia. Algunas de las estrategias para resolver una DPFS, empleadas en este trabajo, provienen de este artículo, como aplicar la heurística NEH en cada una de las fábricas.

Actualmente existen numerosos artículos respecto a la DPFS. Recientemente, Perez-Gonzalez y Framinan (2023) realizaron una revisión del estado del arte del problema, mostrando la gran cantidad de variantes, funciones objetivo y restricciones que se han añadido al problema. Este artículo ha sido muy importante, ya que en

él se recogen todos los artículos que tratan sobre la variante heterogénea de la DPFS (la tratada en este trabajo), lo que ha permitido al autor identificarlos y consultarlos con facilidad. Además, se detalla qué variaciones (funciones objetivo y restricciones distintas) se han probado, facilitando la tarea de realizar aportaciones originales. En este caso, el estudio del impacto de la heterogeneidad, que no se había llevado a cabo hasta ahora.

Otros artículos consultados a este respecto incluyen Ruiz, Pan y Naderi (2018), en el que se emplean métodos Iterated Greedy para resolver la DPFS y Ruiz y Naderi (2014), en el que se usa Scatter Search. En un primer momento se contempló la posibilidad de implementar estas heurísticas para la variante heterogénea del problema, pero esto se ha dejado para trabajos posteriores.

3.3. Variantes de la DPFS

Se han consultado diversos artículos que cubren diferentes variantes de la DPFS con el objetivo de aprender y desarrollar técnicas que fueran extrapolables al caso heterogéneo. Una primera variante de la DPFS incluye los tiempos de cambio entre lotes, lo que se aborda en Ruiz, Hatami y Andres-Romano (2015). Otra variante es la DAPFS, en la que una máquina final (llamada assembler) debe realizar la última operación de todos los trabajos. Este problema es analizado por primera vez por Ruiz, Hatami y Andres-Romano (2013). Por último, Schulz, Schönheit y Neufeld analizan la DPFS multi objetivo y con restricciones de transporte. Estos artículos han sido útiles para familiarizarse con los distintos modos de la DPFS, aunque finalmente no se haya empleado directamente ninguna de las soluciones propuestas.

También se han revisado artículos especializados en la variante heterogénea de la DPFS, tema central de este trabajo. Wang et al. (2017) fue el primer artículo en abordar la variante heterogénea de la DPFS, en el que se analiza un caso con dos fábricas. Se trata de minimizar el número de trabajos con tardanza, para lo cual proponen un modelo MILP. De este artículo se extrae la impracticabilidad de emplear modelos de programación lineal para instancias que no sean muy pequeñas. En la tabla de tiempos de computacion que muestran puede observarse claramente como su crecimiento es exponencial a partir de tamaños de instancias reducidas.

Un artículo que trata sobre la DHPFS con tiempos de cambio es Meng y Pan (2021). Aunque en este trabajo TFM no se han tenido en cuenta tiempos de cambio, este artículo ha resultado muy útil ya que propone la heurística NEHb (heurística NEH aplicada a la DPFS), con dos variantes según la regla de decisión empleada. Esta estrategia ha sido implementada en este trabajo con buenos resultados (heurísticas NEHb1 y NEHb2). Además, sobre esta base se ha desarrollado una tercera variante por parte del autor (NEHb3).

Otro artículo que trata sobre la DHPFS, esta vez con fraccionamiento de lotes, es Li, Li y Gao (2021). De nuevo, aunque este trabajo no tenga en cuenta el fraccionamiento de lotes, ha sido posible adaptar una heurística desarrollada por estos autores. Se trata de la DNEH, sobre la cual el autor ha desarrollado una segunda variante.

Por lo tanto, estos dos artículos suponen el punto de partida más directo para este trabajo. Otros artículos consultados incluyen Shao, Shao y Pi (2023) y Wang, Li, Gao y Li (2021), que se centran en el aspecto de la eficiencia energética. No se han derivado soluciones directas de estos artículos, pero han sido útiles para mejorar la comprensión del problema.

3.4. Contribución de este trabajo al estado del arte

Como se indica en Perez-Gonzalez y Framinan (2023), la variante heterogénea de la DPFS ha sido poco estudiada. La revisión bibliográfica llevada a cabo por el autor confirma que sólo existen cinco artículos sobre el problema, lo que deja numerosas heurísticas y variantes para explorar. En este trabajo, el autor ha tratado de innovar en la búsqueda de algoritmos que generen soluciones eficientes para este problema.

En primer lugar, para realizar este trabajo ha sido necesario generar un set de instancias adaptado al DPFS heterogéneo. Este es el primer set de estas características abierto al público del que se tiene constancia. Para

hacerlo, el autor se ha basado en la Testebed TB1 de Naderi y Ruiz.

En segundo lugar, se han desarrollado y evaluado cinco heurísticas distintas. Algunas de ellas provienen de trabajos de otros autores. En otras, el autor ha contribuido a la literatura científica adaptando heurísticas de la versión regular de la DPFS a la heterogénea. Por último, también se han desarrollado dos heurísticas nuevas a partir de heurísticas anteriores.

En tercer lugar, se ha desarrollado un nuevo método de búsqueda local para este tipo de problema. De esta manera, se pueden mejorar soluciones obtenidas a partir de cualquier otra heurística.

En último lugar y más importante, se ha estudiado cómo el grado de heterogeneidad del problema afecta a la calidad de las soluciones. No se han encontrado artículos o referencias en la literatura en los que se haya tratado este punto, por lo que supone una aportación original.

Capítulo 4

Metodología

4.1. Heurísticas desarrolladas

Para resolver el problema se han adaptado y desarrollado una serie de algoritmos heurísticos, combinados con una búsqueda local posterior. Se ha decidido no emplear técnicas exactas (como un modelo MILP), ya que incluso para instancias pequeñas, los tiempos de procesamiento son extraordinariamente largos y escalan de manera exponencial. Naeri y Ruiz (2010) mencionan que el número posible de soluciones en una DPFS es muy superior al de una PFSP regular. Por ejemplo, con $n = 10$ y $F = 4$ tenemos $f_1(n, F) = \binom{n+F-1}{F-1} n! = \binom{13}{3} 10! = 1,037,836,800$ soluciones. Esto es un tamaño inmanejable computacionalmente incluso para una instancia tan pequeña.

4.1.1. Heurística NEH

Las heurísticas constructivas desarrolladas están basadas en la heurística NEH. Esta heurística fue propuesta por Newaz et al. (1983) y es considerada la mejor heurística para el PFSP actualmente, según Framinan y Perez-Gonzalez (2022). Se basa en la idea de que los trabajos con tiempos de procesamiento elevados en todas las máquinas deben programarse lo antes posible en la secuencia. El procedimiento para aplicarla es el siguiente:

Paso 1. Los tiempos totales de procesamiento de los trabajos se calculan de la siguiente manera: para cada trabajo i, con $i = 1, ..., n$, el tiempo de proceso total es $P_i = \sum p_{ij}$. Es decir, la suma de los tiempos de procesamiento en cada máquina. Los trabajos se ordenan en orden no creciente de P_i.

Paso 2. Se toman los dos primeros trabajos (aquellos dos con P_i más alto) y se realiza la asignación inicial. Se calculan dos secuencias (primer trabajo secuenciado primero y viceversa) y se selecciona la opción con menor tiempo de finalización.

Paso 3. Para el resto de trabajos, se toma el trabajo i, $i = 3, ..., n$, y se busca la mejor programación colocándolo en todas las posiciones i posibles de la secuencia de trabajos ya programados. Por ejemplo, si $i = 4$, la secuencia ya construida contendría los tres primeros trabajos de la lista ordenada calculada en el paso 2, entonces el cuarto trabajo podría colocarse en la primera, en la segunda, en la tercera o en la última posición de la secuencia. La mejor secuencia de las cuatro se seleccionaría para la siguiente iteración.

En la figura 4.1 se expone un ejemplo con tres máquinas y tres trabajos. En primer lugar, se calculan los tiempos de procesamiento totales, y se ordenan los trabajos de manera no creciente.

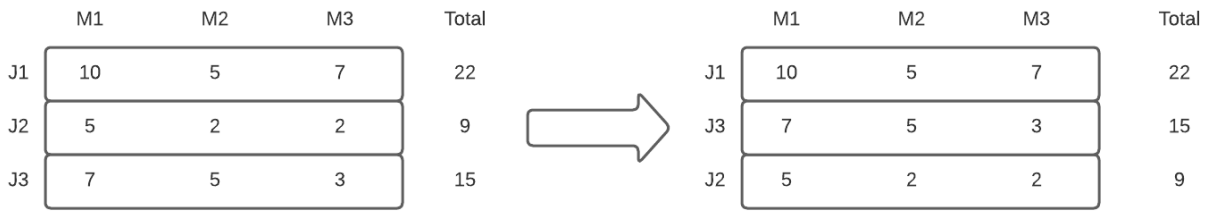

Figura 4.1: Primer paso de la heurística NEH

Se calcula la primera secuencia posible, procesar J1 y luego J3. Obtenemos un tiempo de finalización de 25 unidades de tiempo (figura 4.2).

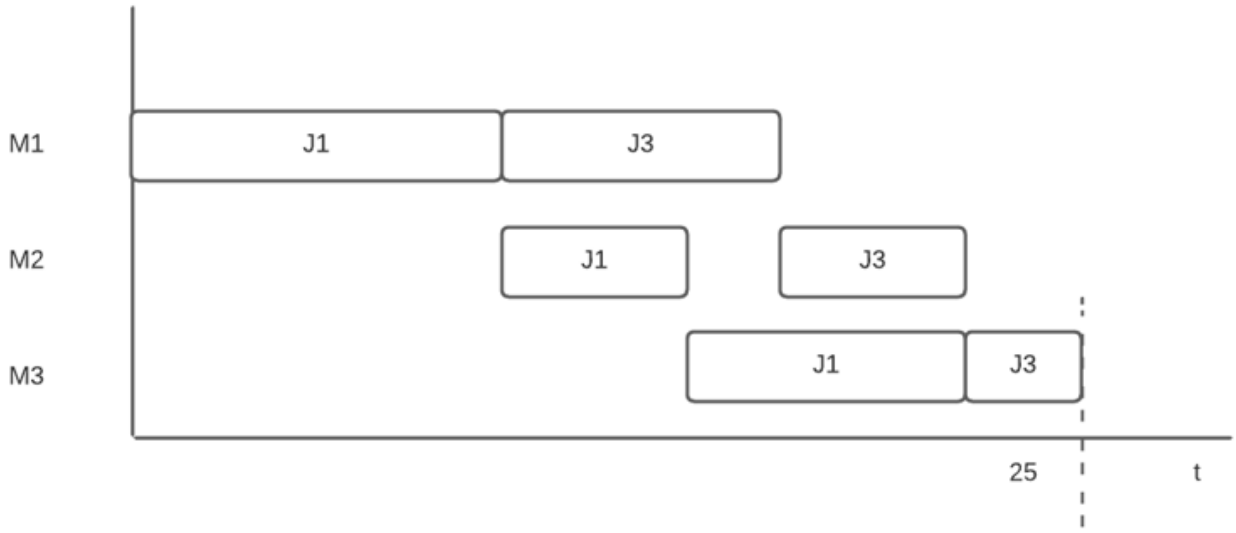

Figura 4.2: Segundo paso de la heurística NEH, primera secuencia

Se calcula la segunda secuencia, tal y como muestra la figura 4.2 (procesar J3 y luego J1). Obtenemos un tiempo de finalización de 29 unidades de tiempo (figura 4.3).

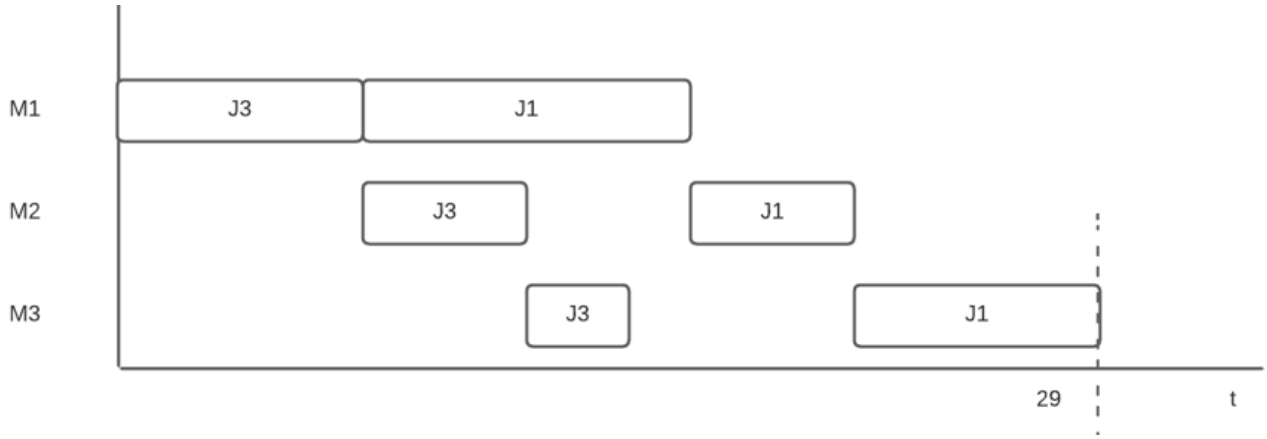

Figura 4.3: Segundo paso de la heurística NEH, segunda secuencia

Como la primera secuencia es mejor (tiene menor C_{max}), es la que resulta escogida finalmente. Para determinar la posición del segundo trabajo J2, se probará en todas las posiciones posibles y se calculará el tiempo de finalización, escogiendo finalmente la mejor secuencia.

Como la secuencia tres (figura 4.6) tiene el mejor tiempo de finalización, resulta escogida. Por lo tanto, aplicando la heurística NEH se llega a la secuencia final J1-J3-J2. Las tres secuencias posibles pueden verse en las figuras 4.4, 4.5 y 4.6.

