Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Heurística en taller de flujo regular con permutación y objetivo de sostenibilidad

Autor: Pilar Megías Calvo
Tutor: Paz Pérez González

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Heurística en taller de flujo regular con permutación y objetivo de sostenibilidad

Autor:
Pilar Megías Calvo

Tutor:
Paz Pérez González
Profesor titular

Dpto. de Ingeniería de las Tecnologías Industriales
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Grado: Heurística en taller de flujo regular con permutación y objetivo de sostenibilidad

Autor: Pilar Megías Calvo

Tutor: Paz Pérez González

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

Llega el momento de cerrar una etapa muy importante en mi vida y solo me queda agradecer enormemente el apoyo que he recibido por parte de mi familia, amigos y profesores.

Gracias a todas las personas que me han acompañado en este duro camino, por apoyarme, aconsejarme, animarme y creer en mí.

Pilar Megías Calvo

Sevilla, 2021

Resumen

En este trabajo fin de grado se ha realizado un estudio sobre la minimización del consumo energético total de todas las máquinas dentro de un entorno de taller de flujo regular de permutación, también conocido como *flowshop*. El objetivo de este estudio es doble. Por un lado, analizar el impacto medioambiental que supone mantener máquinas encendidas o en estado de *stanby* mientras no se procesan trabajos, y por otro, minimizar el coste energético que esto supone. Para ello, se ha propuesto el algoritmo NEH, empleando diversas reglas de despacho como órdenes iniciales para aplicar el algoritmo, las cuales están estrechamente relacionadas con los costes asociados a cada máquina, y una regla de desempate basada en un objetivo común en la programación de operaciones, el *total flowtime.*

El problema planteado se ha resuelto con el algoritmo propuesto en sus diferentes versiones (secuencias iniciales y regla de desempate) sobre una batería de instancias. Para contar con un análisis completo y poder comparar las versiones de la NEH, los resultados se han medido en función de los tiempos de ejecución de cada método empleado, de los valores de la función objetivo, sus valores medios y ARPD (*average relative percentage deviation*).

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El presente trabajo fin de grado se enmarca en el departamento de Organización y Gestión de Empresas I y se ha realizado con los conocimientos adquiridos en la asignatura de programación de operaciones perteneciente a la titulación de Grado en Ingeniería de las Tecnologías Industriales. Este capítulo contará con una contextualización la programación de operaciones en la industria con el fin de comprender el desarrollo y objetivo del proyecto. Además, se define la estructura seguida en el documento.

1.1 Objeto y justificación

La programación de la producción se ocupa de la asignación de recursos a las tareas a lo largo del tiempo (Jose M. Framinan, Leisten, and Ruiz García 2014). En general, esta definición puede abarcar un gran número de aplicaciones de la vida real, como la asignación de enfermeras a los turnos de sus hospitales, la programación de aviones en los aeropuertos, las unidades de procesamiento en un entorno informático, etc. La programación de operaciones es una forma de toma de decisiones que desempeña un papel crucial en las industrias manufactureras y de servicios. Además, se han convertido en una necesidad para sobrevivir en el mercado, ya que las empresas tienen que cumplir con las fechas de entrega que se han comprometido con los clientes y programar las actividades de forma que se utilicen los recursos disponibles de manera eficiente (Pinedo 2012).

Actualmente, los elevados consumos de energía suponen un problema vital para las industrias manufactureras debido al elevado coste que esto supone y los impactos medioambientales negativos que conllevan, tales como las emisiones de dióxido de carbono a la atmósfera o calentamiento global (Öztop et al. 2020). Como consecuencia, se ha comenzado a investigar sobre algoritmos de programación de la producción para alcanzar un objetivo de sostenibilidad energética en las empresas, las cuales centran sus esfuerzos en desarrollar maquinaria y equipamiento energéticamente eficiente con el fin de conseguir una reducción de sus costes energéticos. Sin embargo, varios investigadores (Bierer and Götze 2011) han observado como el consumo puede reducirse mediante una gestión más eficiente de recursos y la programación de la producción (Fang et al. 2013). Un ejemplo de este tipo de estudios fue el realizado por (Fang et al. 2013) donde se consideró un problema de programación del taller de flujo con una restricción en el consumo máximo de energía, además de los objetivos tradicionales basados en el tiempo.

Es por ello por lo que el enfoque del presente documento se basa en minimizar el consumo energético de las máquinas de una planta mediante la programación de la producción. En este caso, se ha hecho una simplificación del problema propuesto por (Öztop et al. 2020), donde se considera un objetivo de eficiencia productiva. Por lo tanto, en este trabajo se ha estudiado como función objetivo la minimización del tiempo ocioso total entre las máquinas multiplicado por su coste energético. Para ello, se emplea un método aproximado, en concreto, la heurística constructiva NEH y se proponen diferentes variantes de la misma usando reglas de despacho o *priority rules* además de una regla de desempate basada en el objetivo clásico de *total flowtime.* El motivo de emplear como regla de desempate el valor total de los tiempos de finalización de cada trabajo es para no perder de vista otros objetivos esenciales en las empresas.

Un aspecto importante a tener en cuenta es que cuando se habla de tiempo ocioso de una máquina se hace referencia al tiempo que dicha máquina se encuentra parada tras haber procesado un trabajo y se encuentra a la espera de procesar el siguiente, es decir, las máquinas se encuentran en un estado de *standby* donde, aunque no estén funcionando, están consumiendo energía. Por lo tanto, es esencial poder reducir el tiempo que las máquinas se encuentran en estado de *standby*, ya que dicha reducción de tiempo se verá traducida en una reducción de costes.

Para futuras líneas de investigación, resulta interesante analizar el coste asociado a cada máquina en función de la franja horaria del día en la que se encuentre funcionando, siendo en muchos casos más bajo por la noche. De esta forma, los costes no serían constantes en el tiempo, si no que variarían en función de la hora del día y sería

conveniente estudiar en que horario se puede procesar cada trabajo, procesando en aquellas horas donde el coste es más barato los trabajos en las máquinas con costes más elevados.

1.2 Estructura

El presente documento se divide en 6 capítulos, donde se sigue la estructura descrita a continuación.

En el primer capítulo se introduce el tema a tratar en el estudio, así como los objetivos y justificación del mismo. Finalmente, se indica como se encuentra estructurado el documento, indicando el contenido de cada capítulo.

Posteriormente, en el segundo capítulo, se definen los conceptos básicos y los métodos de resolución empleados en la programación de operaciones.

A continuación, en el capítulo tres, se describe el problema y se presentan los métodos de resolución planteados, incluyendo la modificación propuesta en la NEH, las reglas de despacho empleadas y la regla de desempate elegida.

En el capítulo 4 se expresan todos los resultados obtenidos, la comparativa de la NEH con diferentes secuencias iniciales, comparativa de la NEH con regla de desempate y, finalmente, sin tener en cuenta los pesos. Para analizar la consistencia de las soluciones se muestran los ARPD de cada solución, valores de la función objetivo y tiempos de cómputo. Además, se indica la herramienta empleada en el estudio.

Por último, en el capítulo cinco, se comentan las conclusiones obtenidas en el trabajo.

2 PROGRAMACIÓN DE OPERACIONES

En primer lugar, se procede a introducir y explicar los conceptos básicos sobre la programación de operaciones. Además, se describen los métodos de resolución empleados en el presente documento.