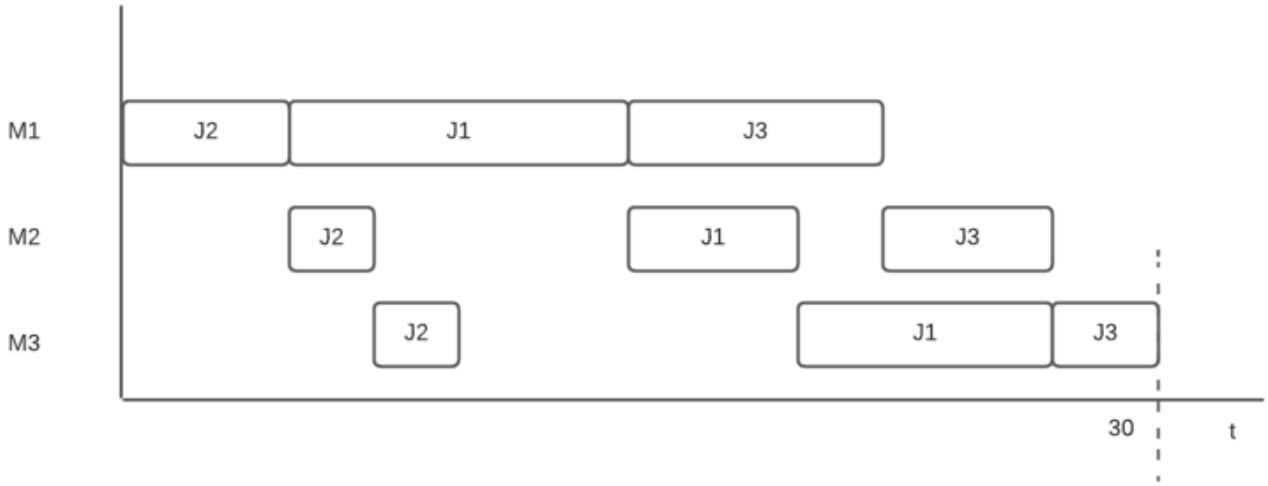

Figura 4.4: Tercer paso de la heurística NEH, primera secuencia posible

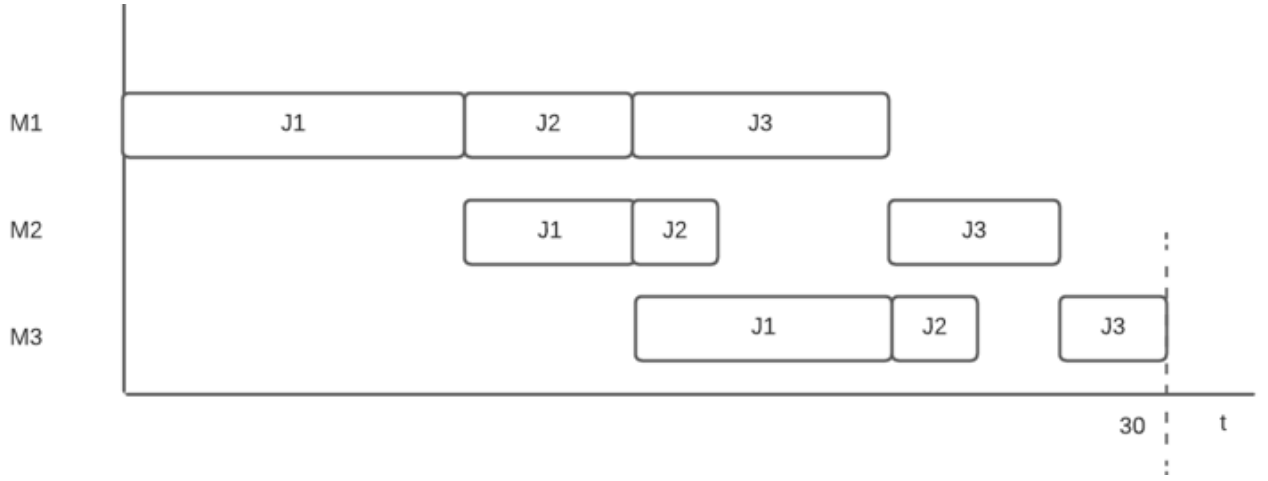

Figura 4.5: Tercer paso de la heurística NEH, segunda secuencia posible

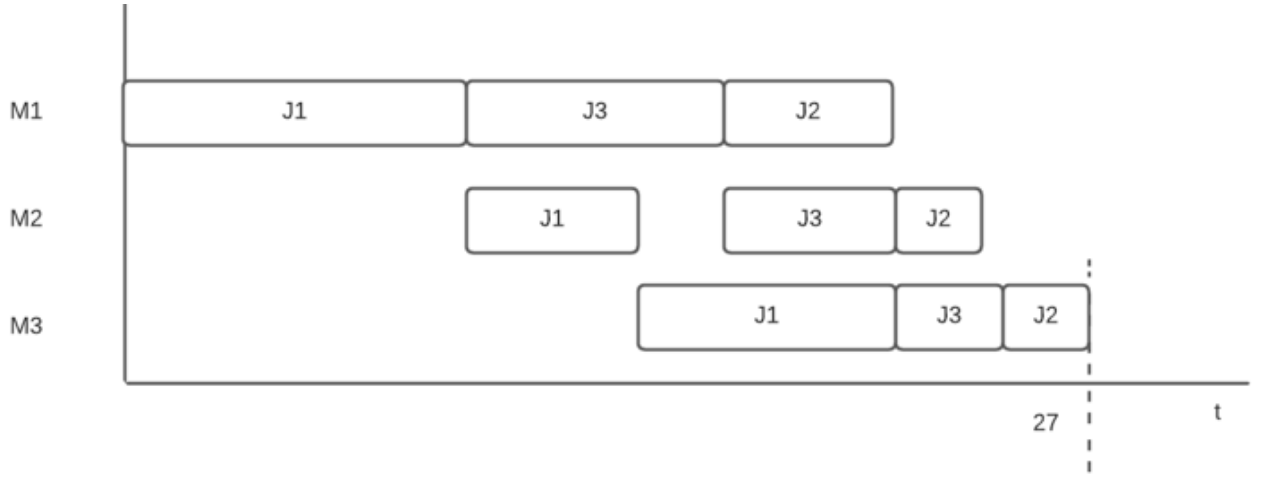

Figura 4.6: Tercer paso de la heurística NEH, tercera secuencia posible

4.1.2. Heurística NEHb

Los autores Meng et al. (2021) usan el punto de partida de la NEH para adaptar esta exitosa heurística a la DPFS, creando la heurística NEHb. Esta posee dos variantes (NEHb1 y NEHb2), según la regla de asignación final que se emplee. El procedimiento para aplicarla es el siguiente:

Paso 1. Sumar los tiempos de proceso totales de cada trabajo para cada fábrica. Así, si la instancia del problema tiene tres fábricas ($F = 3$), cada trabajo tendrá un vector de tamaño 3 con su P_j (tiempo de procesamiento) total en cada fábrica.

Paso 2. Ordenar los trabajos de mayor a menor tiempo de proceso total.

Paso 3. Asignar los primeros F trabajos en orden hasta que cada fábrica tenga un trabajo secuenciado. El primer trabajo de la lista irá a la fábrica $f = 1$, el segundo a la fábrica $f = 2$, y así sucesivamente.

El siguiente trabajo a secuenciar siempre se escoge de la lista ordenada. Para decidir a qué fábrica será asignado, se puede usar tanto la Regla 1 (variante NEHb1) como la Regla 2 (variante NEHb2). Posteriormente, en cada fábrica el trabajo será ordenado según la regla NEH.

Regla 1. El trabajo se secuencia en la fábrica con menor C_i actual. Se prueba en todas las posiciones posibles, como en la heurística NEH original, y se escoge la mejor secuencia.

Regla 2. El trabajo se prueba en todas las fábricas, escogiéndose finalmente la fábrica que presenta menor C_i total. Recordemos que para determinar el C_i de una fábrica, se prueba el trabajo en todas las posiciones, y se escoge la mejor secuencia.

A continuación, se expone un ejemplo de la aplicación de la heurística NEHb. Existen tres trabajos, dos fábricas, y tres máquinas en cada fábrica. En primer lugar, disponemos de los tiempos de procesamiento de los trabajos. Estos formarán una matriz NxM para cada fábrica:

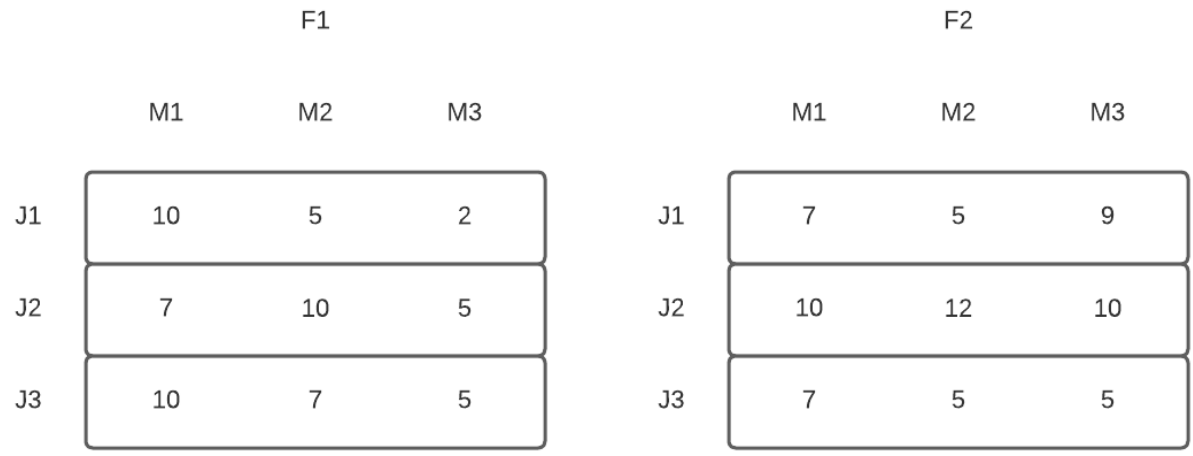

Figura 4.7: Tiempos de proceso de los trabajos

El primer paso consiste en sumar los tiempos de los trabajos en todas las máquinas y fábricas. Así, $p_1 = 38$, $p_2 = 54$ y $p_3 = 39$.

El segundo paso consiste en ordenar los trabajos de manera descendente según su p_j. Por lo tanto, formamos la lista ordenada [J2, J3, J1].

En el tercer paso, se asignan los primeros F trabajos. Por lo tanto, J2 es asignado a F1, y J3 es asignado a F2.

Si nos encontramos aplicando la heurística NEHb1, seguiremos la Regla no. 1. Calculamos los C_i en cada fábrica. La fábrica F1 tendrá un C_1 de 54, tiempo de finalización del trabajo J2 (el único secuenciado en ella, por ahora). De igual manera, la fábrica F2 tendrá un C_2 de 39. Escogemos la fábrica F2 para secuenciar el trabajo J1.

Siguiendo la regla NEH, probamos el trabajo J1 en todas las posiciones posibles. Existen dos alternativas: secuenciarlo antes de J3, o después de J3.

La primera secuencia posible puede verse en la figura 4.8:

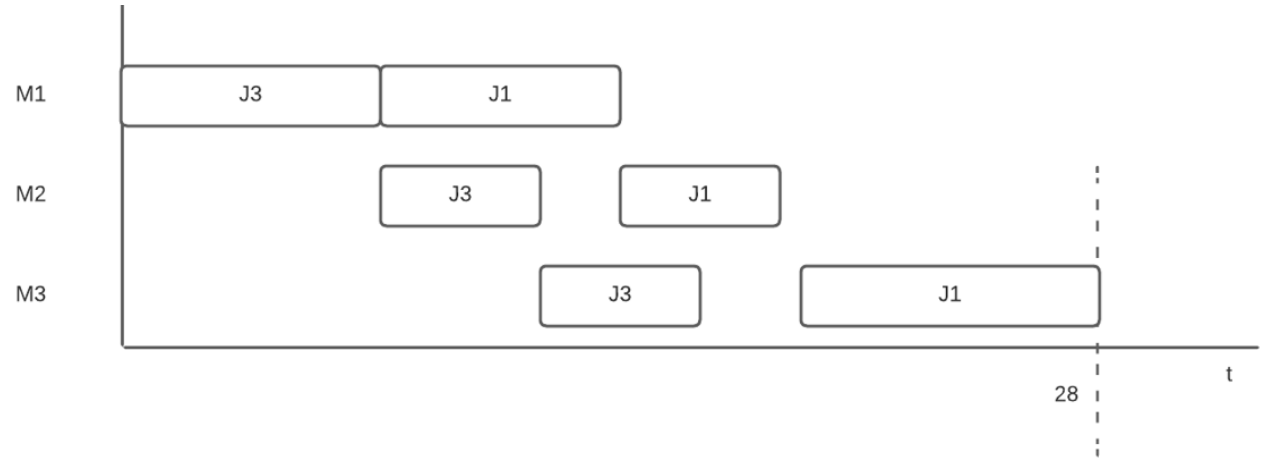

Figura 4.8: Primera secuencia posible

La segunda secuencia posible puede verse en la figura 4.9:

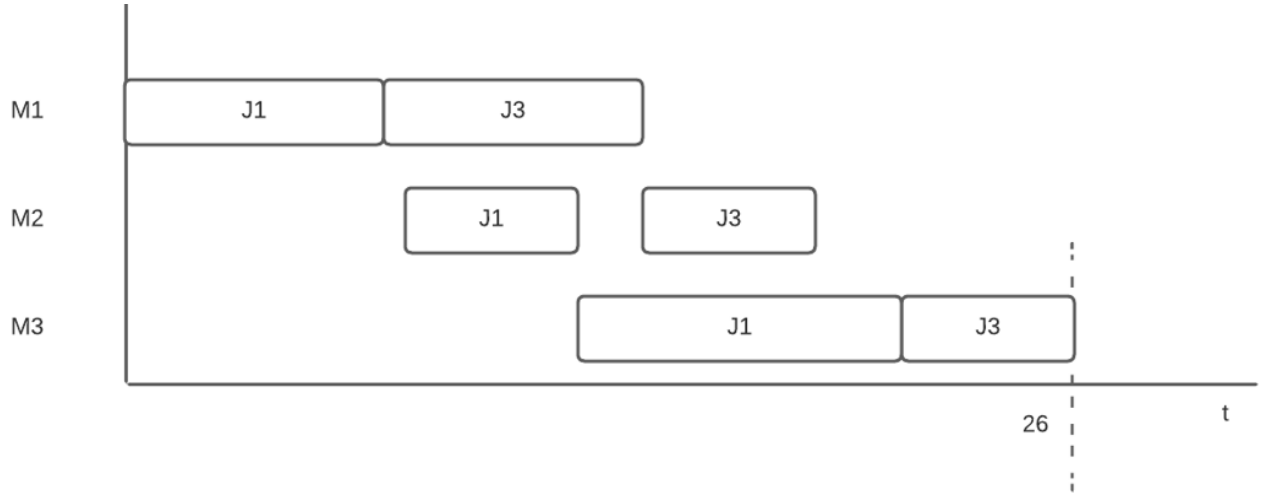

Figura 4.9: Segunda secuencia posible

Como la segunda secuencia tiene mejor tiempo de finalización, es escogida. El reparto final de trabajos es J2 en la fábrica F1 y J1-J3 en la fábrica F2.

Si se hubiera seguido la heuristica NEHb2 (aplicando la Regla no. 2), se hubiera probado el trabajo J1 en todas las posiciones de la fábrica F1 y la fábrica F2 (siguiendo la regla NEH). Después, se hubiera asignado finalmente a la fábrica con menor C_i, respetando la secuencia que ha obtenido el menor C_i.

4.1.3. Heurística DNEH

Las heurísticas NEHb1 y NEHb2 son la aplicación directa de NEH al problema distribuido. Por lo general, presentan un buen rendimiento. Sin embargo, específicamente en el DPFS heterogéneo, presenta un punto débil muy destacable precisamente debido a la naturaleza heterogénea del problema: la asignación de trabajos a fábricas se realiza según los C_i actuales, no según el propio tiempo de procesamiento p_j del propio trabajo en cada fábrica. De esta manera es muy posible que trabajos que tienen p_j reducidos en una fábrica, sean asignados arbitrariamente a otras con p_j más largos.

Para solucionar esto, se ha implementado la regla DNEH. Esta heurística proviene de Shao et al. (2023) y sus siglas significan *distributed NEH* (en inglés). Tiene en cuenta la heterogeneidad de los p_j para asignar el trabajo a la fábrica que en principio tardará menos en procesarlo. Después, se secuencia en cada fábrica siguiendo la regla NEH. El procedimiento para aplicarla es el siguiente:

Paso 1. Sumar los tiempos de procesamiento de los trabajos para cada fábrica mediante la fórmula $P_i = \sum p_{ij}$ con $j = 1, ..., N$ y $f = 1, ..., F$.

Paso 2. Ordenar los trabajos según su tiempo de procesamiento P_i total

Paso 3. Asignar el trabajo a la fábrica donde tenga menor P_i.

Paso 4. En cada fábrica, secuenciar los trabajos en el orden obtenido según la regla NEH.

Para este ejemplo, partimos de la misma instancia que en el ejemplo anterior. Es decir, la tabla de tiempos de proceso es la figura 4.7. En primer lugar se suman los tiempos de proceso en cada fábrica para cada trabajo.