2.1. Conceptos básicos.

Se entiende por programación de la producción o *manufacturing scheduling* el proceso de asignación de recursos para la fabricación de un conjunto de productos. Como resultado, se obtiene un programa de producción o *production schedule.* Además, también resulta interesante conocer el proceso de control de la producción, llamado así al conjunto de mecanismos utilizado para monitorizar las desviaciones del programa de producción y la ejecución de acciones correctoras. (Framiñan Torres, Pérez González, and Fernández-Viagas Escudero 2020). También se puede definir la programación de la producción como el proceso de toma de decisiones que consiste en asignar un conjunto de operaciones o tareas necesarias para fabricar un conjunto de productos a los recursos existentes en el proceso de fabricación, así como los plazos para iniciar estas operaciones o tareas (Pinedo 2012). Por lo tanto, se asume que el programa está determinado por un conjunto, conocido y finito, de trabajos que deberán ser procesados por máquinas. Además, las máquinas y los trabajos son independientes y todos los datos siguientes son deterministas y conocidos a priori (Jose M. Framinan, Leisten, and Ruiz García 2014). A continuación, se definen algunos de los elementos básicos que componen un problema de programación de la producción y la notación empleada en este trabajo:

- Trabajos (*jobs*): $N = \{1, ..., n\}$. Los trabajos son el resultado o producto de una operación en una máquina. Es decir, los productos que se van a fabricar, o las unidades de trabajo en las que se puede dividir la actividad de fabricación se denomina en la literatura de programación como trabajos.
    - Índices para los trabajos: $j \in N$
- Máquinas (machines): $M = \{1, \ldots, m\}$. Se define como el recurso productivo capaz de realizar operaciones de transporte o transformación de material. Estos recursos productivos pueden ser herramientas, accesorios o incluso recursos humanos. Sin embargo, por simplificar el problema todos estos recursos serán englobados en la categoría de máquinas (Jose M. Framinan, Leisten, and Ruiz García 2014).
    - Índices para las máquinas: $i \in M$
- Tiempo de llegada o *release date* (*rj*): El *release date* de un trabajo *j* se refiere al instante de tiempo en el que un trabajo está listo para poder ser procesado en el sistema.
- Tiempo de proceso o *processing time* (*pij*): Representa el tiempo de proceso del trabajo *j* en la máquina *i*. El subíndice i es omitido si el tiempo de proceso del trabajo *j* no depende de la máquina donde será procesado o si solo es procesado en una máquina.
- Fecha de llegada o *due dates* (*dj*): El due date dj representa la fecha comprometida de envío o fecha de finalización, por ejemplo, la fecha en la que se promete al cliente que se entregará el trabajo. Sin embargo, entregar un envío tarde está permitido, pero se impondrá una penalización. Tan solo es obligatoria en los casos que se establecen *deadlines* y se denota como $\overline{dj}$.
- Peso o *weight* (*wi*): El peso *wj* es un factor de prioridad, dándole más importancia a un trabajo que a otro en el sistema

2.1.1 Programa

Un programa o *schedule* es la asignación, en una escala temporal concreta, de las máquinas de una empresa para la fabricación de un conjunto de trabajos (Framiñan Torres, Pérez González, and Fernández-Viagas Escudero 2020). En general, un programa determina el comienzo y el final de cada operación a realizar en cada recurso productivo y en muchos casos, puesto que el tiempo de proceso es conocido, es suficiente obtener el tiempo de comienzo o de fin. (Framiñan Torres, Pérez González, and Fernández-Viagas Escudero 2020). No obstante, hay veces en las que si existen operaciones interrumpibles esta información puede no ser suficiente, ya que se debería establecer el instante de interrupción y de continuación de las operaciones para poder determinar el tiempo de finalización.

La forma más habitual para representar la programación es mediante diagramas de Gantt. Este diagrama fue popularizado por Henry Laurence Gantt entre 1910 y 1915 y tiene como objetivo la representación del plan de trabajo, mostrando todas las actividades a realizar, el momento de su comienzo, su terminación y la forma en la que las distintas actividades se encadenan entre sí (Escuela Superior de Ingenieros de la Universidad de Sevilla. 2007). En la Figura 1 se puede observar como en el eje de abscisas se representa la escala de tiempo que transcurre y en el eje de ordenadas se muestra la ruta en cada una de las máquinas. En este caso concreto, se sigue la misma secuencia en cada máquina y cada trabajo realiza la misma ruta.

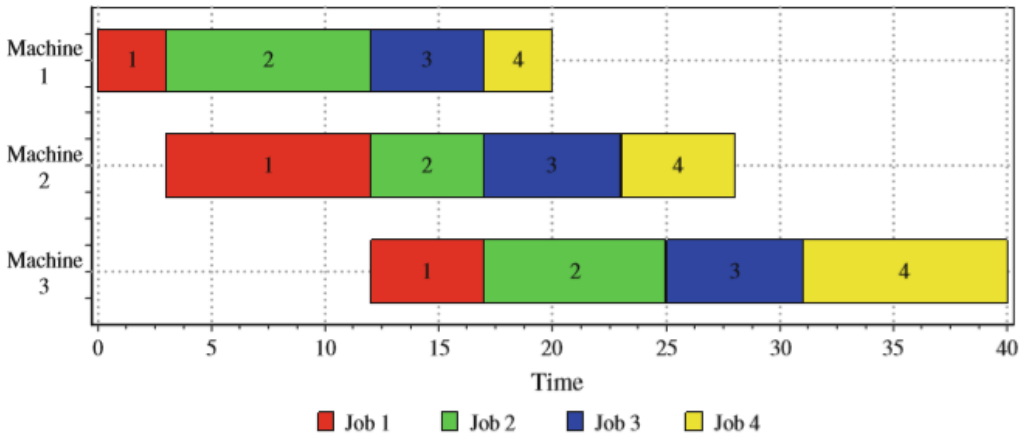

Figura 1. Diagrama de Gantt (Jose M. Framinan, Leisten, and Ruiz García 2014).

Es importante tener en cuenta que no todos los programas que se quieran probar pueden tener en cuenta todas las características de los trabajos, máquinas u operaciones. En este caso, cuando no se pueden cumplir con todas las características, el programa se denomina no admisible o *unfeasible shedule*, ya que no puede traducirse directamente como órdenes para el taller. El objetivo principal de un proceso de programación de producción es encontrar al menos un programa factible, es decir, que cumpla todas las características y restricciones del proceso productivo. Siempre que exista más de un programa admisible se deberá elegir al menos uno, que será el que se ejecute en planta. Para decidir cual elegir se establecen uno o varios criterios de evaluación para la posterior selección de un único programa.

Otra apreciación importante es la diferenciación entre programa y secuencia. Una secuencia establece el orden en que cada trabajo comienza a procesarse en cada máquina, pudiendo ser representada mediante un vector para cada máquina. Sin embargo, como ya se ha definido anteriormente, un programa determina la asignación de fechas de comienzo a fin de cada trabajo en cada máquina (Framiñan Torres, Pérez González, and Fernández-Viagas Escudero 2020). Además, dependiendo del problema, dada una secuencia se puede obtener de forma única un programa, y de ahí que a veces se confundan los términos secuencia y programa.

2.1.2 Modelos

Para conseguir la definición y comprensión de un problema de programación de operaciones, en el año 1917 Grahm, Lawler, Lenstra y Rinnooy Kann introdujeron la notación $\alpha|\beta|\gamma$ para referirse a los tres conceptos fundamentales por los que se rige el problema: entorno, restricciones y objetivo, (Graham et al. 1979). A

continuación, se definen cada uno de ellos.

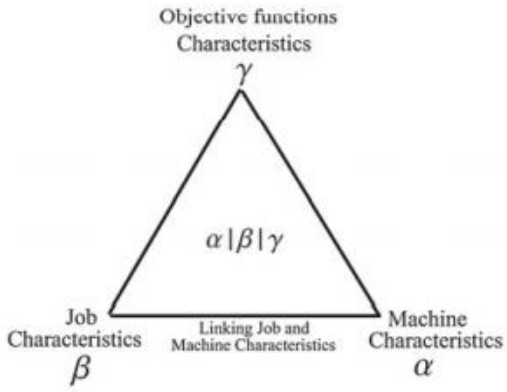

Figura 2. Modelos de programación de la producción. (Jose M. Framinan, Leisten, and Ruiz García 2014).

La primera de ellas (α) hace referencia a las máquinas y su distribución. Actualmente, el diseño o *layout* de una planta de producción resulta esencial en la planificación de la cadena de suministro, ya que puede suponer una considerable minimización de los costes. Además, desde la filosofía de Lean Manufacturing se considera que mediante un correcto *layout* se puede ayudar a la empresa a alcanzar la reducción tanto de costes como de materiales. Por otro lado, es necesario no solo saber cómo las diferentes máquinas están dispuestas en la planta, si no también conocer las rutas de cada trabajo (Medero 2012).

Existen distintos tipos de entorno o *layouts* en función de las características de las máquinas, pero en este trabajo se aborda solo el entorno tipo taller de flujo de permutación, también conocido como *flowshop*, el cúal se explica más adelante.

A continuación, se explican brevemente cada uno de los entornos posibles, así como su notación:

- Una sola máquina o *single machine*: α=1. El entorno está formado por una sola máquina
- Máquinas paralelas o *parallel machine*:
    - α=P: Hace referencia a un conjunto de máquinas paralelas idénticas (*identical parallel machines*)
    - α=Q: Conjunto de máquinas paralelas uniformes (*uniform parallel machines*), donde cada máquina puede adquirir una velocidad diferente al resto.
    - α=R: Se refiere a las máquinas paralelas no relacionadas (*unrelated parallel machines*), es decir, el tiempo de proceso de cada trabajo depende de la máquina a la que sea asignado.
- Taller de flujo o *flowshop*: α=Fm. Cada trabajo sigue la misma ruta.
- Taller de trabajos o *jobshop*: α: Jm. Existen diferentes rutas predeterminadas.
- Taller abierto o *openshop*: α=Om. En este caso no hay una ruta predeterminada y es considerado el taller más complejo.

El subíndice *m* indica el número de máquinas que existen en dicho entorno. Por ejemplo, en el caso de tener un taller de flujo con tres máquinas la notación sería F3.

El segundo concepto de denota con la letra β e indica las restricciones en cuanto a los trabajos. Entre las más comunes se encuentran:

- Interrupciones: Indica que el procesamiento de un trabajo en una máquina puede verse interrumpido debido a diversas causas como, por ejemplo, tener que parar por necesidad del sistema o por avería de los recursos.
    - *Non-resumable*: β=*pmtn-non-resumable*. El trabajo realizado anteriormente se pierde por complete y se debe empezar de nuevo.
    - *Semi-resumable*: β=*pmtn-semi-resumable.* En este caso el trabajo realizado con anterioridad no se pierde por completo, tan solo parcialmente, por lo que se puede recuperar.

- *Resumable:* β=*pmtn-resumable.* A diferencia de los casos anteriores aquí no se pierde nada del trabajo realizado previo a la interrupción.

- Tiempo de llegada (*rj*): β=*rj.* Se utiliza para denotar que el momento de llegada del trabajo *j* no ha sido en el instante cero. En caso de que no se indique como restricción se asume que todos los trabajos pueden ser procesados a partir del instante cero.
- *Deadlines* ($\overline{dj}$): β=$\overline{dj}$. Esta restricción obliga a cumplir con la fecha estipulada de entrega del producto. En caso de que se denote como *dj* implica que se pueden retrasar dichas fechas, pero supondrá una penalización.

Finalmente, el tercer concepto es el referente a la función objetivo, Actualmente, las funciones objetivo en función de qué criterios se empleen puedes ser clasificadas en función al coste, tiempo, calidad y flexibilidad. Generalmente están relacionadas con el tiempo de terminación de los trabajos, intentando en todo momento que este sea el menor posible. Este último concepto se denota como γ.

Entre las funciones más habituales se pueden encontrar las siguientes medidas:

- *Cj*: Tiempo de terminación del trabajo *j* (*completion time*). Instante en el que el trabajo termina de procesarse en una máquina.
- *Fj*: Tiempo de flujo del trabajo *j* (*flowtime*). Tiempo que el trabajo *j* está en el entorno.

$$Fj = Cj - rj$$

- *Lj*: Retraso del trabajo *j* (*lateness*).

$$Lj = Cj - dj$$

- *Tj*: Tardanza del trabajo *j* (*tardiness*). No siempre se termina antes de la fecha de entrega.

$$Tj = máx\{0, Lj\} = máx\{0, Cj - dj\}$$

- *Ej*: Adelanto del trabajo *j* (*earliness*). No siempre el trabajo termina antes de su fecha de entrega.

$$Ej = máx\{0, -Lj\} = máx\{0, dj - Cj\}$$

- *Uj*: Trabajo tarde (*tardy job*)

  $Uj = 1 \quad si\ \ Tj > 0 (Cj > dj)$

  $Uj = 0 \quad en\ caso\ contrario$

El objetivo siempre sera minimizar la función objetivo (f) y encontrar un programa factible. Es por ello por lo que existen dos tipos de funciones a estudiar:

- Max-form:

$$f = \max_{1 \le j \le n} g(Cj)$$

- Sum-form

$$f = \sum_{j=1}^{n} g(Cj)$$

Por otro lado, las funciones objetivos pueden ser ponderados, es decir, incluyendo pesos:

- Max-form:

$$f = \max_{1 \le j \le n} wj * g(Cj)$$

- Sum-form

$$f = \sum_{j=1}^{n} wj * g(Cj)$$

2.2. Métodos de resolución

Un método de programación es un procedimiento formal que se puede aplicar a cualquier instancia de un modelo de programación con el fin de obtener un programa factible y que obtenga buenos resultados respecto a un objetivo buscado. Además, puede considerarse que un método de programación puede considerarse como un procedimiento que toma una instancia de un modelo de programación como entrada para producir (al menos) un programa a la salida. Ya que el procedimiento puede aplicarse a cualquier instancia de un modelo de programación, se puede afirmar informalmente que un método de programación resuelve un problema de programación. Dado que el procedimiento debe ser formal, puede describirse mediante un conjunto finito de pasos, por lo que puede ser codificado y ejecutado en un ordenador (Jose M. Framinan, Leisten, and Ruiz García 2014).

La necesidad de programar la producción se remonta varios años atrás donde la programación se hacía manualmente. En la actualidad aún existen muchas programaciones manuales donde los trabajos se lanzan al taller y son secuenciados en las máquinas para que se pongan en marcha tan pronto como estas estén listas. Sin embargo, rara vez hay precisión en este tipo de tarea. Por lo tanto, la mayoría de los sistemas de producción suelen estar fuera de las empresas manufactureras (Jose M. Framinan, Leisten, and Ruiz García 2014).

La programación manual suele basarse en las llamadas reglas de despacho o *dispatching rules,* las cuales lanzan tareas a las máquinas, normalmente en función de un cálculo que devuelve un índice o importancia relativa u orden en el que deben procesarse los trabajos. Las reglas de despacho son métodos clásicos y, por tanto, no pueden atribuirse fácilmente a un solo autor ya que sus orígenes son difíciles de rastrear. Básicamente, una regla de despacho mantiene una lista de tareas elegibles o pendientes. Estas tareas son todos los trabajos en una *single machine* o las tareas que componen un trabajo completo en un *flowhop* o *jobshop* (Jose M. Framinan, Leisten, and Ruiz García 2014). Las reglas de despacho más comunes son:

- *Shortest Processing Time* (SPT). Las tareas son ordenadas de menor a mayor tiempo de proceso de cada trabajo en todas las máquinas.
- *Longest Processing Times* (LPT). En este caso, se da prioridad a los trabajos con mayor tiempo de proceso en todas las máquinas, al contrario que SPT.
- *Earliest Due Date* (EDD). Con esta regla recibe mayor prioridad aquel trabajo con la menor fecha de entrega o *deadline*.
- *Minumum Slack* (MS). Se procesan en función de la mínima holgura, siendo esta la cantidad de tiempo que tiene una tarea antes de que se retrase. Esta holgura disminuye a medida que se programan más trabajos.

El principal objetivo de toda programación de la producción, como se ha comentado con anterioridad, será siempre encontrar un programa admisible o *feasible schedule,* el cual cumpla con todas las restricciones del proceso productivo, y el procedimiento formal por el cual se obtienen uno o varios programas se denomina algoritmo. (Framiñan Torres, Pérez González, and Fernández-Viagas Escudero 2020). Actualmente existen numerosos estudios acerca de los algoritmos y se pueden clasificar en función del tipo de solución que aportan:

- Algoritmos exactos, los cuales ofrecen una solución óptima y única.
    - Algoritmos constructivos exactos, que engloban algoritmos de problemas polinomiales como, por ejemplo, el algoritmo de Lawler o Moore.
    - Algoritmos enumerativos, los cuales no son polinomiales y solo proporcionan soluciones óptimas en instancias pequeñas.
- Algoritmos aproximados, los cuales no garantizan un óptimo al problema propuesto. Entre ellos se encuentran las heurísticas constructivas, heurísticas de mejora y las metaheurísticas. En principio, se

puede pensar que los algoritmos aproximados no son una buena opción si existen enfoques exactos y, de hecho, su uso generalizado se justifica por la complejidad computacional inherente a la mayoría de los modelos de programación de la producción. Los algoritmos aproximados suelen dividirse, como se ha mencionado anteriormente, en heurísticos y metaheurísticos, siendo la principal diferencia entre ellos que los primeros son para empleados para un modelo concreto, mientras los segundos constituyen procedimientos más genéricos (Jose M. Framinan, Leisten, and Ruiz García 2014).