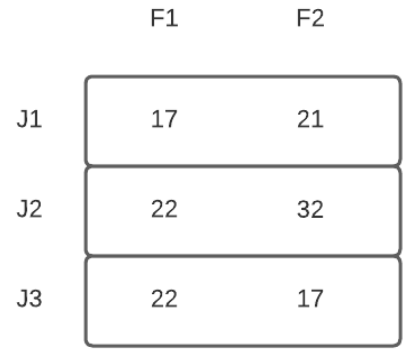

Figura 4.10: Suma de tiempos de procesamiento por fábrica

Después, se ordenan los trabajos según su tiempo de procesamiento total y se asignan a la fábrica donde tengan un menor tiempo de procesamiento (círculo en rojo en la figura 4.11)

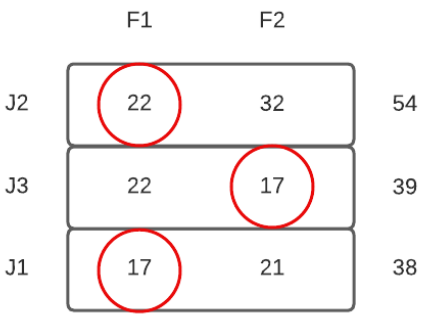

Figura 4.11: Asignación a la fábrica con menor tiempo de procesamiento

Por último, los trabajos se secuencian en cada fábrica según la regla NEH, en el orden en el que aparecen en la lista. En la fábrica F1 se tendrá que probar la secuencia J2-J1 y después la secuencia J1-J2, eligiendo la que tenga menor C_i. En la fábrica F2 sólo está el trabajo J3, por lo que no es necesario realizar ningún cálculo adicional.

4.1.4. Heurística DNEH2

La heurística DNEH presentada en el apartado anterior, incurre en un problema precisamente por su mayor cualidad, que es asignar cada trabajo a la fábrica donde tiene menor tiempo de procesamiento. Esta técnica provoca que la carga de trabajo entre las distintas fábricas no esté necesariamente equilibrada. Si, por ejemplo, de un total de 100 órdenes de fabricación similares, 70 tienen menor tiempo de procesamiento en la fábrica A (aunque sea por una sola unidad de tiempo), todas serán asignadas a la fábrica A. Esto provoca que la fábrica B, con 30 órdenes, esté infrautilizada, perjudicando a la función objetivo C_{max}.

La siguiente heurística, a la cual se ha decidido llamar DNEH2, ha sido creada por el autor para tratar de mejorar la regla DNEH. Asigna cada trabajo a la fábrica donde tiene un menor tiempo de proceso, a la vez que equilibra la carga de trabajo entre fábricas. Emplea un parámetro $alpha$ que designa el porcentaje de trabajos, con un p_j similar, que serán asignados a las fábricas con menor carga de trabajo. De esta manera, los trabajos más homogéneos, con menor impacto negativo si se secuencian en una fábrica cualquiera, son utilizados para equilibrar la carga de trabajo entre fábricas. Por ejemplo, un alpha de 0.2 indica que el 20 por ciento de los trabajos con este propósito. El procedimiento para aplicar esta heurística es el siguiente:

Paso 1. Sumar los tiempos de procesamiento de los trabajos para cada fábrica mediante la fórmula $P_i = \sum p_{ij}$ con $i = 1, ..., n$ y $j = 1, ..., f$.

Paso 2. Ordenar los trabajos según su tiempo de procesamiento P_i total, en orden decreciente.

Paso 3. Separar el α porcentaje de trabajos con p_j más similares. Esto se calcula como el porcentaje de trabajos con menor desviación típica de sus p_j entre fábricas. En otras palabras, se calcula la desviación típica para todos los trabajos (a partir de sus p_j), se ordena de manera descendente, y se escogen los n x alpha últimos. Por ejemplo, si alpha es de 0.2, y n es de 10, se escogerán los últimos dos trabajos.

Paso 4. Asignar los $n - (nx\alpha)$ trabajos según la heurística DNEH estándar. Esto es, en la fábrica con menor p_j total y siguiendo la regla NEH.

Paso 5. Asignar los $nxalpha$ trabajos de la lista α de trabajos similares secuencialmente a las fábricas con menor C_i, equilibrando la carga de trabajo.

A continuación se pondrá un nuevo ejemplo para ilustrar esta heurística. Se contarán con cinco trabajos, dos fábricas y tres máquinas por fábrica. El valor de alpha será de 0.2.

Los trabajos propuestos para este ejemplo son los siguientes. Ya se han realizado los dos primeros pasos: se han sumado los tiempos de procesamiento totales para cada fábrica y ordenado de manera descendente.

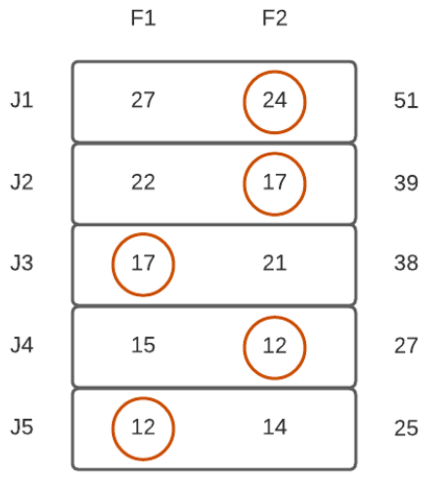

Figura 4.12: Trabajos para la heurística DNEH3

A continuación, se calculan las desviaciones típicas de cada trabajo y se ordenan de manera descendente. Por ejemplo, para el primer trabajo, la varianza es $\frac{(27-25{,}5)^2+(24-25{,}5)^2}{2} = 2{,}25$. Su desviación típica es $1{,}5$. En instancias más complejas, con más máquinas por fábrica, la desviación típica es calculada con un mayor número de datos. En la figura 4.13 se muestra la lista ordenada:

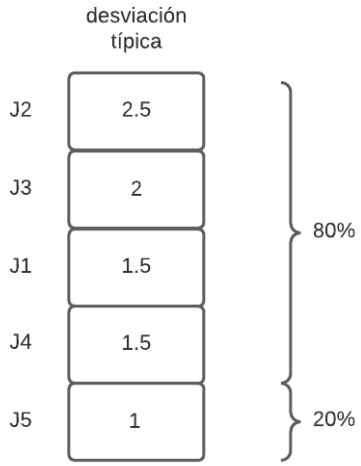

Figura 4.13: Trabajos ordenados según su desviación típica

El trabajo J5 resulta ser el que tiene una menor desviación típica, por lo que es escogido para conformar la lista $alpha$.

Se reserva dicho trabajo, y se secuencia el resto de trabajos según la regla DNEH estándar. El resultado puede comprobarse en las figuras 4.14 y 4.15. Para la fábrica F1 se obtiene un C_i de 62, y para la fábrica F2, un C_i de 17, claramente inferior.

Como la fábrica F2 es la que tiene menor C_i, el trabajo J5 es secuenciado en ella según la regla NEH. Si hubiera trabajos adicionales en la lista $alpha$, tendría que recalcularse el C_i de la fábrica F2, obtener la nueva fábrica con menor C_i, y secuenciarlo en ella.

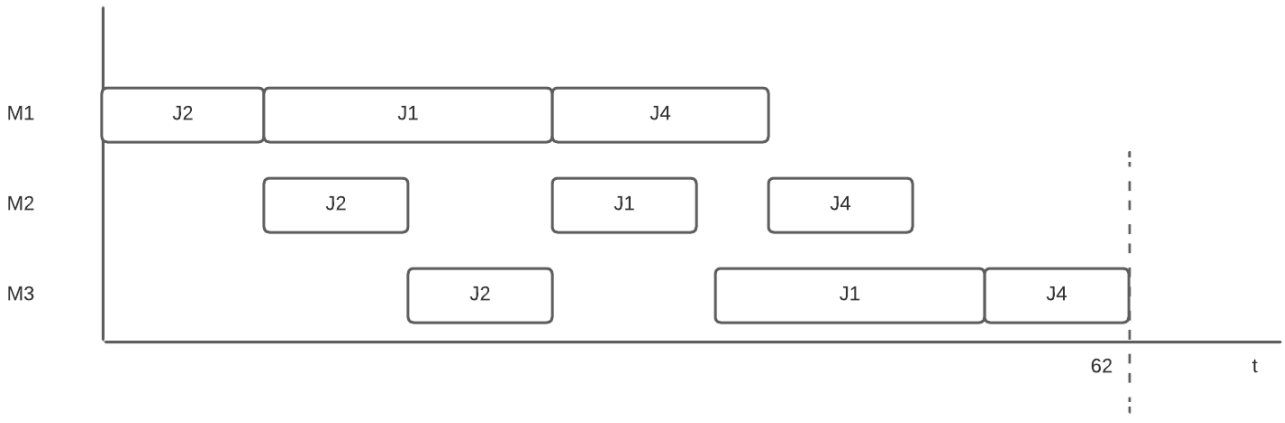

Figura 4.14: Secuencia en la fábrica F1

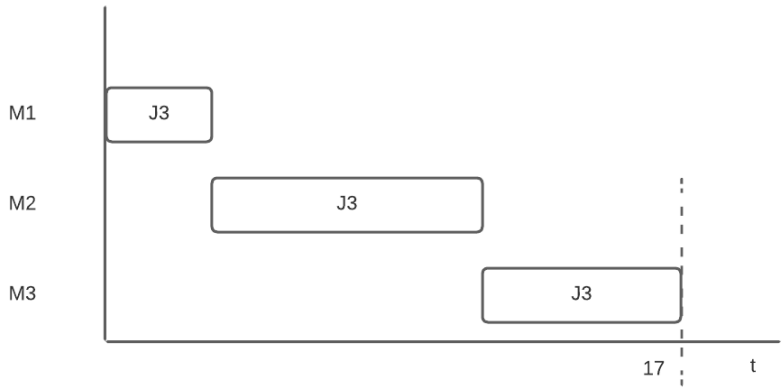

Figura 4.15: Secuencia en la fábrica F2

4.1.5. Heurística NEHb3

Esta heurística ha sido desarrollada por el autor con el objetivo de mejorar la regla NEHb2 presentada en apartados anteriores. Es similar, salvo que mediante la nueva regla se secuencian dos trabajos a la vez, generando dos soluciones y escogiendo la mejor de ellas. De esta manera, la heurística tiene en cuenta un espacio de soluciones posibles mayor. El procedimiento es el siguiente:

Paso 1. Sumar los tiempos de procesamiento de los trabajos para cada fábrica mediante la fórmula $P_i = \sum p_{ij}$ con $i = 1, ..., n$ y $j = 1, ..., f$.

Paso 2. Ordenar los trabajos según su suma de P_i total, en orden decreciente.

Paso 3. Escoger los dos primeros trabajos de la lista. Al primero se le llama arbitrariamente A, y al segundo B.

Paso 4. Generar la primera solución secuenciando primero el trabajo A y posteriormente el trabajo B según la regla NEHb2.

Paso 5. Generar la segunda solución secuenciando primero el trabajo B y posteriormente el trabajo A según la regla NEHb2.

Paso 6. Escoger la solución con menor C_{max}. Posteriormente, se sigue iterando hasta que todos los trabajos hayan sido secuenciados.

Sería posible extender este método, escogiendo los trabajos de tres en tres en vez de dos en dos y generando seis soluciones, para posteriormente escoger la mejor combinación. Esto se deja para trabajos posteriores.

4.2. Búsqueda local

Con el objetivo de mejorar las soluciones obtenidas a partir de las heurísticas, se ha desarrollado una búsqueda local. La característica clave de la búsqueda local es que no considera todo el espacio de búsqueda, sino que examina únicamente soluciones en la proximidad de la solución actual. Esto hace que este método sea particularmente útil para resolver problemas en los que el espacio de búsqueda es demasiado grande para explorar exhaustivamente.

Los algoritmos de búsqueda local se utilizan a menudo en problemas de secuenciacion de la producción. Generalmente son eficientes y pueden mejorar en gran medida las soluciones (como veremos en el capítulo de Resultados), aunque siguen sin garantizar el óptimo.

La mayoría de algoritmos de búsqueda local generan un vecindario, que consiste en el espacio de soluciones próxima a la solución actual. Una solución próxima es aquella a la que se llega realizando un movimiento definido. Por ejemplo, si tenemos la solución 1,2,3,4 y realizamos movimientos de permutación (un tipo de movimiento muy típico en estos problemas), el vecindario sería el conjunto 2,1,3,4, 1,3,2,4 y 1,2,4,3.

Existen dos estrategias. La primera de ellas es *"first improvement"*. En ella, sólo se calcula un vecino en vez de todo el vecindario. Si la nueva solución mejora la función objetivo, es escogida. Si no, se calcula el siguiente vecino, y se continúa el proceso. Este método es efectivo cuando es costoso computacionalmente calcular los vecinos o su función objetivo, ya que se hace de uno en uno.

En la segunda estrategia, llamada *"best improvement"*, se calcula el vecindario al completo y luego se elige la mejor solución. En este problema y con los tamaños de instancia utilizados, sí que es posible realizar este método, aunque sea más costoso. Sin embargo, converge mucho más rápido hacia el óptimo local.

Los pasos que sigue el método elegido son los siguientes:

Paso 1. Generar la solución inicial a partir de cualquier heurística. La búsqueda local tratará de mejorar esta solución.

Paso 2. Seleccionar un trabajo al azar de la fábrica con mayor C_i.

Paso 3. Eliminar el trabajo de su posición actual e insertarlo en todas las posiciones posibles. Esto significa probar el trabajo en todas las fábricas, tanto en primera como última posición, y todas las posiciones intermedias.

Paso 4. Se calcula el C_{max} de cada solución y se escoge la mejor. De esta manera, se sigue una filosofía *"best improvement"*.

Paso 5. Este proceso se repite hasta que la solución no mejora tres veces seguidas.

La lógica de esta búsqueda local es la siguiente: se trata de mejorar la función objetivo C_{max} seleccionando la fábrica con mayor C_i. Se escoge un trabajo al azar y se sitúa en la posición donde tenga un impacto menor. Es importante escogerlo al azar, ya que esto tiene dos ventajas. En primer lugar, es difícil saber a priori qué trabajo es el que empeora la función objetivo en mayor medida. En segundo lugar, si la solución no mejora, se puede volver a intentar este proceso con otro trabajo distinto.

En el siguiente ejemplo, se cuenta con diez trabajos, tres fábricas y tres máquinas por fábrica. La solución inicial (secuencia en cada fábrica) puede verse en la figura 4.16:

Fábrica	Secuencia	Ci	Cmax
F1	J2 - J3 - J7	60	62
F2	J5 - J1 - J9 - J10	62	
F3	J8 - J4 - J6	55	

Figura 4.16: Solución inicial para la búsqueda local

Se escoge un trabajo al azar entre la fábrica F2, que es la que tiene un mayor C_i. El trabajo escogido resulta ser el J10. Ahora, se prueba el trabajo J10 en todas las posiciones posibles de todas las fábricas, generando una solución por cada una de ellas. Obtenemos 12 nuevas soluciones posibles, una por posición.

En una primera solución ejemplo, se secuencia el trabajo J10 en la primera posición de la fábrica F1. El nuevo C_{max} es de 66, peor que antes. En la figura 4.17 se puede ver la secuencia en cada una de las tres fábricas, el C_i en cada fábrica y el C_{max}, que es el máximo entre los C_i y que representa el valor de la función objetivo.

Fábrica	Secuencia	Ci	Cmax
F1	J10 - J2 - J3 - J7	66	66
F2	J5 - J1 - J9	54	
F3	J8 - J4 - J6	55	

Figura 4.17: Ejemplo de una solución vecina

En una segunda solución ejemplo, se secuencia el trabajo J10 en la segunda posición de la fábrica F3. El nuevo C_{max} es de 60, dos puntos mejor que en la solución original. De hecho, resulta ser la mejor entre las 12 soluciones, por lo que resulta ser la seleccionada (figura 4.18).

Este proceso se repite hasta que la solución no mejora por tres veces consecutivas.

Fábrica	Secuencia	Ci	Cmax
F1	J2 - J3 - J7	60	60
F2	J5 - J1 - J9	54	
F3	J8 - J10 - J4 - J6	59	

Figura 4.18: Mejor solución en el vecindario

4.3. Software utilizado

Las heurísticas se han programado en Python. Este es un lenguaje de programación versátil y popular que se destaca por su sintaxis clara y legible. Ofrece una amplia gama de bibliotecas y frameworks, es multiplataforma y se integra fácilmente con otros lenguajes. Python cuenta con una comunidad activa y un sólido soporte, y es considerado un lenguaje fácil de aprender.

También se ha empleado NumPy, una biblioteca de Python ampliamente utilizada para realizar cálculos numéricos y operaciones eficientes. Ofrece un conjunto de funciones y herramientas que permiten manipular grandes conjuntos de datos y matrices. Algunas de las características clave de NumPy son su capacidad para realizar operaciones vectorizadas, su amplio conjunto de funciones matemáticas y su integración con otras bibliotecas de Python, como Pandas y Matplotlib. En resumen, NumPy es una biblioteca fundamental para el procesamiento numérico en Python, facilitando el trabajo con datos y operaciones matemáticas de manera eficiente.

La evaluación de resultados y el ANOVA se han realizado en R. Este es un lenguaje de programación y entorno de software utilizado ampliamente en la estadística y el análisis de datos. Es especialmente conocido por su capacidad para manipular, visualizar y analizar datos de manera eficiente. R proporciona una amplia gama de bibliotecas y paquetes especializados para diferentes áreas, lo que permite a los usuarios realizar análisis estadísticos avanzados, modelado predictivo y visualizaciones de datos. Además, R es un lenguaje de código abierto con una comunidad activa, lo que garantiza la disponibilidad de recursos, tutoriales y soporte. En resumen, R es una herramienta poderosa y versátil para el análisis estadístico y el procesamiento de datos.

Por último, se ha usado el software estadístico StatGraphics CENTURION para la elaboración de algunos gráficos ANOVA como el gráfico de interacciones o el de medias de un factor.

Capítulo 5

Análisis de resultados

En esta sección se analizan los resultados de las diversas heurísticas respecto al valor logrado de la función objetivo. Para ello, se usará análisis de la varianza (ANOVA en inglés). La aplicación de esta técnica en este trabajo se ha realizado siguiendo las pautas de Montgomery (2001).

Como medida de la eficacia de las heurísticas probadas, se emplea la desviación porcentual relativa (RPD) para realizar comparaciones. La RPD se calcula con la siguiente expresión:

$$RPD_i = \frac{SOL_i - BEST}{BEST} \cdot 100$$

donde se toma como $BEST$ la mejor solución encontrada entre las cinco heurísticas para una instancia concreta, y después se compara este valor contra las soluciones SOL_i del resto de heurísticas.

El análisis de varianza es una técnica estadística utilizada para comparar las medias de tres o más grupos independientes. Su objetivo es determinar si las diferencias observadas entre las medias de los grupos son estadísticamente significativas o si se deben a la variación aleatoria. El ANOVA se basa en el supuesto de que las observaciones en cada grupo siguen una distribución normal y tienen varianzas iguales y residuos independientes. Utiliza la varianza para medir la variabilidad entre los grupos y dentro de los grupos.

El ANOVA calcula dos tipos de varianza: la varianza entre los grupos y la varianza dentro de los grupos. Compara estas dos fuentes de variación para determinar si las diferencias entre las medias de los grupos son significativas. Para realizar un ANOVA, se utiliza un modelo estadístico que descompone la variación total en tres componentes: la variación entre los grupos, la variación dentro de los grupos y la variación residual, que es la variación aleatoria no explicada por el modelo.

Se ha escogido esta técnica para evaluar el rendimiento de cada heurística, descartando conclusiones que pudieran derivarse de causas coyunturales, como la generación aleatoria de instancias o procesos aleatorios dentro de cada método. En este trabajo se ha realizado un ANOVA multifactor con cuatro factores: factor cualitativo tipo de heurística (NEHb, NEHb2, NEHb3, DNEH y DNEH2) y tres factores cuantitativos, número de fábricas, número de máquinas y número de trabajos. La variable respuesta es la RPD.

Se ha hecho tanto un análisis global como por parejas. El análisis global resultará significativo (hipótesis nula rechazada) si cualquiera de las heurísticas se desempeña de manera significativamente distinta al resto. El análisis por parejas mide, para cada caso, si una heurística es significativamente distinta de otra. Con este fin se han empleado intervalos de Tukey.

5.1. Generación de instancias

Uno de los sets de instancias más empleados en la DPFS es la Testbed TB1, propuesta por Naderi y Ruiz (2010). Está formado por dos subsets, uno de tamaño pequeño y otro de tamaño grande, basado en instancias clásicas para el PFSP. Está disponible actualmente en http://soa.iti.es. El set de instancias pequeño tiene un tamaño de n = {4, 6, 8, 10, 12, 14, 16}, m = {2, 3, 4, 5} y F = {2, 3, 4} para un total de 84 instancias. Por otro lado, el set de instancias grande tiene un tamaño de n = {20, 50, 100}, m = {5, 10, 20} y F = {2, 3, 4, 5, 6, 7} para un total de 54 instancias

Por otro lado, Meng y Pan (2021) utilizan otro set de instancias para el problema heterogéneo. Igual que en el caso anterior, hay dos subsets, uno pequeño y otro grande. El pequeño tiene n = {4, 5, 6, 7, 8}, m = {2, 3} y f = {2, 3, 4}. El grande tiene n = {20, 40, 60, 80, 100}, m = {2, 3} y f = {2, 4, 6}. Los tiempos de proceso *en cada máquina* se distribuyen uniformemente en el rango (5, 20).

Shao, Shao y Pi (2023) emplean el mismo set que Shao et al. (2021). Este es un set de 20 instancias con n = {20, 40, 60, 80, 100}, m = {5, 10} y f = {2, 3}. El tiempo de proceso es uniformemente distribuido en el rango (10,50).

Para este trabajo, el autor ha decidido utilizar la Testbed TB1, adaptado al caso heterogéneo. El principal motivo para tomar esta decisión ha sido el hecho de que la Testbed TB1 está disponible para el público, al contrario que el otro set de instancias. Además, este conjunto de datos ha sido empleado típicamente por los investigadores desde su generación en 2010. En el capítulo Resultados se explica en profundidad tanto las instancias empleadas como el método para generarlas.

Se han generado tres grupos de instancias: un conjunto de instancias pequeñas, otro de instancias grandes y por último, otro de instancias ”homogéneas”basado en el set de instancias pequeño. Este último grupo ha sido generado para comprobar el rendimiento de las distintas heurísticas según la heterogeneidad del problema, tal y como se ha comentado anteriormente. Su estructura es idéntica al del set de instancias pequeño, excepto por los propios tiempos de proceso de cada trabajo.

El número de instancias, de trabajos, fábricas y máquinas por instancias, y las distribuciones para generar los tiempos de proceso está basado en la Testbed TB1. Sin embargo, para adaptarlo al DPFS heterogéneo, cada fábrica tendrá una matriz de tiempos de procesamiento distinta. Se añade un nuevo subíndice (p_{ij} pasa a ser p_{ijf}, tiempo de procesamiento del trabajo j en la fábrica f).

De esta manera, cada trabajo tiene un tiempo de procesamiento distinto en cada fábrica. Por ejemplo, la primera posición de la primera matriz (posición con índices i = 1, j = 1, en la matriz f = 1) representa el tiempo de procesamiento del primer trabajo en la primera máquina de la fábrica uno. Este puede ser distinto para la segunda fábrica (indices i = 1, j = 1 y f = 2).

La estructura para instancias pequeñas es la siguiente:

- $n \in \{4, 6, 8, 10, 12, 14, 16\}$
- $m \in \{2, 3, 4, 5\}$
- $f \in \{2, 3, 4\}$
- $p_{ij} = U(1, 99)$

Para un total de 84 instancias individuales.

Y para instancias grandes:

- $n \in \{20, 50, 100\}$
- $m \in \{5, 10, 20\}$

- $f \in \{2, 3, 4, 5, 6, 7\}$
- $p_{ij} = U(1, 99)$

Para un total de 54 instancias individuales.

El conjunto ”heterogéneo”sigue la misma distribución que el set de instancias pequeño excepto en la distribución de los tiempos de proceso p_{ij}. En primer lugar, para cada trabajo se ha generado un p_j central. Después se han generado los p_{ij} para cada fábrica variando un máximo de $+-10\,\%$ este valor. De esta manera, cada trabajo tiene un tiempo de procesamiento similar en cada fábrica, variando un máximo de un 20 por ciento de una fábrica a otra.

5.2. Hipótesis para el ANOVA

Los tres principales requisitos que debe cumplir un análisis de varianza (ANOVA) para considerarse válido son los siguientes:

- Independencia de observaciones: Las observaciones en cada grupo deben ser independientes entre sí. Esto significa que los valores en un grupo no deben estar relacionados o influenciados por los valores en otros grupos. La independencia se puede lograr mediante un diseño de experimentos adecuado.
- Normalidad: Los residuos deben seguir una distribución normal. La normalidad se puede verificar utilizando pruebas estadísticas, como la prueba de normalidad de Shapiro-Wilk, o mediante métodos gráficos. Si los datos no siguen una distribución normal, se pueden aplicar transformaciones o considerar métodos alternativos.
- Homogeneidad de varianzas: Las varianzas de los grupos deben ser iguales o muy similares. Esto se conoce como homogeneidad de varianzas. La homogeneidad de varianzas se puede evaluar utilizando pruebas estadísticas, como la prueba de Levene o la prueba de Bartlett. Si los grupos tienen varianzas desiguales, los resultados del ANOVA pueden verse afectados y se pueden aplicar ajustes o métodos alternativos, como el ANOVA con corrección de Welch. Este requisito también es conocido como ”homocedasticidad”.

Estos tres requisitos son fundamentales para garantizar la validez de los resultados del ANOVA y obtener conclusiones confiables sobre las diferencias significativas entre las medias de los grupos. Es importante evaluar y cumplir estos requisitos antes de realizar un ANOVA y, en caso de no cumplirse, considerar alternativas o técnicas estadísticas adecuadas para el análisis de los datos.

Para verificar la hipótesis de normalidad, se han analizado los residuos en el papel probabilístico normal. Este gráfico se muestra en la figura 5.1. Podemos comprobar que la mayoría de residuos se organizan en torno a la diagonal, señalando que nos encontramos ante una distribución normal. Ocurre algo de deformación en los extremos, pero esta no invalida los resultados del análisis.

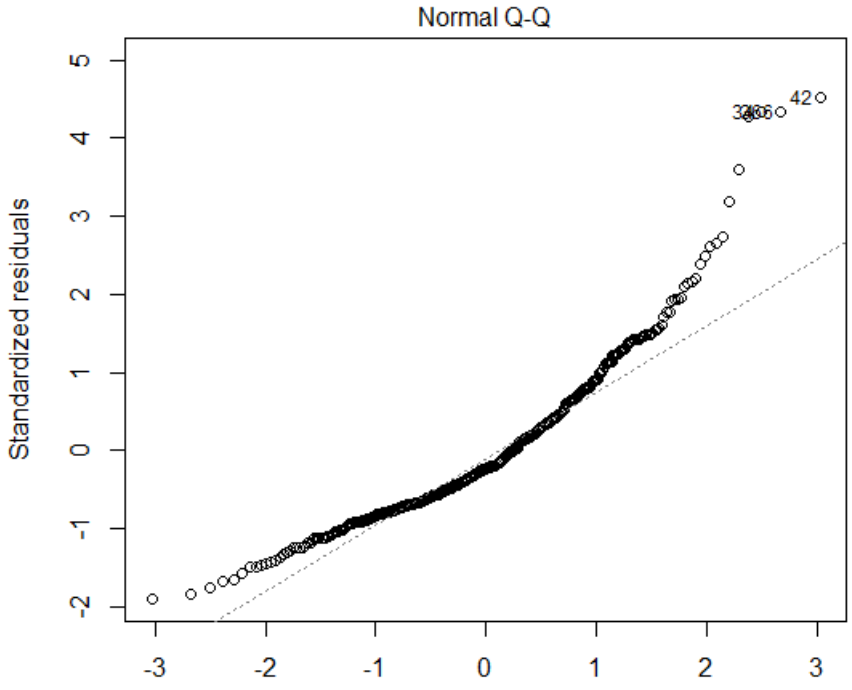

Figura 5.1: Residuos en papel probabilístico normal

Respecto a la homocedasticidad e independencia de los residuos, disponemos del gráfico de residuos contra observaciones (figura 5.2).

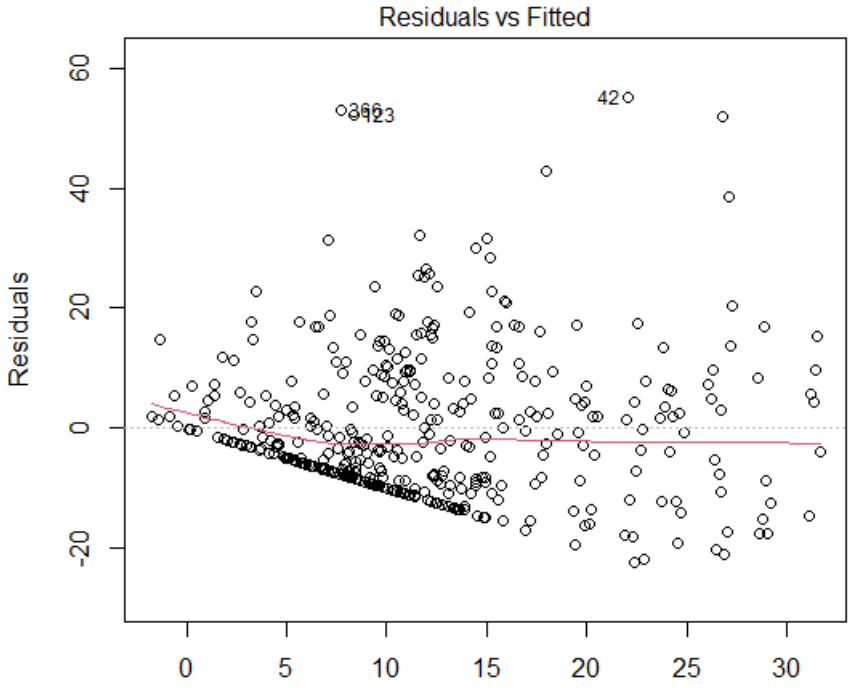

Figura 5.2: Gráfico de residuos contra observaciones

Podemos observar que el único patrón reconocible es el área sin puntos que va desde el (0,0) hasta el (-20,20). Por lo tanto, se puede asegurar que existe tanto independencia de residuos como homocedasticidad.

En realidad, la independencia de los residuos (y las observaciones) se da por garantizado, ya que las instancias y heurísticas no guardan relación entre sí. Los datos, al ser generados aleatoriamente y provenir de un experimento computacional, no pueden verse contaminados por fuentes de variabilidad común no explicada.