Además, a la hora de analizar el tipo de solución obtenida en cada heurística, existen distintas clasificaciones de problemas en función de su complejidad. En el problema estudiado en este documento se tratará un problema *NP-hard*, es decir, resulta imposible encontrar un algoritmo eficiente para obtener una solución óptima.

3 DESCRIPCIÓN DEL PROBLEMA Y MÉTODOS DE RESOLUCIÓN PLANTEADOS

En este capítulo se describe el problema objetivo estudiado en el presente documento, la notación del problema y las suposiciones generales a tener en cuenta. Además, se definen los métodos de resolución empleados para abordar el problema.

3.1 Descripción del problema

El problema objetivo abordado en este trabajo se basa en la minimización de tiempo ocioso entre las máquinas de un entorno *flowshop*. En este tipo de entorno En un entorno tipo taller donde existe una misma ruta predeterminada, se dice que es un taller de flujo regular o *flowshop*. En un flowshop, hay un conjunto finito de máquinas en serie y cada una de ellas realiza una operación diferente. Por otro lado, cada trabajo es procesado por todas las máquinas en el mismo orden, es decir, todos tienen la misma ruta. (Jose M. Framinan, Leisten, and Ruiz García 2014). Además, existe el caso en el que la secuencia es la misma para cada máquina, también conocido como taller de flujo regular con permutación o *permutation flowshop*, que será el caso abordado en este estudio. A continuación, en la Figura 3 se muestra un esquema de la distribución de máquinas y flujo en este tipo de entornos.

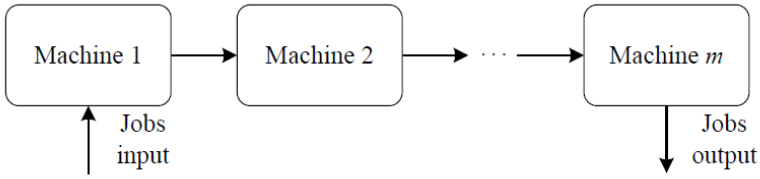

Figura 3. Representación del flujo de trabajo en un *flowshop* (Jose M. Framinan, Leisten, and Ruiz García 2014).

El número de soluciones que se pueden obtener en un problema de taller de flujo es $(n!)^m$ posibles soluciones, siendo n el número de trabajos y m el número de máquinas. En este caso, sería necesario dar una secuencia para cada máquina. Sin embargo, en el caso de la restricción de permutación, este número de reduce a $(n!)$ programas, y sería necesario dar tan solo una secuencia, ya que será la misma para todas las máquinas (Framiñan Torres, Pérez González, and Fernández-Viagas Escudero 2020).

Tabla 1 Representación de los tiempos de proceso (Jose M. Framinan, Leisten, and Ruiz García 2014).

Machine (i)	Job (j)				
	1	2	3	4	5
1	31	19	23	13	33
2	41	55	42	22	5
3	25	3	27	14	57
4	30	34	6	13	19

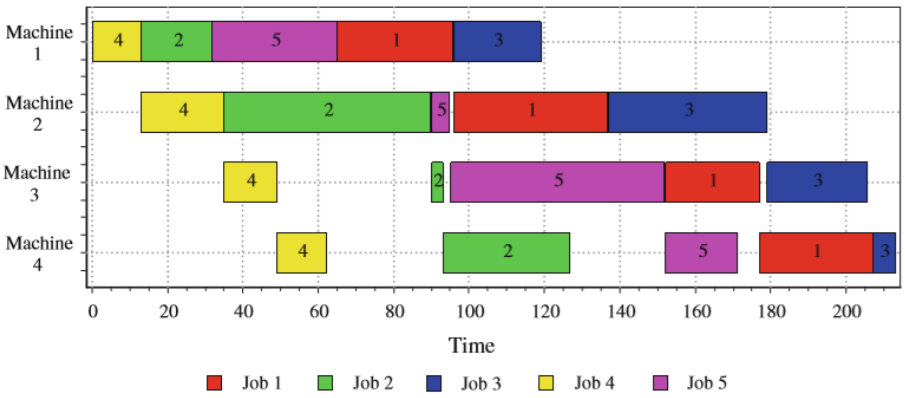

Figura 4. Diagrama de Gantt de un *flowshop* (Jose M. Framinan, Leisten, and Ruiz García 2014).

En la Tabla 1 se observa un ejemplo un diagrama de Gantt de cuatro máquinas en serie y cinco trabajos. Por otro lado, se muestran los tiempos de proceso de cada uno de los trabajos en cada máquina. Se asume la secuencia π = (4,2,5,1,3) y el resultado del diagrama de Gantt es el mostrado en la Figura 4. Cabe resaltar como aparecen tiempos ociosos en algunas de las máquinas, considerando tiempo ocioso o *idle time* al tiempo que transcurre en una máquina desde que termina de procesar un trabajo y comienza a realizar el siguiente. Por ejemplo, en la máquina tres, el trabajo cuatro deja de procesarse en el instante cuarenta y cuatro y el trabajo dos no comienza hasta el instante ochenta y cuatro, debido al retraso producido por la máquina dos. Por lo tanto, el tiempo ocioso en la máquina tres entre los trabajos cuatro y dos sería de cuarenta. También es interesante observar como en la primera máquina el tiempo ocioso total es cero, ya que no depende de ninguna máquina anterior.

A continuación, se muestra a notación α|β|γ del problema y se describen cada una de las restricciones que posee. Además, en la

Tabla 2 se muestra la notación empleada en el problema.

$$Fm|prmu|\sum_{i=0}^{m} w_i * \theta_i$$

- Entorno (α=*Fm*), se trata de un entorno de taller de flujo regular, el cual está formado por un conjunto finito de *m* máquinas en serie donde se procesan n trabajos siguiendo sigue la misma ruta (*Rj*).
- Restricciones (*β=prmu*), donde se expone un caso particular de *flowshop* donde en cada máquina se sigue la misma secuencia, es decir, siempre se ejecutan y procesan los trabajos en el mismo orden en cada máquina.
- Objetivo ($\gamma=\sum_{i=0}^{m} w_i * \theta_i$). El objetivo consiste en la minimización del sumatorio del tiempo ocioso total de todas las máquinas o *core idle time.* En la actualidad, existen diferentes definiciones del tiempo ocioso, en función si se tienen en cuenta el tiempo previo al procesamiento del primer trabajo (*front delay*) o si se tienen en cuenta los tiempos de finalización de cada trabajo en cada máquina (*back delay*) con el *makespan*.

  A la hora de estudiar el objetivo, se calcula teniendo en cuenta los pesos o costes (*wi*) asociados a cada máquina, tratando de minimizar en todo momento el sumatorio de tiempo ocioso asociado a cada una de ellas multiplicado por el coste de esta.

$$min\ \gamma\ =\ \sum_{i=1}^{m} w_i * \theta_i$$

  En este trabajo también se va a considerar el objetivo sin tener en cuenta los pesos (o lo que es lo mismo, w_i=1 para todo *i*). La función objetivo sería la siguiente:

$$min\,\gamma = \sum_{i=1}^{m} \theta_i$$

Tabla 2. Notación del problema

Notación	Descripción
n	Número de trabajos
m	Número de máquinas
j	Índice para los trabajos
i	Índice para las máquinas
pij	Processing Time o tiempo de procesamiento de cada trabajo *j* en cada máquina *i*
Cij	Tiempo de finalización de cada trabajo *j* en cada máquina *i*
wi	Peso o coste asociado a la máquina *i*
Θi	Tiempo ocioso en la máquina *i*
TFTi	*Total flowtime* o sumatorio de tiempo de finalización en cada máquina *i*

Para simplificar el problema a estudiar, se tienen en cuenta las siguientes suposiciones generales:

- Los trabajos estarán disponibles al principio del horizonte de programación, es decir, se consideran que las fechas de llegada o *release dates* son cero.
- Los trabajos no se pueden interrumpir.
- Todas las máquinas estarán siempre disponibles, desde el primer momento.
- Cada máquina puede hacer un trabajo, y un trabajo solo puede ser procesado por una máquina a la vez.
- El buffer entre máquinas se supone infinito.
- El tiempo de transporte de una máquina a otra se considera despreciable.
- Todas las máquinas tienen la misma velocidad.
- Los tiempos de set-up se incluyen en el tiempo de cada trabajo en cada máquina.
- No se tendrán en cuenta las fechas de entrega.

3.2 Métodos propuestos

En este subcapítulo se explican los algoritmos propuestos en este trabajo. Como se ha mencionado anteriormente, la función objetivo consiste en la minimización del consumo energético de las máquinas. Para ello se plantean diferentes variantes de la NEH, teniendo en cuenta diversas reglas de despacho y técnicas de desempate.

3.2.1 NEH

El estudio de entornos de taller de flujo regular o *flowshop* es uno de los problemas más frecuentes en programación de la producción. Los problemas en entornos *flowshop* pueden ser abordados y estudiados con diversos métodos como, por ejemplo, el algoritmo desarrollado por Nawaz, Enscore y Ham. Dicho algoritmo, también conocido como NEH, fue desarrollado en 1983 por Nawaz, Enscore y Ham con la idea de minimizar el *makespan* (Nawaz, Enscore, and Ham 1983). Este método ha sido ampliamente reconocido por su alto reconocimiento, flexibilidad y eficiencia (Liu, Jin, and Price 2017). Además, NEH se utiliza como solución inicial para técnicas metaheurísticas como, por ejemplo, el recocido simulado, algoritmo genético, etc., ya que para determinados problemas, principalmente de *flowshop*, proporciona soluciones muy buenas en poco

tiempo de cómputo.

La heurística NEH se plantea inicialmente para resolver el problema clásico de programación del taller de flujo regular de permutación con la minimización del *makespan* o tiempo máximo de finalización de los trabajos como función objetivo. Esta heurística ordena los trabajos según un índice de importancia y luego determina trabajo por trabajo su mejor posición mientras mantiene la secuencia relativa de los trabajos ya programados (Jose M. Framinan, Leisten, and Ruiz García 2014). El procedimiento de la NEH, tal y como se indica en el pseudocódigo de la Figura 5, sigue los siguientes pasos:

1. Se calculan los tiempos de proceso totales de cada trabajo en todas las máquinas.
2. Se ordenan según la regla LPT o *largest processing time*, es decir, se ordenan los trabajos de mayor a menor tiempo de proceso total en todas las máquinas.
3. Una vez ordenados, se genera una secuencia inicial con la que comenzará el algoritmo. Seguidamente, se establece una solución parcial, formada por el primer trabajo de la secuencia inicial. El resto de los trabajos pertenecen a un conjunto de trabajos aún no programados, siguiendo el orden original de la secuencia inicial.
4. A continuación, se itera. Cada trabajo del conjunto de trabajos aún no programados se introduce en todas las posiciones posibles de la secuencia parcial, evaluando la función objetivo en cada iteración y seleccionando aquella secuencia que minimice la función objetivo. Además, el orden en el que se van eligiendo los trabajos está determinado por la secuencia inicial y una vez alcanzado el mejor valor de función objetivo en una inserción no se alteran las posiciones de los trabajos de la anterior subsecuencia.
5. La solución final proporcionada por el NEH se dará por finalizada cuando el último trabajo se haya insertado, obteniendo el menor valor de la función objetivo.

NEH Heuristic

$P_i = Calculate\ total\ processing\ time\ of\ each\ job\ i\ \left(\sum_{j=1}^{m} p_{ij}\right)$
$\psi = Sort\ P_i\ in\ decreasing\ order$
$\pi_1 = \psi_1$
$for\ i = 2\ to\ n\ do$
  $\pi = Insert\ job\ \psi_i\ into\ best\ position\ of\ \pi\ (\pi, \psi_i)$
$end\ for$
$return\ \pi\ and\ f(\pi)$

Figura 5. Pseudocódigo heurística NEH (Öztop et al. 2020)

3.2.2 Modificaciones de la NEH

La modificación de la NEH propuesta se basa en el artículo (Öztop et al. 2020), donde se estudian dos objetivos contradictorios: el *total flowtime* y la energía total consumida, la cual se calcula en función de la potencia asociada a cada máquina, los niveles de velocidad asociados a cada una de ellas y sus *processing time*s. Sin embargo, en este trabajo solo se tendrá en cuenta una función objetivo, tal y como se ha mencionado en apartados anteriores. La forma en la que se medirá el consumo energético será mediante los pesos o costes asociados a cada una de las máquinas. Es importante resaltar que se considera tiempo ocioso al tiempo que una máquina permanece en espera o estado de *standby* desde que termina de procesar un trabajo hasta que comienza a procesar el siguiente. Además, por simplificación se supondrán las mismas velocidades para cada máquina, a diferencia del artículo (Öztop et al. 2020).

Así, siguiendo la idea del artículo (Öztop et al. 2020), en este trabajo se va a usar el mismo método de resolución planteado por los autores de dicho artículo, la NEH, modificándolo de acuerdo con las consideraciones que se han hecho para el problema. Debido a la importancia que cobra el orden de la secuencia inicial proporcionada al algoritmo, se irán probando diversas reglas de despacho o *priority rules* (PR), ya que la secuencia inicial empleada resulta muy significativo a la hora de medir la calidad de las soluciones obtenidas. Además, se utilizará una regla de desempate o *tie breaking rule* (TBR), basada en el

total flowtime, en caso de empate de soluciones en cada una de las inserciones del método NEH. Resulta una regla de desempate interesante porque, para secuencias que impliquen el mismo consumo energético, se seleccionan aquellas que minimicen el tiempo medio que los trabajos están en el sistema (indicador relacionado con la satisfacción de los clientes).

En el estudio (J. M. Framinan, Leisten, and Rajendran 2003) se proponen 177 diferentes órdenes iniciales para la NEH considerando las siguientes funciones objetivos de minimización: *makespan*, *idletime* y *flowtime*. Por lo tanto, debido a la alta influencia que ocasiona la regla de despacho elegida para generar la secuencia inicial del algoritmo NEH, se procede a presentar diferentes reglas para la función objetivo de minimización de tiempo ocioso.

En primer lugar, se emplea la regla *largest processing time* (LPT), la cual es la regla habitual que se emplea en dicho algoritmo cuando la función objetivo consiste en la minimización del *makespan*. Sin embargo, debido a que la función objetivo estudiada consiste en la minimización de los tiempos ociosos y costes asociados a cada máquina, además de utilizar la regla LPT se probarán diferentes variantes de la NEH usando las siguientes propuestas:

- PR A: Las reglas de despacho empleadas se basan en analizar y ordenar los trabajos en función de los costes asociados a cada máquina, ya que es decisivo en la función objetivo al ser un factor que multiplica. Para ello, se tendrá en cuenta el 50% de los pesos más altos y se ordenarán los trabajos en función de la suma de sus *processing times* exclusivamente en esas máquinas. Los pasos a seguir son los siguientes:
    - Primero, se localizan las máquinas con los costes más elevados, en el caso de PR A, solo de los 50% más altos.
    - Posteriormente, se suman los tiempos de proceso de cada trabajo en el 50% de las máquinas identificadas en el paso anterior.
    - A continuación, se ordenan los trabajos en orden descendente en función de la suma obtenida en el apartado anterior.
    - Finalmente, se genera la secuencia inicial que será con la que comenzará el algoritmo NEH.
- PR B: Consiste en realizar los mismos pasos ejecutados con la PR A, pero en este caso se escogerán el 80% de las máquinas con mayor coste.
- PR C: En este caso se consideran el 20% de las máquinas con pesos más grandes.