En conclusión, podemos dar por válidas las hipótesis del ANOVA. El análisis realizado resulta adecuado a todos los efectos, permitiéndonos extraer las debidas conclusiones.

5.3. Heurísticas en instancias pequeñas

Se ha calculado el valor de la función objetivo logrado por cada heurística (NEHb1, NEHb2, NEHb doble, DNEH y DNEH2) en cada subinstancia, y se ha pasado a R para el cálculo del ANOVA.

En la siguiente tabla se pueden ver los RPD obtenidos. La columna N muestra el número de trabajos, la columna M el número de máquinas por fábrica y la columna F el número de fábricas para esa instancia determinada. No existen instancias duplicadas (con los mismos parámetros). Por ejemplo, en esta tabla el primer experimento se ha realizado para 10 trabajos, dos fábricas y dos máquinas en cada fábrica. El resto de columnas (cada una con el nombre de su heurística) muestra el RPD obtenido.

Tabla 5.1: RPD para instancias pequeñas

N	M	F	NEHb	NEHb2	DNEH	DNEH2	NEHb3
10	2	2	18.98	2.37	15.93	0.00	0.34
12	2	2	22.62	0.00	1.64	2.62	0.00
14	2	2	1.10	7.42	17.58	1.65	0.00
16	2	2	30.69	0.00	3.71	18.56	0.25
4	2	2	0.00	0.00	43.87	0.00	0.00
6	2	2	15.27	0.00	0.00	0.00	0.00
8	2	2	40.00	3.27	4.90	0.00	3.27
10	3	2	6.76	2.06	21.47	0.00	2.06
12	3	2	22.35	2.58	44.41	11.75	0.00
14	3	2	15.82	5.71	0.44	5.93	0.00
16	3	2	22.36	4.81	19.95	0.00	13.46
4	3	2	24.26	24.26	16.83	0.00	24.26
6	3	2	27.11	1.06	17.96	0.00	1.06
8	3	2	4.15	0.00	27.48	4.15	0.00
10	4	2	9.43	6.95	5.71	0.00	1.74
12	4	2	13.23	0.00	16.79	12.47	0.00
14	4	2	15.40	0.00	2.23	7.37	3.57
16	4	2	20.53	0.00	5.84	5.84	0.94
4	4	2	8.06	0.00	6.45	6.45	0.00
6	4	2	19.52	0.00	4.11	23.63	0.00
8	4	2	33.76	25.80	4.78	0.00	23.25
10	5	2	28.96	7.92	0.00	8.82	7.92
12	5	2	17.89	8.13	7.11	0.00	7.52
14	5	2	32.40	8.60	0.00	3.20	8.60
16	5	2	3.73	2.49	23.33	23.33	0.00
4	5	2	6.73	6.73	26.30	0.00	6.73
6	5	2	23.42	6.05	0.00	13.16	6.05
8	5	2	10.42	2.00	0.00	7.54	1.55
10	2	3	9.66	0.00	28.98	28.98	0.00
12	2	3	65.67	0.00	16.42	16.42	0.00
14	2	3	40.98	18.05	27.80	0.00	20.98
16	2	3	47.76	0.00	10.82	13.81	6.72
4	2	3	0.00	0.00	1.69	1.69	0.00
6	2	3	78.70	0.00	0.00	37.96	0.00
8	2	3	5.81	0.00	33.55	27.74	0.00
10	3	3	5.33	6.15	4.92	13.93	0.00
12	3	3	27.17	0.00	3.15	23.62	0.00

Tabla 5.1: RPD para instancias pequeñas

14	3	3	10.53	4.95	17.96	0.00	4.02
16	3	3	24.21	13.68	0.00	20.70	13.68
6	3	3	26.34	0.00	6.99	16.67	0.00
8	3	3	12.18	0.84	0.00	5.88	0.84
10	4	3	77.23	60.71	8.93	0.00	60.71
12	4	3	10.29	0.00	0.29	5.88	0.00
14	4	3	4.25	0.00	5.67	5.67	0.00
16	4	3	26.60	0.00	19.18	18.67	0.00
6	4	3	3.95	1.32	0.00	18.86	1.32
8	4	3	23.33	23.33	0.00	32.92	23.33
10	5	3	18.92	0.00	20.27	3.51	0.00
12	5	3	23.46	18.48	24.41	0.00	7.35
14	5	3	16.93	0.00	20.37	8.01	1.37
16	5	3	24.24	9.32	27.04	7.46	0.00
4	5	3	21.99	21.99	0.00	21.99	21.99
6	5	3	24.36	9.40	6.41	0.00	9.40
8	5	3	36.73	20.41	0.00	19.05	20.41
10	2	4	35.71	0.00	38.10	13.49	0.00
12	2	4	41.21	0.00	48.48	48.48	0.00
14	2	4	46.79	0.00	60.90	43.59	0.00
16	2	4	27.59	1.97	0.00	4.43	1.97
6	2	4	60.23	60.23	0.00	0.00	60.23
8	2	4	37.04	37.04	0.74	0.00	37.04
10	3	4	45.76	29.38	36.16	0.00	29.38
12	3	4	20.15	5.97	0.37	0.00	6.34
14	3	4	11.58	2.81	25.96	3.51	0.00
16	3	4	16.73	0.00	17.93	17.93	0.00
4	3	4	89.76	89.76	0.00	38.58	89.76
6	3	4	18.55	18.55	2.26	0.00	18.55
8	3	4	13.73	4.29	0.00	0.00	4.29
10	4	4	21.05	18.60	0.00	11.23	18.60
12	4	4	6.33	3.16	0.00	0.00	3.16
14	4	4	18.89	1.86	3.72	27.24	0.00
16	4	4	16.04	0.00	21.70	4.72	0.00
4	4	4	12.33	12.33	0.00	12.33	12.33
6	4	4	39.55	6.36	0.00	10.45	6.36
8	4	4	36.09	3.48	1.74	0.00	3.48
10	5	4	27.48	13.58	0.00	14.90	13.58
12	5	4	30.54	14.97	0.90	0.00	14.97
14	5	4	20.22	3.37	14.89	2.25	0.00
16	5	4	30.40	0.28	20.74	20.74	0.00
4	5	4	25.36	25.36	0.00	0.00	25.36
6	5	4	30.00	23.75	0.00	60.83	23.75
8	5	4	37.31	11.04	0.60	0.00	11.04
10	3	2	24.42	8.95	11.34	10.38	8.46
Media			23.43	7.96	11.9	11.15	7.47

El ANOVA nos indica que sí que existen diferencias significativas entre los distintos algoritmos, ya que el factor **Algoritmo** es significativo. Se ha elegido un nivel de confianza del 5 %, con lo que si un p-valor es

inferior a 0.05, el factor es significativo.

Tabla 5.2: ANOVA para instancias pequeñas

	gl	SC	MC	F-valor	p-valor	Significativo
Algoritmo	4	13732	3433	22.68	2e-16	Sí
N	1	202	202	1.33	0.24880	No
M	1	548	548	3.62	0.05777	No
F	1	1828	1828	12.01	0.00057	Sí
Algoritmo:N	4	2350	588	3.88	0.00419	Sí
Algoritmo:M	4	1090	273	1.8	0.12795	No
Algoritmo:F	4	640	160	1.05	0.37717	No
Residuos	385	58272				

Para seleccionar el mejor, emplearemos un gráfico (figura 5.3) con intervalos HSD de Tukey. En él se comparan todas las heurísticas a la vez. Si los intervalos se solapan, significa que no existen diferencias significativas.

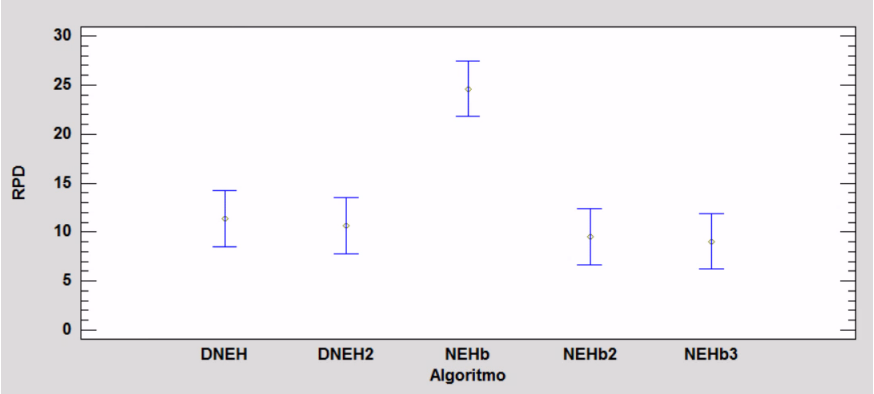

Figura 5.3: Gráfico de intervalos de Tukey para instancias pequeñas

Se puede concluir que, para instancias pequeñas, la peor heurística es NEHb, siendo el resto igual de buenas.

Como puede verse en el resumen del ANOVA (tabla 5.2), el factor número de fábricas **F** también resulta significativo. En el gráfico de medias (figura 5.4) observamos que, cuanto mayor es el número de fábricas, más aumenta la RPD (empeora la eficacia de los algoritmos).

Figura 5.4: Gráfico de medias para el factor F

Aunque el número de trabajos N no resulta significativo, su interacción con cada algoritmo sí que lo es (factor **Algoritmo:N**). La figura 5.5 es el gráfico de interacción para el factor. Se puede observar que algunos algoritmos mejoran su eficacia a medida que aumenta N (NEHb2 y NEHb3), otros la empeoran (DNEH) y otros no se ven afectados (DNEH2 y NEHb).

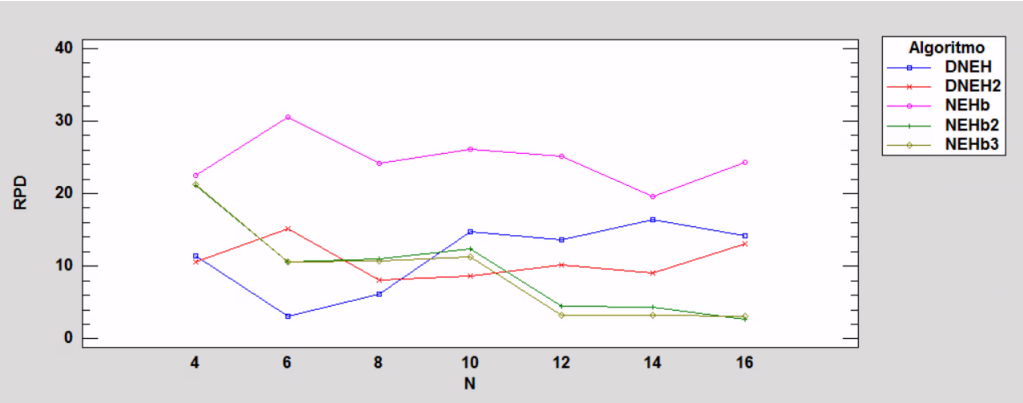

Figura 5.5: Gráfico de interacción para el factor Algoritmo:N

5.4. Heurísticas en instancias grandes

La tabla con los RPD obtenida para el set de instancias grande es la siguiente:

Tabla 5.3: RPD para instancias grandes

N	M	F	NEHb	NEHb2	DNEH	DNEH2	NEHb3
100	10	2	7.58	3.93	5.09	0.00	1.72
20	10	2	12.49	3.60	6.69	5.00	0.00
50	10	2	7.85	0.06	0.00	5.31	1.02
100	20	2	6.91	5.34	5.99	0.00	0.92
20	20	2	7.61	0.98	13.07	0.00	1.78
50	20	2	4.25	0.00	3.54	6.32	0.83
100	5	2	15.69	1.98	5.28	0.00	2.14
20	5	2	22.65	0.80	6.86	0.00	0.80
50	5	2	18.18	4.58	6.09	0.00	1.50
100	10	3	12.76	1.97	1.25	0.00	0.87
20	10	3	9.39	2.02	13.20	1.31	0.00
50	10	3	12.38	1.17	0.23	1.71	0.00
100	20	3	6.96	0.66	4.16	0.00	2.04
20	20	3	20.85	1.84	4.89	12.55	0.00
50	20	3	4.47	2.36	11.99	0.55	0.00
100	5	3	30.78	4.73	5.56	1.85	0.00
20	5	3	32.92	0.00	36.25	27.08	3.54
50	5	3	12.58	0.00	15.25	6.08	1.60
100	10	4	16.78	0.67	18.35	0.73	0.00
20	10	4	16.95	0.00	23.25	12.61	0.00
50	10	4	20.20	0.00	12.08	2.08	0.30
100	20	4	9.77	1.01	2.64	1.76	0.00
20	20	4	14.57	0.00	2.87	5.18	0.00
50	20	4	6.50	0.00	14.65	4.16	1.71
100	5	4	25.45	0.00	15.85	16.45	3.68
20	5	4	27.49	0.00	3.16	2.19	2.19
50	5	4	21.77	0.00	28.23	8.44	1.32
100	10	5	13.65	0.00	7.65	1.50	0.50
20	10	5	27.11	11.49	0.00	2.60	11.49
50	10	5	18.13	0.00	2.28	6.19	1.19
100	20	5	12.83	0.00	8.14	4.50	1.55
20	20	5	18.21	6.97	8.78	0.00	6.97
50	20	5	15.37	0.00	7.13	3.18	3.05
100	5	5	34.99	4.68	39.67	9.46	0.00
20	5	5	26.80	3.23	0.00	0.00	3.23
50	5	5	28.50	5.17	16.83	3.67	0.00
100	10	6	15.29	0.00	15.29	10.30	1.19
20	10	6	18.09	8.36	8.97	0.00	4.26
50	10	6	17.38	0.00	20.97	6.03	0.00
100	20	6	12.11	0.58	3.53	0.68	0.00
20	20	6	14.88	0.00	4.01	1.67	0.00
50	20	6	11.95	0.00	3.35	1.57	0.00
100	5	6	36.12	0.00	10.39	7.60	2.15
20	5	6	18.04	2.58	4.38	0.00	2.58

Tabla 5.3: RPD para instancias grandes

50	5	6	36.28	0.00	6.91	2.69	0.00
100	10	7	22.39	1.00	9.34	8.25	0.00
20	10	7	18.00	0.00	8.84	6.06	0.00
50	10	7	23.74	0.00	15.40	9.34	1.14
100	20	7	10.35	0.00	13.00	7.20	0.00
20	20	7	16.20	0.00	11.35	6.33	0.00
50	20	7	11.82	1.51	0.00	12.54	1.15
100	5	7	35.37	4.47	39.30	10.70	0.00
20	5	7	17.30	0.00	5.87	3.81	0.00
50	5	7	34.33	0.00	55.58	22.10	1.93
56	11	4	17.98	1.62	10.99	4.99	1.30
Media			17.98	1.62	10.99	4.99	1.30

En este caso, el ANOVA también indica que sí que existen diferencias significativas. De nuevo, el nivel de confianza es del 5 %. El cuadro resumen puede verse en la tabla 5.4:

Tabla 5.4: ANOVA para instancias grandes

	gl	SC	MC	F-valor	p-valor	Significativo
Algoritmo	4	10930	2732.5	73	2e-16	Sí
N	1	3	3	0.09	0.76642	No
M	1	1175	1175	31.37	5.61e-8	Sí
F	1	541	541	14.44	0.00018	Sí
Algoritmo:N	4	68	17	0.45	0.76874	Sí
Algoritmo:M	4	1271	318	8.49	1.94e-6	Sí
Algoritmo:F	4	481	120.4	3.21	0.01347	Sí
Residuos	250	9361	37.4			

El gráfico de intervalos HSD de Tukey aparece en la figura 5.4. Como indica el ANOVA, existen algoritmos con un rendimiento significativamente superior a otros:

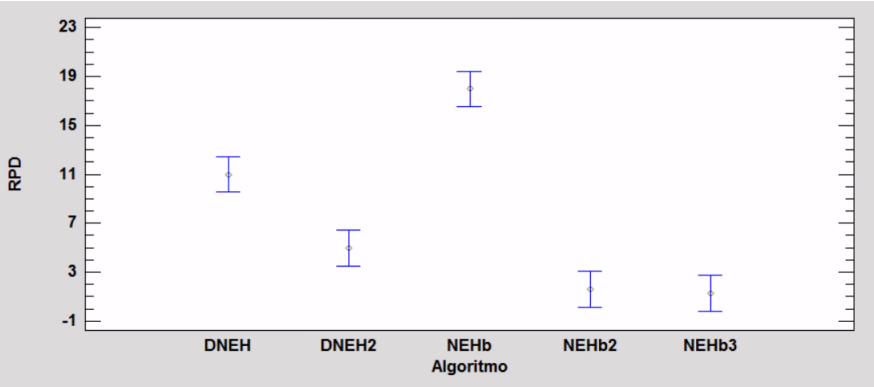

Figura 5.6: Gráfico de intervalos de Tukey para instancias grandes

Para heurísticas grandes, el peor algoritmo es NEHb, seguido de DNEH y DNEH2, con una diferencia significativa entre cada uno de ellos. Los mejores son NEHb2 y NEHb3, sin diferencias significativas.