Los procedimientos aproximados más eficientes para los problemas de programación de *flowshop* están basados en la construcción de una secuencia insertando iterativamente, uno por uno, los trabajos no programados en todas las posiciones de una subsecuencia existente, y luego, seleccionando aquella con menor valor de la función objetivo. Este tipo de procedimiento conlleva a numerosos empates, por lo que es necesario aplicar una regla de desempate o *tie-breaking rule* (Fernandez-Viagas and Framinan 2014). En este trabajo, esto ocurre en el método seleccionado para resolver el problema planteado, la NEH, como se ha explicado anteriormente. En el artículo (Fernandez-Viagas and Framinan 2014), se propone una regla de desempate basada en los tiempos ociosos de las máquinas. Sin embargo, en este trabajo se propone una regla de desempate basada en la minimización del *total flowtime.* Los pasos a seguir son los siguientes:

- Una vez generada la secuencia inicial con la que comenzará el algoritmo, se establece una solución parcial, formada por el primer trabajo de la secuencia inicial. El resto de los trabajos pertenecen a un conjunto de trabajos aún no programados, siguiendo el orden original de la secuencia inicial.
- A continuación, se itera. Cada trabajo del conjunto de trabajos aún no programados, se introducen en todas las posiciones posibles de la secuencia parcial, evaluando la función objetivo en cada iteración y seleccionando aquella secuencia que minimice dicha función objetivo. En caso de empate, se mide el *total flowtime* de cada secuencia y elige el que menor obtenga. Además, el orden en el que se van eligiendo los trabajos está determinado por la secuencia inicial y una vez alcanzado el mejor valor de función objetivo en una inserción no se alteran las posiciones de los trabajos de la anterior subsecuencia.

- La solución final proporcionada por el NEH se dará por finalizada cuando el último trabajo se haya insertado, obteniendo el menor valor de la función objetivo.

A continuación, en la Tabla 3, se muestra un resumen de cada uno de los métodos planteados en este trabajo.

Tabla 3.Resumen de cada variante de la NEH, la solución inicial empleada en cada caso y regla de desempate.

Heurística	Solución inicial	Regla de desempate	Notación
NEH	LPT	-	NEH
NEH	PR A	-	NEH (PR A)
NEH	PR B	-	NEH (PR B)
NEH	PR C	-	NEH (PR C)
NEH	LPT	TB	NEH + TB
NEH	PR A	TB	NEH (PR A) + TB
NEH	PR B	TB	NEH (PR B) + TB
NEH	PR C	TB	NEH (PR C) + TB

4 EXPERIMENTACIÓN

En este capítulo se explica detalladamente las herramientas utilizadas para el desarrollo de los algoritmos, así como el software empleado, las instancias y problemas analizados.

4.1 Herramientas utilizadas

Para codificar los algoritmos propuestos se ha usado *Codebloks,* en su versión 17.12. Este programa es un *software* libre y un entorno de desarrollo integrado de código abierto que soporta múltiples compiladores, entre ellos *C.* El lenguaje elegido para la programación de los algoritmos ha sido el lenguaje C, por ser un lenguaje básico, universal y sencillo. Además, se ha contado con la librería *Schedule*, desarrollada por un profesor de la Escuela Técnica Superior de Ingenieros de Sevilla, que facilita la programación de métodos de resolución de problemas de programación de la producción en lenguaje C. Los motivos principales por los que se ha utilizado la herramienta *Codeblocks* ha sido por:

- Posee un espacio de trabajo adaptable
- Su sistema de construcción (*build*) es rápido
- Se puede hacer uso de las librerías que ya contiene.

Debido al elevado número de datos y soluciones que se manejan en este trabajo se utiliza la herramienta de *Microsoft Excel*. Con ella se comparan soluciones, se añaden gráficos, tablas, etc.

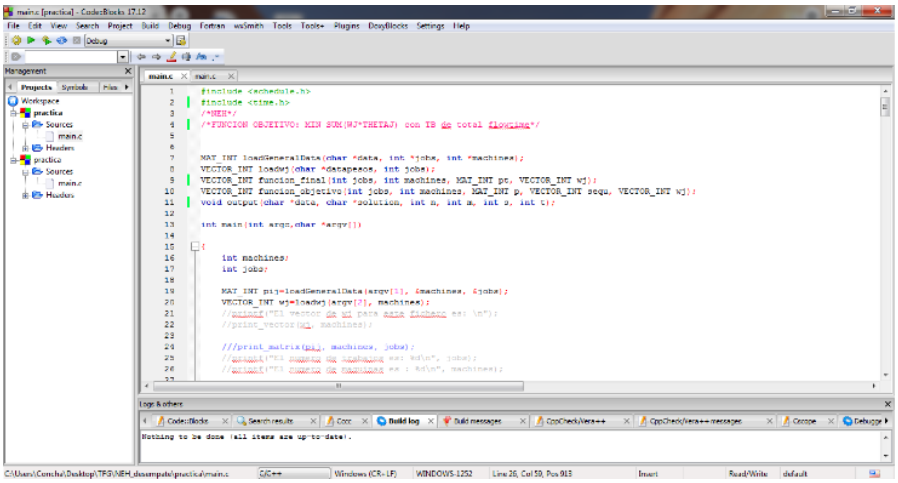

Ilustración 1. Visualización de pantalla de *Codeblocks* 17.12.

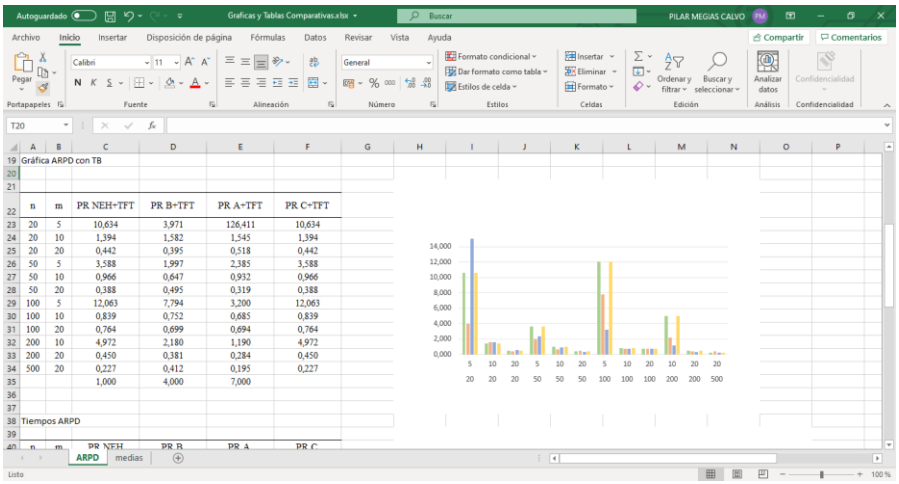

Ilustración 2. Visualización de pantalla de *Microsoft Excel* para el análisis de datos.

Además, resulta conveniente resaltar las características del equipo empleado para los experimentos:

- Ordenador de uso personal con procesador INTEL CORE i5- CPU, 2.50GHz y RAM.
- El sistema operativo empleado será *Microsoft Windows 7 Home*.

4.2 Instancias del problema

Para la resolución del problema planteado usando el método propuesto se ha empleado la famosa batería de instancias de *Taillard,* creadas por Éric Taillard en 1993 (Taillard 1993). Se agrupan en 12 conjuntos de 10 instancias cada una, tal y como se muestra en la Tabla 4.

Tabla 4. Instancias de Taillard (Taillard 1993)

Trabajos	Máquinas
20	5
20	10
20	20
50	5
50	10
50	20
100	5
100	10
100	20
200	10
200	20
500	20

Estas instancias muestran los tiempos de proceso de cada uno de los diferentes problemas, dándoles un valor generado con una distribución uniforme (un ejemplo se muestra en la Figura 6). Además, los pesos asociados a cada una de las máquinas también han sido generados siguiendo el modelo de instancias de Taillard, es decir, han sido generados de manera aleatoria, obteniendo un valor comprendido entre 0 y 99, siguiendo una distribución uniforme.

```
 5
20
54        79        16        66        58
83         3        89        58        56
15        11        49        31        20
71        99        15        68        85
77        56        89        78        53
36        70        45        91        35
53        99        60        13        53
38        60        23        59        41
27         5        57        49        69
87        56        64        85        13
76         3         7        85        86
91        61         1         9        72
14        73        63        39         8
29        75        41        41        49
12        47        63        56        47
77        14        47        40        87
32        21        26        54        58
87        86        75        77        18
68         5        77        51        68
94        77        40        31        28
```

Figura 6. Instancia de Taillard número uno (TA001) con 5 máquinas y 20 trabajos.

Para ejecutar y obtener las diferentes soluciones proporcionadas por los algoritmos descritos con anterioridad, se debe crear una carpeta donde se encuentre el ejecutable de *Codeblocks* y todos los archivos proporcionados por Taillard. Posteriormente se crea un archivo tipo *batch* (.bat) que contiene en cada línea los argumentos que se deben pasar a *Codeblocks* para que se ejecute el código. Es decir, en total contendrá 120 líneas, una por cada instancia. Este archivo *batch* sirve para ejecutar por lotes todas las instancias desde la opción de símbolo de sistema de *Windows*, siendo una manera fácil y cómoda de obtener las soluciones, las cuales se generan y almacenan en un archivo de texto dentro de la misma carpeta.

4.3 Indicadores para la evaluación de las soluciones

Tal y como se ha mencionado con anterioridad, todas las soluciones se han exportado a un archivo de *Microsoft Excel* con el fin de trabajar mejor con los resultados, pudiendo profundizar más en la interpretación de los mismos.

Al igual que en (Öztop et al. 2020), (J. M. Framinan, Leisten, and Rajendran 2003) y (Fernandez-Viagas and Framinan 2014), con el fin de medir la calidad de las soluciones y los tiempos de ejecución, se proporcionan los ARPD y tiempos de CPU de cada una de las soluciones obtenidas:

- El ARPD o *average relative percentage deviation* mide la desviación relativa media de las soluciones obtenidas tras la ejecución de un algoritmo en comparación con la mejor solución encontrada. Es por ello, previamente se ha debido calcular los RPD o relative percentage deviation de cada solución:

$$RPD = \frac{OS - BS}{BS} * 100$$

  Siendo BS la mejor solución obtenida o *best solution* y OS será la solución que se quiere evaluar (*obtained solution*). Por lo tanto, una solución que sea la mejor de todo el conjunto tendrá un ARPD cuyo valor será 0.

  Finalmente, el ARPD se calculará como la media entre las *k* soluciones:

$$ARPD = \frac{1}{k} * \sum_{j=1}^{k} RPD_j$$

- El tiempo de la CPU mide en milisegundos el tiempo transcurrido desde que se inicia la ejecución del algoritmo hasta que finaliza. Para llevar a cabo una evaluación más complete y exhaustiva se han medido todos los tiempos de ejecución de cada una de las soluciones.

4.4 Resultados

A continuación, se muestran todos los resultados obtenidos tras la experimentación. En primer lugar, se compararán los resultados obtenidos en la NEH tras probar diferentes soluciones iniciales. Posteriormente, se mostrarán los resultados una vez se haya aplicado la regla de desempate y, finalmente, los resultados del problema sin pesos.

4.4.1 Comparativa NEH con diferentes soluciones iniciales

En primer lugar, se han comparado variantes de la NEH con diferentes secuencias iniciales, NEH (PR A), NEH (PR B) y NEH (PR C) con la NEH original (que tiene como secuencia inicial la obtenida con la regla de despacho LPT). Como ya se ha mencionado con anterioridad, resulta muy significativa la secuencia inicial empleada en dicho algoritmo, ya que condiciona notablemente el resultado obtenido. Para ello se han calculado los ARPD y tiempos de CPU, mostrados a continuación en la Tabla 5, Tabla 6, Figura 7 Figura 8 respectivamente.

Tabla 5. ARPD de las variantes según la secuencia inicial.

n	m	NEH	NEH (PR B)	NEH (PR A)	NEH (PR C)
20	5	3,506	4,002	44,909	3,961
20	10	1,394	1,607	1,534	1,125
20	20	0,485	0,441	0,569	0,788
50	5	3,542	2,103	2,365	2,054
50	10	0,991	0,632	0,826	1,200
50	20	0,406	0,496	0,261	0,328
100	5	12,673	11,646	2,493	1,464
100	10	0,766	0,704	0,721	0,644
100	20	0,736	0,654	0,677	0,612
200	10	5,394	1,530	1,677	1,207
200	20	7,606	0,295	0,285	0,446
500	20	0,272	0,243	0,248	0,13
Promedio		3,148	2,029	4,714	1,163

Tabla 6. Tiempos de ejecución las variantes según la secuencia inicial.

n	m	NEH	NEH (PR B)	NEH (PR A)	NEH (PR C)
20	5	0,001	0,001	0,001	0,001
20	10	0,001	0,001	0,001	0,001
20	20	0,002	0,002	0,002	0,002
50	5	0,007	0,007	0,007	0,007
50	10	0,013	0,013	0,013	0,013
50	20	0,023	0,023	0,023	0,025
100	5	0,040	0,041	0,041	0,041
100	10	0,085	0,087	0,087	0,092
100	20	0,165	0,169	0,172	0,171
200	10	0,585	0,599	0,610	0,621
200	20	1,183	1,210	1,233	1,264
500	20	16,908	17,243	17,589	18,171
Promedio		1,584	1,616	1,648	1,701

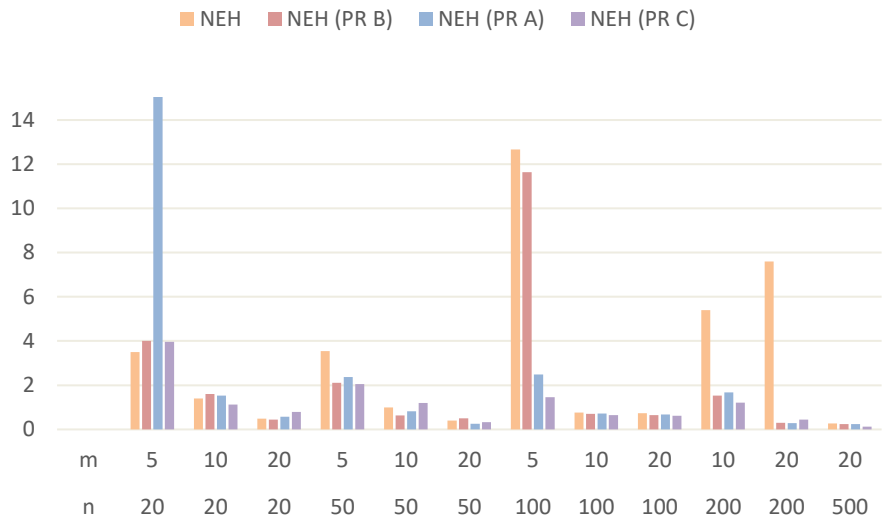

Figura 7. Representación de los ARPD de las variantes según la secuencia inicial.

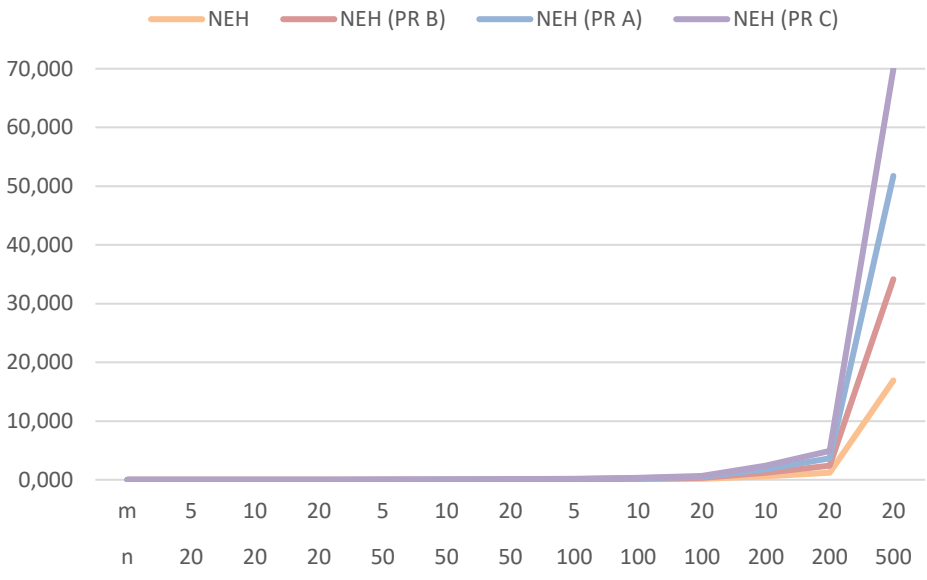

Figura 8 . Representación de los tiempos de ejecución de las variantes según la secuencia inicial.

Se puede observar en la Figura 7 y Tabla 5 como la variante de la NEH con la secuencia inicial PR C, la cual cuenta solo el 20% de máquinas cuyos costes son más elevados, alcanza el menor valor promedio de ARPD que el resto, siendo bastante uniforme independientemente de la dimensión del problema. Sin embargo, a pesar de mostrar el mejor resultado en el 58% de los casos, presenta un mayor tiempo de ejecución, siendo su promedio 1,7 segundos.