Como puede verse en en resumen del ANOVA (tabla 5.4), el factor número de trabajos **N** sigue sin resultar significativo. El número de máquinas **M** pasa a ser significativo (a diferencia de en instancias pequeñas). En la

figura 5.7 puede apreciarse que, cuanto mayor es el número de máquinas, más homogénea es la calidad de las soluciones aportadas por los algoritmos (disminuye el RPD).

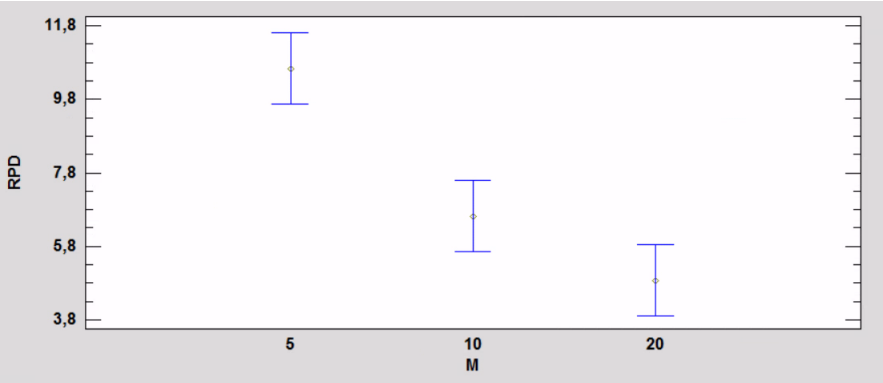

Figura 5.7: Gráfico de medias para el factor M

Respecto a las interacciones, todas resultan ser significativas. La figura 5.8 muestra el gráfico de interacciones para el factor **Algoritmo:M**. Puede comprobarse que, para los algoritmos NEHb, DNEH y DNEH2, su RPD mejora cuanto mayor es el número de máquinas. Los algoritmos NEHb2 y NEHb3 no presentan esta relación.

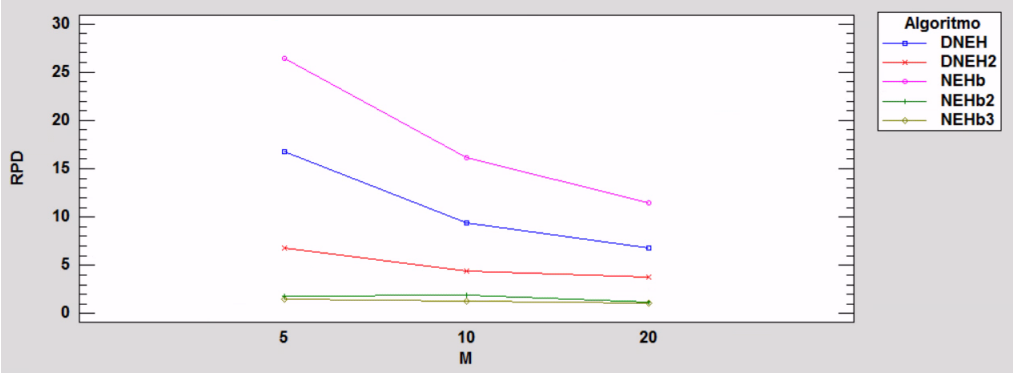

Figura 5.8: Gráfico de interacción para el factor Algoritmo:M

El factor **Algoritmo:F** también resulta significativo. En la figura 5.9 puede verse cómo NEHb, DNEH y DNEH2 empeoran su eficiencia a medida que aumenta el número de fábricas, mientras que NEHb2 y NEHb3 permanecen invariables.

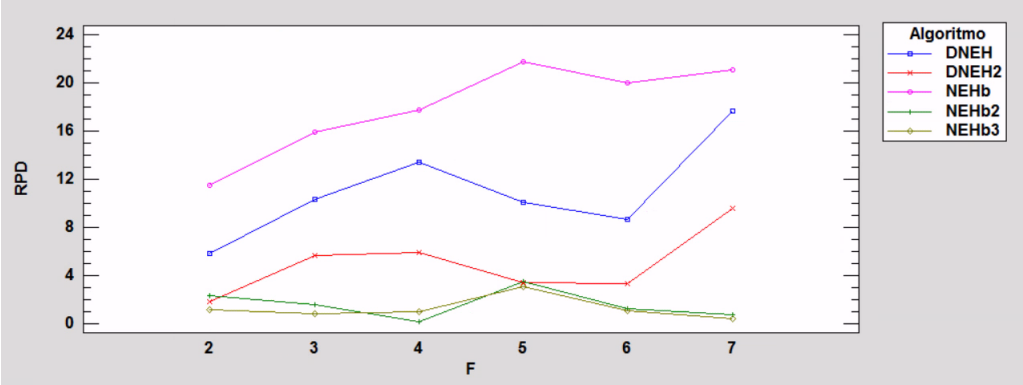

Figura 5.9: Gráfico de interacción para el factor Algoritmo:F

5.5. Búsqueda local para instancias pequeñas

La siguiente tabla muestra los RPD de las heurísticas originales, y tras haberles aplicado la búsqueda local. Es importante tener en cuenta que los RPD de las heurísticas originales cambian, ya que las búsquedas locales mejoran las soluciones, modificando el valor de OPT de la fórmula.

A diferencia de las tablas anteriores, sólo se muestran los valores medios para economizar espacio. Las heurísticas con el sufijo ls indica que se ha aplicado una búsqueda local posterior.

Tabla 5.5: RPD para instancias pequeñas, heurísticas sin búsqueda local

	NEHb	NEHb2	DNEH	DNEH2	NEHb3
Media	30.31	14.19	17.10	14.42	13.66

Tabla 5.6: RPD para instancias pequeñas con búsqueda local

	NEHb_ls	NEHb2_ls	DNEH_ls	DNEH2_ls	NEHb3_ls
Media	22.66	13.30	4.16	3.01	12.58

Se observa que los valores de RPD aumentan para todas las heurísticas sin búsqueda local. De hecho, la búsqueda local mejora cada una de las heurísticas de la siguiente manera:

Tabla 5.7: Mejora al aplicar búsqueda local en instancias pequeñas

Heurística	Mejora media	Desviación típica
NEHb	7.97	10.32
NEHb2	0.71	2.05
DNEH	12.92	12.5
DNEH2	11.89	15.1
NEHb3	0.62	1.62

Los valores de esta tabla están expresados en términos porcentuales. De esta manera, una mejora media del 12.92 significa que, de media, la búsqueda local mejora el parámetro C_{max} un 12.92 por ciento respecto de la solución lograda originalmente.

Las heurísticas que más se benefician de la búsqueda local son DNEH y DNEH2, mientras que NEHb2 y NEHb3 apenas se ven afectadas. Esto sucede porque la búsqueda local, por su naturaleza, tiende a equilibrar la carga entre fábricas. Esta cualidad compensa el principal defecto de DNEH y DNEH2, que es una mayor eficiencia para cada orden a cambio de no realizar un reparto equitativo de la carga de trabajo.

De nuevo, empleamos intervalos de Tukey para determinar diferencias significativas (figura 5.5). En esta ocasión, se han hecho comparaciones por parejas. Si el intervalo contiene el 0, no hay diferencias significativas, mientras que cuanto más esté desplazado hacia un extremo, más diferencia media existe.

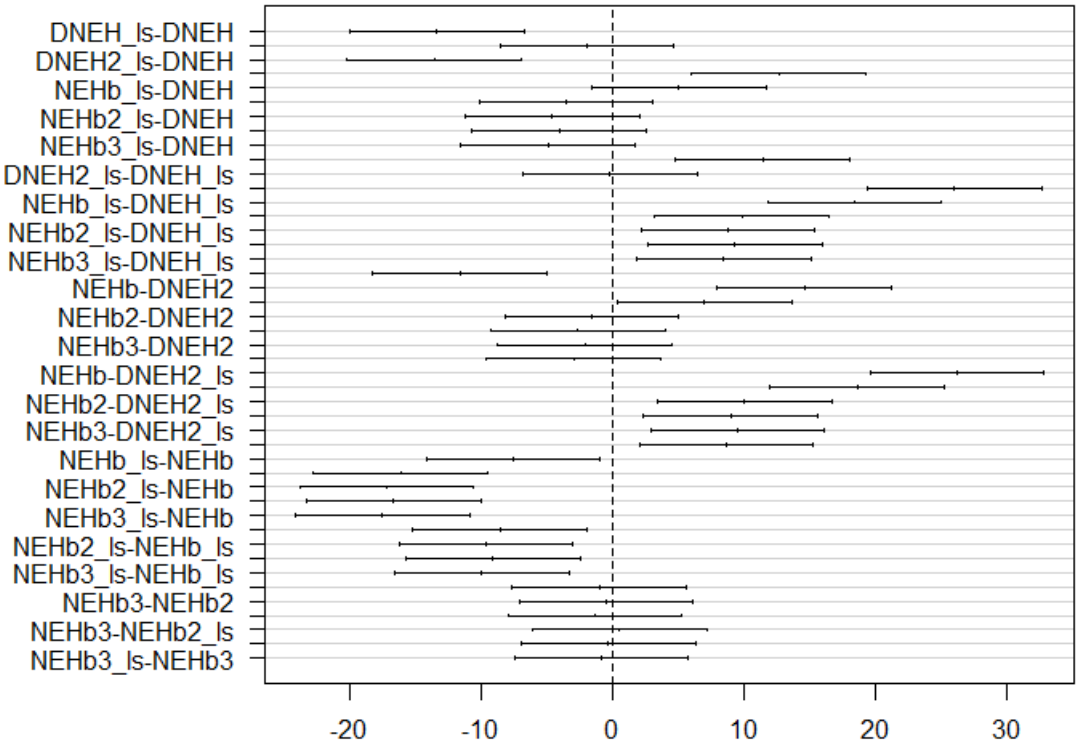

Figura 5.10: Gráfico de intervalos de Tukey para instancias pequeñas con búsqueda local

La diferencia que produce la búsqueda local puede verse de manera gráfica en la figura 5.11.

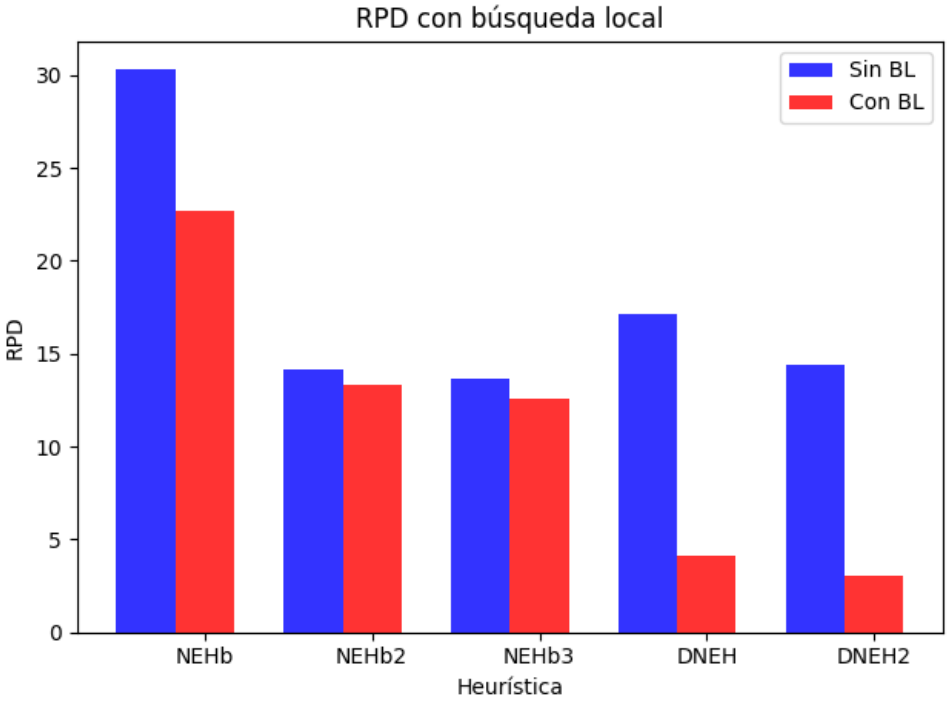

Figura 5.11: RPD media con y sin búsqueda local

A la vista de estos resultados, se puede concluir que la mejor heurística para resolver el problema de la DPFS heterogénea en instancias pequeñas es la heurística DNEH2 con búsqueda local posterior. Este método

no presenta diferencias significativas con DNEH con búsqueda local, que también resulta válido. El peor método es la heurística NEHb sin búsqueda local, por un amplio margen.

5.6. Búsqueda local para instancias grandes

Se ha realizado el mismo proceso para el set de instancias grande. La tabla RPD en este caso es la siguiente:

Tabla 5.8: RPD para instancias grandes, heurísticas sin búsqueda local

	NEHb	NEHb2	DNEH	DNEH2	NEHb3
Media 21.94	5.03	14.52	8.43	4.70	

Tabla 5.9: RPD para instancias grandes con búsqueda local

	NEHb_ls	NEHb2_ls	DNEH_ls	DNEH2_ls	NEHb3_ls
Media 17.74	4.56	2.70	3.56	4.12	

La búsqueda local mejora cada una de las heurísticas de la siguiente manera:

Tabla 5.10: Mejora al aplicar búsqueda local en instancias grandes

Heurística	Mejora media	Desviación típica
NEHb	3.97	5.02
NEHb2	0.62	1.01
DNEH	10.37	8.39
DNEH2	4.91	7.02
NEHb3	0.43	0.65

De nuevo, aplicar una búsqueda local mejora los valores de todas las heurísticas. La heurística DNEH es la principal beneficiada, y la siguen las heurísticas NEHb y DNEH2. Las heurísticas NEHb2 y NEHb3 apenas experimentan una mejoría.

El gráfico de intervalos de Tukey se muestra en la figura 5.6:.

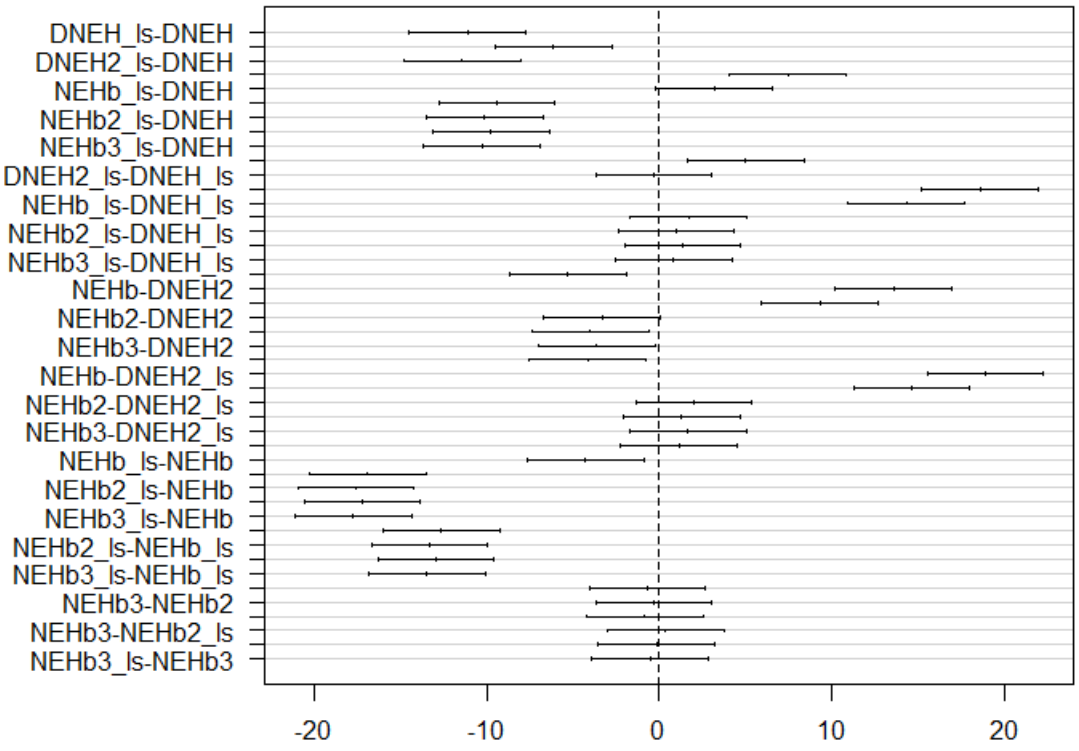

Figura 5.12: Gráfico de intervalos de Tukey para instancias grandes con búsqueda local

La diferencia que produce la búsqueda local puede verse de manera gráfica en la figura 5.7.

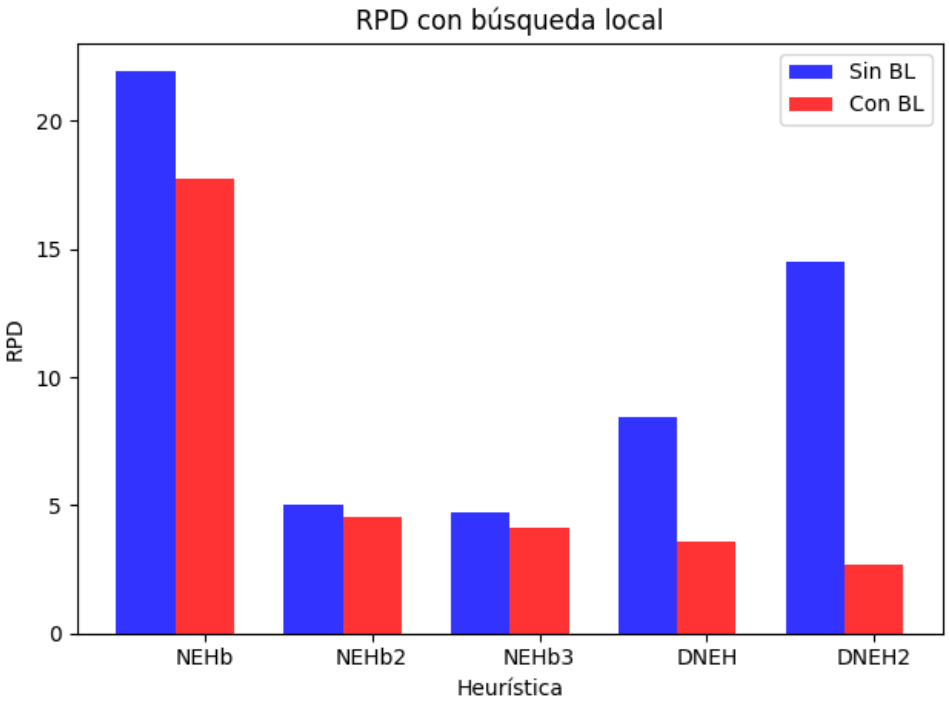

Figura 5.13: RPD media con y sin búsqueda local

A la vista de estos resultados, se puede concluir que la mejor heurística para resolver el problema de la DPFS heterogénea en instancias grandes es la heurística DNEH con búsqueda local posterior. Este método no presenta

diferencias significativas con NEHb2, DNEH2 y NEHb3 con búsqueda local, y con NEHb3 sin búsqueda local. El peor método es, de nuevo, la heurística NEHb sin búsqueda local.

5.7. Heurísticas en instancias semi-homogéneas

En los sets de instancias anteriores, cada trabajo podría tener una duración p_{ij} cualquiera. Para este apartado, definimos una instancia semi-homogénea como un conjunto donde la duración de una orden en una máquina, está relacionada con la duración en el resto de máquinas.

Para este trabajo, se ha generado una instancia donde los tiempos de procesamiento de un trabajo p_{ij} no pueden variar más de un 20 por ciento de una máquina a otra. Para lograr esto, el autor ha generado un tiempo estándar para cada trabajo. Después ha sumado o restado (de manera aleatoria) un valor que corresponde al 20 por ciento de su duración, se lo asignado a la máquina correspondiente, y ha repetido el proceso hasta completar todas las máquinas necesarias.

Este proceso sólo ha sido realizado para el set pequeño de instancias. De nuevo, se ha condensado la tabla RPD en la media para cada heurística, para facilitar una mejor comprensión de los datos (tablas 5.11 y 5.12).

Tabla 5.11: RPD para instancias homogéneas, heurísticas sin búsqueda local

	NEHb	NEHb2	DNEH	DNEH2	NEHb3
Media 8.88	3.38	22.66	15.56	2.65	

Tabla 5.12: RPD para instancias homogéneas con búsqueda local

	NEHb_ls	NEHb2_ls	DNEH_ls	DNEH2_ls	NEHb3_ls
Media 5.74	3.06	4.99	3.81	2.36	

El gráfico de intervalos de Tukey es el siguiente:

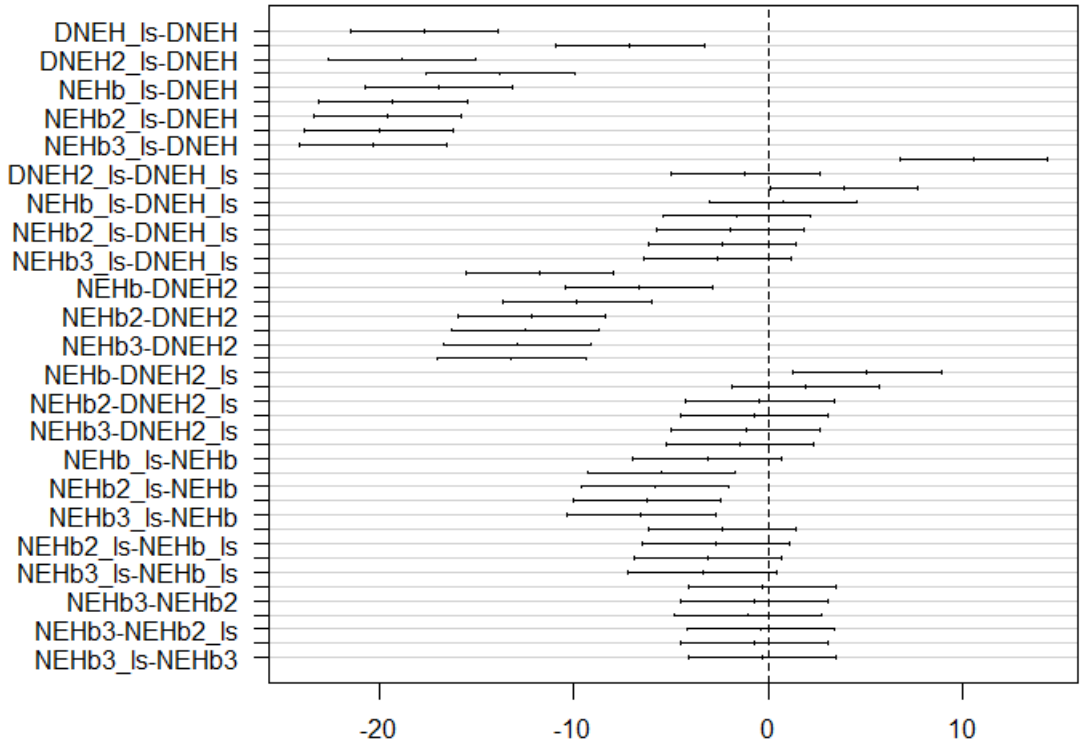

Figura 5.14: Gráfico de intervalos de Tukey para instancias pequeñas homogéneas con búsqueda local

La diferencia que produce la búsqueda local puede verse de manera gráfica en la figura 5.15.

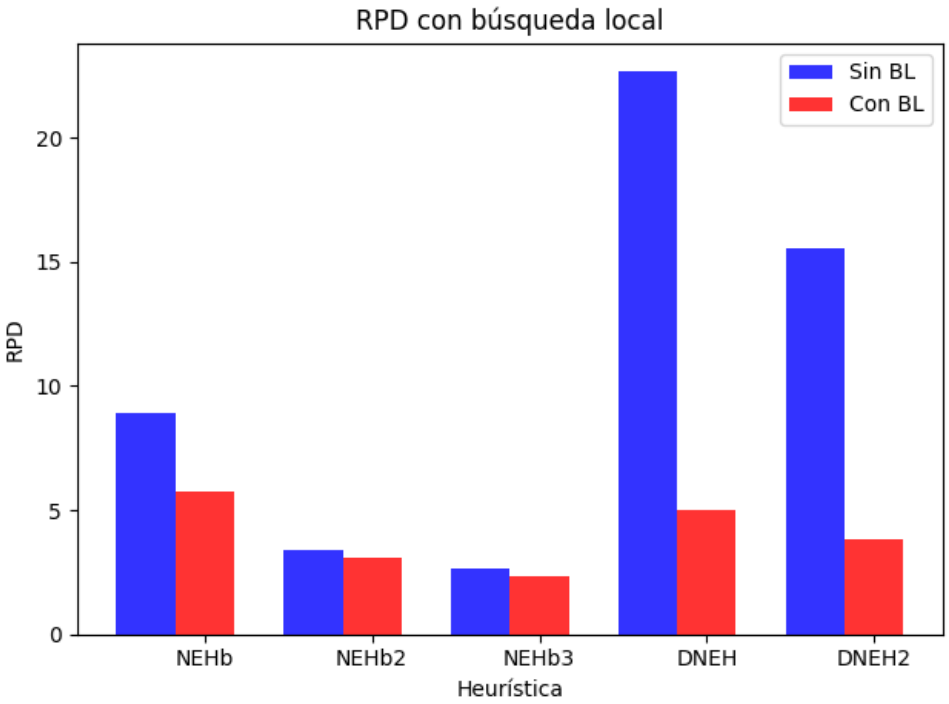

Figura 5.15: RPD media con y sin búsqueda local

Se puede apreciar que, en instancias homogéneas del problema, el rendimiento de las heurísticas es más uniforme.

Las peores heurísticas en este caso son DNEH y DNEH2 sin búsqueda local. Este resultado es esperado, ya que la heurística DNEH se basa en asignar el trabajo a la fábrica con menor tiempo de proceso. En instancias con tiempos de proceso similares, esta cualidad pierde sentido. Por otro lado, las mejores heurísticas son la NEHb2 y NEHb3 sin búsqueda local, y NEHb2, NEHb3, DNEH y DNEH2 con búsqueda local. No hay diferencia significativa entre todos estos métodos.

Es importante destacar que la heterogeneidad del problema sí ha afectado al rendimiento de las distintas heurísticas, haciendo necesario tener en cuenta este factor para seleccionar y continuar desarrollando métodos para resolver este problema. Es decir, cada heurística tiene ventajas o desventajas frente a determinados tipos de problemas, según cómo hayan sido construidas.

5.8. Coste computacional

El coste computacional de cada algoritmo es muy importante, ya que determina su viabilidad para instancias más grandes del problema. Cada algoritmo será más o menos rápido, para una potencia de procesamiento dada. Por otro lado, el tiempo de cálculo también estará influenciado por la instancia, es decir, el número de trabajos **N**, de máquinas **M** y de fábricas **F**.

En esta sección, se estudian los tiempos de cálculo mediante un ANOVA. Sólo se han tenido en cuenta los tiempos en instancias grandes, ya que para instancias pequeñas son demasiado cortos. En la figura 5.13 se puede apreciar los distintos tiempos (en segundos) según los factores mencionados anteriormente:

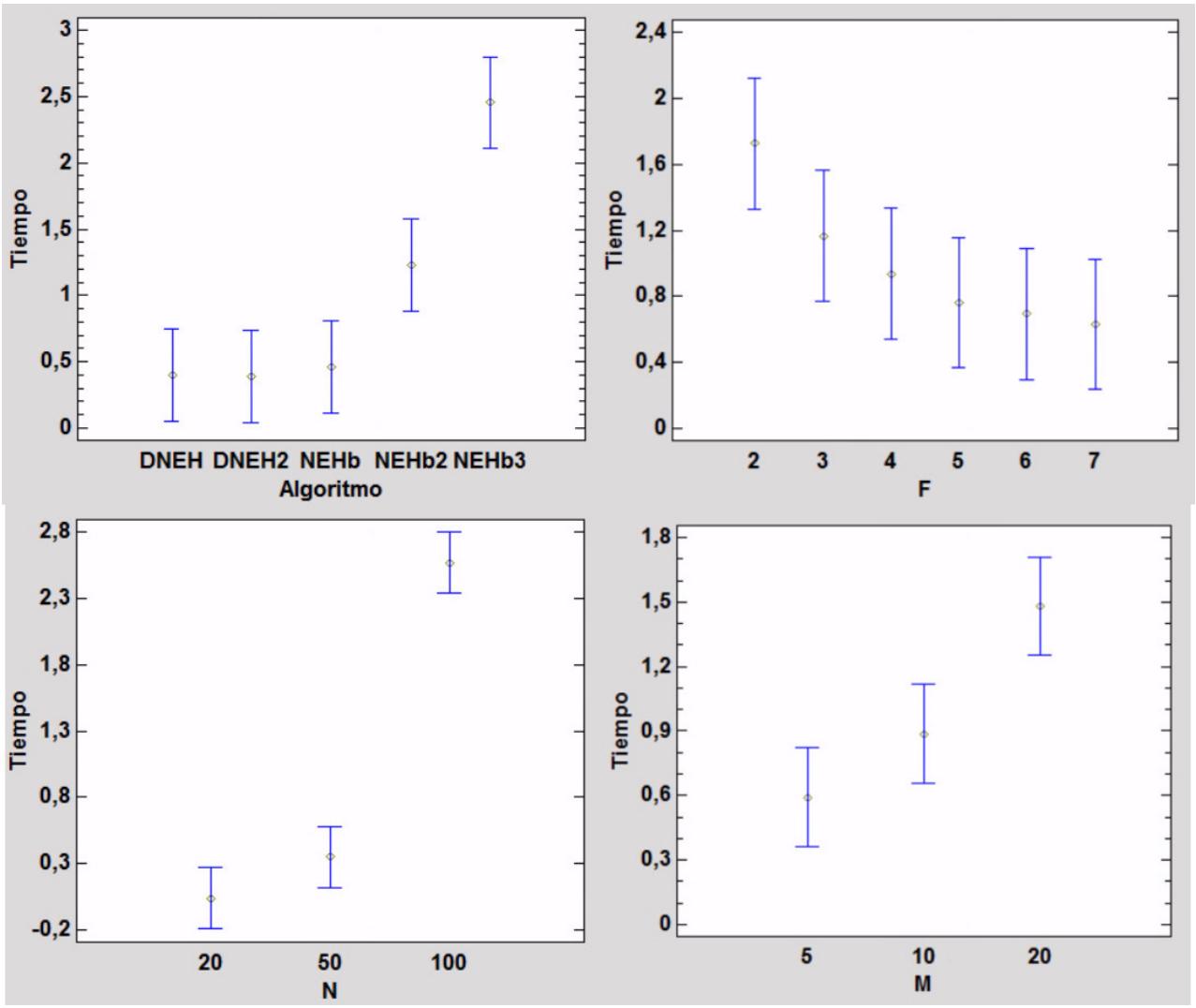

Figura 5.16: Tiempos de procesamiento medios

Como puede verse, las heurísticas tienen tiempos de cálculo distintos. Mientras que DNEH, DNEH2 y NEHb tienen tiempos muy bajos, NEHb2 y NEHb3 son significativamente más lentas.

Por otro lado, aumentar el número de trabajos y de máquinas causa un aumento exponencial del tiempo de cálculo. En este sentido, los resultados experimentales concuerdan con la teoría expuesta en los capítulos segundo (Descripción del problema) y tercero (Revisión bibliográfica).

Aumentar el número de fábricas disminuye el tiempo de cálculo, ya que los algoritmos deben hacer menos operaciones. Por ejemplo, para un problema con seis trabajos y seis fábricas, la solución sería inmediata: asignar cada trabajo a la fábrica donde menor tiempo de procesamiento tiene. Sin embargo, con una fábrica se debe tener

en cuenta la secuencia de los trabajos para obtener una secuencia eficiente.

Como puede verse en la figura 5.14, la interacción entre el algoritmo y el número de trabajos es muy relevante. Las heurísticas NEHb3 y NEHb2 aumentan mucho su coste computacional para instancias grandes del problema, mientras que NEHb, DNEH y DNEH2 se ven menos perjudicadas.

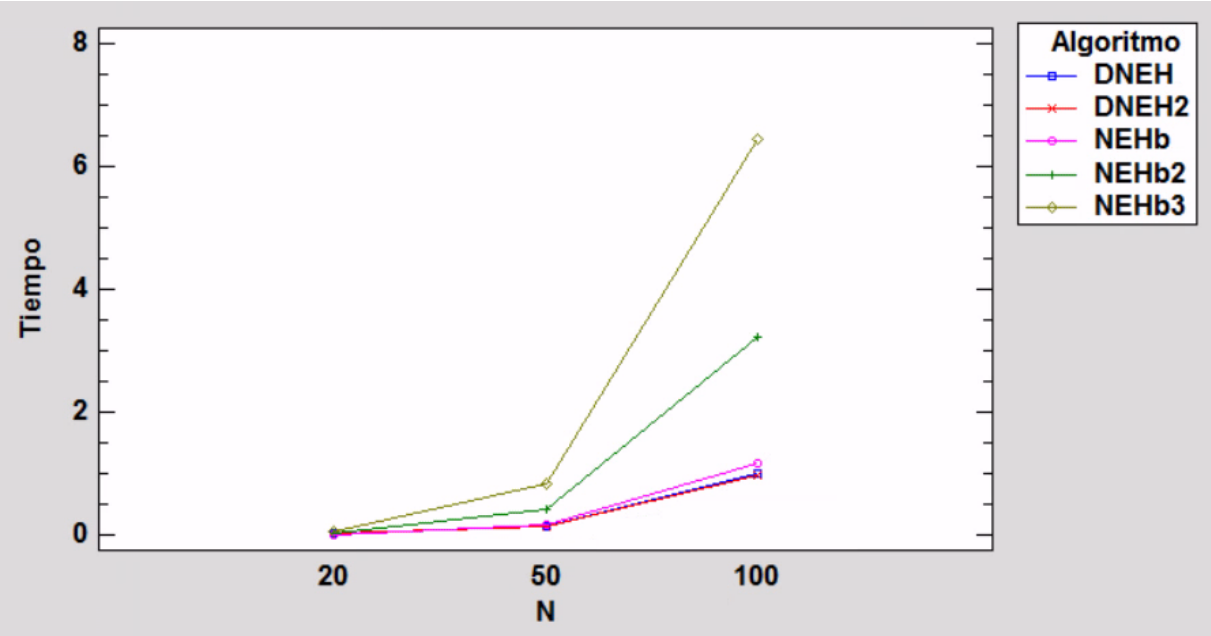

Figura 5.17: Interacción entre algoritmo y número de trabajos n

Capítulo 6

Conclusiones

6.1. Conclusiones

En este TFM se ha abordado una variante del problema DPFS, en concreto la que considera que cada fábrica puede tener tiempos de procesamiento distintos para el mismo trabajo (variante heterogénea). Este problema ha sido relativamente poco estudiado en la literatura científica, y se ha tratado de aportar soluciones aplicables en entornos de producción reales.

En primer lugar, se ha generado un set de instancias basado en trabajos anteriores, adaptado al caso de la DPFS heterogénea. Se espera que este conjunto de datos sirva como base para trabajos posteriores.

En segundo lugar, se han presentado cinco heurísticas, dos de ellas desarrolladas exclusivamente por el autor. Tras comprobar su rendimiento mediante el set de instancias (tanto grande como pequeño), se han extraído conclusiones realizando un análisis ANOVA.

Respecto a la calidad de las soluciones obtenidas, tanto para instancias grandes como pequeñas, las mejores heurísticas han resultado ser NEHb2 y NEHb3 (sin diferencias significativas entre ambas), seguidas de DNEH2 y DNEH. NEHb ha resultado ser la peor heurística.

Cuando se ha incorporado una búsqueda local, las heurísticas DNEH y DNEH2 han experimentado una clara mejoría en la calidad de sus soluciones. Cuando se realiza una búsqueda local posterior, las heurísticas no presentan diferencias significativas (a excepción de NEHb, que sigue siendo la que tiene un menor rendimiento).

En el apartado computacional, se ha encontrado que DNEH y DNEH2 presentan un coste computacional mucho menor que NEHb2 y que NEHb3, y que escalan mejor a medida que aumenta el tamaño de la instancia.

Por lo tanto, para resolver el problema de la DPFS heterogénea se recomienda utilizar la heurística DNEH2 con búsqueda local posterior. Esta heurística es eficaz obteniendo la mejor solución posible, y es eficiente con los recursos computacionales de los que se dispone.

Por último, se ha analizado la relevancia de la homogeneidad de los trabajos a la hora del desempeño de las heurísticas. Se ha encontrado que la homogeneidad sí afecta a los métodos empleados para hallar las soluciones, y se ha sugerido de qué manera lo hace.

6.2. Líneas de trabajo futuras

En el futuro, la DPFS heterogénea continuará recibiendo atención en el campo científico debido a que es un problema reciente, poco estudiado, y de gran aplicación en el ámbito empresarial. A continuación se proponen algunas líneas de investigación que no han sido cubiertas en este trabajo:

- Implementar algoritmos más avanzados como Scatter Search, GRASP o algoritmos genéticos.
- Agregar restricciones de recursos, transporte o tiempos de setup.
- Explorar el impacto de la heterogeneidad del problema mediante nuevos sets de instancias y algoritmos.
- Desarrollar nuevos métodos de búsqueda local.

Apéndice A

Relación con los Objetivos de Desarrollo Sostenible de la Agenda 2030

A.1. Objetivos de Desarrollo Sostenible (ODS)

Los ODS son una iniciativa global establecida por las Naciones Unidas para abordar los desafíos mundiales más cruciales y lograr un futuro más sostenible para todos. Fueron adoptados en septiembre de 2015 como parte de la Agenda 2030 para el Desarrollo Sostenible y constan de 17 objetivos interconectados que abarcan aspectos económicos, sociales y ambientales. Los ODS buscan equilibrar las tres dimensiones del desarrollo sostenible: económica, social y ambiental. En la tabla A.1 se enumeran los 17 ODS así como el grado de relación con el presente trabajo.

Tabla A.1: Tabla ODS

ODS	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar				X
ODS 4. Educación de calidad				X
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante				X
ODS 8. Trabajo decente y crecimiento económico		X		
ODS 9. Industria, innovación e infraestrcuturas	X			
ODS 10. Reducción de las desigualdades				X
ODS 11. Ciudades y comunidades sostenibles				X
ODS 12. Producción y consumo responsables		X		
ODS 13. Acción por el clima				X
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas				X
ODS 17. Alianzas para lograr objetivos				X

Este TFM se adapta principalmente al ODS 9: Industria, Innovación o Infraestructuras. Este ODS busca promover la construcción de infraestructuras resilientes, fomentar la industrialización inclusiva y sostenible y fomentar la innovación en todos los sectores económicos. Este trabajo se vincula con este objetivo por las siguientes razones:

- Innovación en la industria: Este proyecto emplea técnicas avanzadas de Investigación Operativa y programación. La aplicación de estas técnicas en el ámbito de la producción está estrechamente relacionada con la industria 4.0, una nueva revolución en la industria basada en la digitalización, el análisis de datos y la inteligencia artifical, que promete una modernización capaz de optimizar los procesos industriales.
- Mejora de la eficiencia: El uso de la investigación operativa en la industria permite aumentar la eficiencia de los procesos, reduciendo costes, gasto de material y gasto energético. Una secuenciación adecuada usa los recursos de una mejor manera, logrando una mayor producción con el mismo coste.
- Mejora de la competitividad: Al mejorar la eficiencia y optimizar procesos, la industria local es capaz de mejorar su competitividad, logrando tener una posición más fuerte respecto a industrias de otros países. El grado de desarrollo de una empresa está directamente relacionado con su desempeño en el país local frente a competidores, y en el extranjero cuando exporta sus productos.
- Industria resiliente: Una planificación óptima, con el uso de la investigación operativa, acelera y flexibiliza la fabricación y envío de productos, adaptándose mejor a las condiciones de un entorno que cambia cada vez más rápido. La producción distribuida, objeto de este trabajo, es una parte fundamental de las empresas actualmente, ya que permite reducir costes y aumentar la personalización del producto.

Además, el empleo de la investigación operativa en la industria mejora su competitividad, y por lo tanto, el crecimiento económico local, generando empleos de gran calidad. El uso eficiente de los recursos también permite llevar a cabo una producción más responsable.

Bibliografía

[1] Valerie Belanger, Angel Ruiz y Patrick Soriano. "Deployment and redeployment of Ambulance Vehicles in the management of a Prehospital Emergency Service". En: *INFOR* 50.1 (2012), págs. 1-30.

[2] B. Naderi y Ruben Ruiz. "The distributed permutation flowshop scheduling problem". En: *COMPUTERS & OPERATIONS RESEARCH* 37.4 (2010), págs. 754-768. issn: 0305-0548. doi: 10.1016/j.cor.2009.06.019.

[3] Paz Perez-Gonzalez y Jose M. Framinan. "A review and classification on distributed permutation flowshop scheduling problems". En: *European Journal of Operational Research* (2023). issn: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2023.02.001. url: https://www.sciencedirect.com/science/article/pii/S0377221723001170.

[4] Sara Hatami, Ruben Ruiz y Carlos Andres-Romano. "Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times". En: *INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS* 169 (2015), págs. 76-88. issn: 0925-5273. doi: 10.1016/j.ijpe.2015.07.027.

[5] Tao Meng y Quan-Ke Pan. "A distributed heterogeneous permutation flowshop scheduling problem with lot-streaming and carryover sequence-dependent setup time". En: *SWARM AND EVOLUTIONARY COMPUTATION* 60 (2021). issn: 2210-6502. doi: 10.1016/j.swevo.2020.100804.

[6] Weishi Shao, Zhongshi Shao y Dechang Pi. "Modelling and optimization of distributed heterogeneous hybrid flow shop lot-streaming scheduling problem". En: *EXPERT SYSTEMS WITH APPLICATIONS* 214 (2023). issn: 0957-4174. doi: 10.1016/j.eswa.2022.119151.

[7] Ruben Ruiz, Quan-Ke Pan y Bahman Naderi. "Iterated Greedy methods for the distributed permutation flowshop scheduling problem". En: *OMEGA-INTERNATIONAL JOURNAL OF MANAGEMENT SCIENCE* 83 (2019), págs. 213-222. issn: 0305-0483. doi: 10.1016/j.omega.2018.03.004.

[8] Bahman Naderi y Ruben Ruiz. "A scatter search algorithm for the distributed permutation flowshop scheduling problem". En: *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH* 239.2 (2014), págs. 323-334. issn: 0377-2217. doi: 10.1016/j.ejor.2014.05.024.

[9] Sara Hatami, Ruben Ruiz y Carlos Andres-Romano. "The Distributed Assembly Permutation Flowshop Scheduling Problem". En: *INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH* 51.17 (2013), págs. 5292-5308. issn: 0020-7543. doi: 10.1080/00207543.2013.807955.

[10] Sven Schulz, Martin Schoenheit y Janis S. Neufeld. "Multi-objective carbon-efficient scheduling in distributed permutation flow shops under consideration of transportation efforts". En: *JOURNAL OF CLEANER PRODUCTION* 365 (2022). issn: 0959-6526. doi: 10.1016/j.jclepro.2022.132551.

[11] Haoran Li, Xinyu Li y Liang Gao. "A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem". En: *Applied Soft Computing* 100 (2021), pág. 106946. issn: 1568-4946. doi: https://doi.org/10.1016/j.asoc.2020.106946. url: https://www.sciencedirect.com/science/article/pii/S156849462030884X.

[12] Guangchen Wang, Xinyu Li, Liang Gao y Peigen Li. "Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified MOEA/D". En: *Swarm and Evolutionary Computation* 62 (2021), pág. 100858. issn: 2210-6502. doi: https://doi.org/10.1016/j.swevo.2021.100858. url: https://www.sciencedirect.com/science/article/pii/S2210650221000195.

[13] Weishi Shao, Zhongshi Shao y Dechang Pi. "An Ant Colony Optimization Behavior-Based MOEA/D for Distributed Heterogeneous Hybrid Flow Shop Scheduling Problem Under Nonidentical Time-of-Use Electricity Tariffs". En: *IEEE Transactions on Automation Science and Engineering* 19.4 (2022), págs. 3379-3394. doi: 10.1109/TASE.2021.3119353.

[14] Michael L. Pinedo. *Scheduling. Theory, Algorithms and Systems*. Springer, 2012.

[15] R Ruiz y C Maroto. "A comprehensive review and evaluation of permutation flowshop heuristics". En: *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH* 165.2 (2005). 8th International Workshop on Project Management and Scheduling, Univ Valencia, Valencia, SPAIN, APR 03-05, 2002, págs. 479-494. issn: 0377-2217. doi: 10.1016/j.ejor.2004.04.017.

[16] Muhammad Nawaz, E Emory Enscore e Inyong Ham. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". En: *Omega* 11.1 (1983), págs. 91-95. issn: 0305-0483. doi: https://doi.org/10.1016/0305-0483(83)90088-9. url: https://www.sciencedirect.com/science/article/pii/0305048383900889.

[17] Johnson SM. "Optimal two- and three-stage production schedules with setup". En: *Naval Research Logistics Quarterly* (1954).