Por otro lado, las variantes de la NEH con las secuencias iniciales PR A y PR B, que tienen en cuenta el 50% y 80% de las máquinas con mayores pesos respectivamente, supone cada una el 16,67 % de las mejores soluciones obtenidas. En este caso en particular existe una clara diferencia entre ambos casos, ya que NEH (PR B) cuenta con menores tiempos de CPU y además obtiene mejores valores que NEH (PR A) en el 67% de los casos y un valor promedio de ARPD menor. Al mismo tiempo, para problemas con grandes dimensiones, es decir, los que cuentan con más de 200 trabajos, ofrece resultados de mayor calidad que el NEH (PR C).

Finalmente, cabe resaltar que los picos más visibles de la Figura 7 se observan en casos donde el número de máquinas es muy pequeño y la aleatoriedad de los pesos puede suponer un cambio significativo en la función objetivo. En la Tabla 7, se muestran los valores medios de la función objetivo para cada uno de los casos analizados anteriormente.

Tabla 7. Representación de los valores medios de la función objetivo de las variantes según la secuencia inicial.

n	m	NEH	NEH (PR B)	NEH (PR A)	NEH (PR C)
20	5	6280,7	6973,2	7345,4	7858,6
20	10	45673,7	47066,6	49868,9	49171,7
20	20	206406,1	206242,7	206427	213192,4
50	5	17939,2	18692,7	18452,9	18985,9
50	10	58101,6	64906,3	65227,7	70958,8
50	20	306195,2	299499,9	306799,7	322283,9
100	5	13331,5	16187,2	17512,9	23422,8
100	10	124933,7	131514,7	132603,1	144831,9
100	20	416817,2	423572,6	444352	473862,9
200	10	115431,2	129529,6	144346,7	164037,2
200	20	512395,7	612835,4	655960,2	676974,4
500	20	720860,4	816923,4	888595,5	1018740,1
Promedio		212030,5	231162,0	244791,0	265360,1

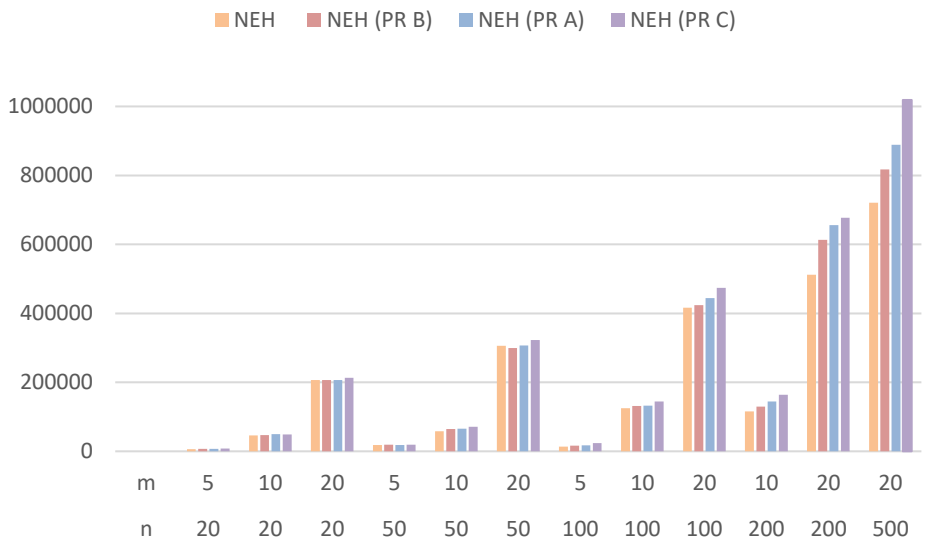

Figura 9 . Representación de los valores medios de la función objetivo de las variantes según la secuencia inicial.

Se observa en la Tabla 7 y Figura 9 como los resultados medios de la función objetivo son menores para el caso NEH y NEH (PR B), donde el número de máquinas significativas elegidas para generar la secuencia inicial es mayor, el 100% y 80% respectivamente. Esto quiere decir que NEH (PR C) consigue los valores mínimos, pero en media obtiene peores valores del objetivo. Para casos donde el número de trabajos es menor a 200 resulta más conveniente utilizar la NEH con la secuencia inicial PR B, ya que los valores son muy semejantes a los obtenidos con la regla de despacho LPT y el promedio de sus ARPD es menor. Sin embargo, para instancias con más de 200 trabajos sigue siendo más ventajoso elegir NEH con la secuencia inicial obtenida con la regla de despacho LPT.

4.4.2 Comparativa NEH con regla de desempate

En este caso se ejecutará la NEH con cada una de las *priority rule* ya definidas e incluyendo la *tie breaking rule* (TB), o regla de desempate, de *total flowtime* empleada en cada uno de los casos.

Tabla 8 . ARPD de cada variante incluyendo regla de desempate

n	m	NEH+TB	NEH (PR B) + TB	NEH (PR A) + TB	NEH (PR C) + TB
20	5	10,634	3,971	16,411	10,634
20	10	1,394	1,582	1,545	1,394
20	20	0,442	0,395	0,518	0,442
50	5	3,588	1,997	2,385	3,588
50	10	0,966	0,647	0,932	0,966
50	20	0,388	0,495	0,319	0,388
100	5	12,063	7,794	3,200	12,063
100	10	0,839	0,752	0,685	0,839
100	20	0,764	0,699	0,694	0,764
200	10	4,972	2,180	1,190	4,972
200	20	0,450	0,381	0,284	0,450
500	20	0,227	0,412	0,195	0,227
Promedio		3,061	1,775	2,363	3,061

Tabla 9 . Tiempos de ejecución de cada variante incluyendo regla de desempate.

n	m	NEH+ TB	NEH (PR B) + TB	NEH (PR A) + TB	NEH (PR C) + TB
20	5	0,001	0,001	0,001	0,001
20	10	0,001	0,001	0,001	0,002
20	20	0,002	0,002	0,002	0,002
50	5	0,011	0,010	0,010	0,012
50	10	0,016	0,017	0,017	0,016
50	20	0,026	0,025	0,025	0,026
100	5	0,076	0,076	0,080	0,075
100	10	0,124	0,127	0,127	0,124
100	20	0,195	0,193	0,194	0,194
200	10	0,897	0,930	0,995	0,902
200	20	1,410	1,458	1,467	1,418
500	20	21,166	21,772	22,901	21,298
Promedio		1,994	2,051	2,152	2,006

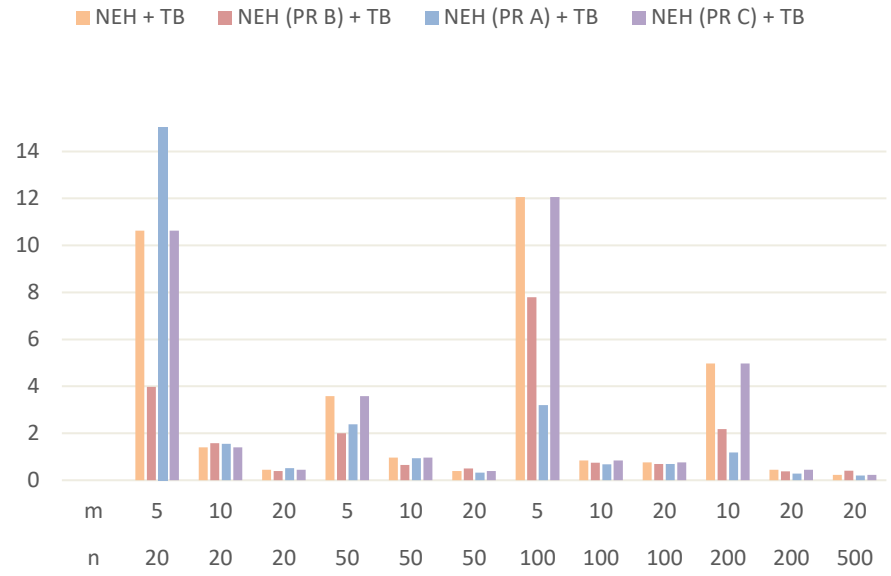

Figura 10 . Representación de ARPD de cada variante según la secuencia inicial y regla de desempate.

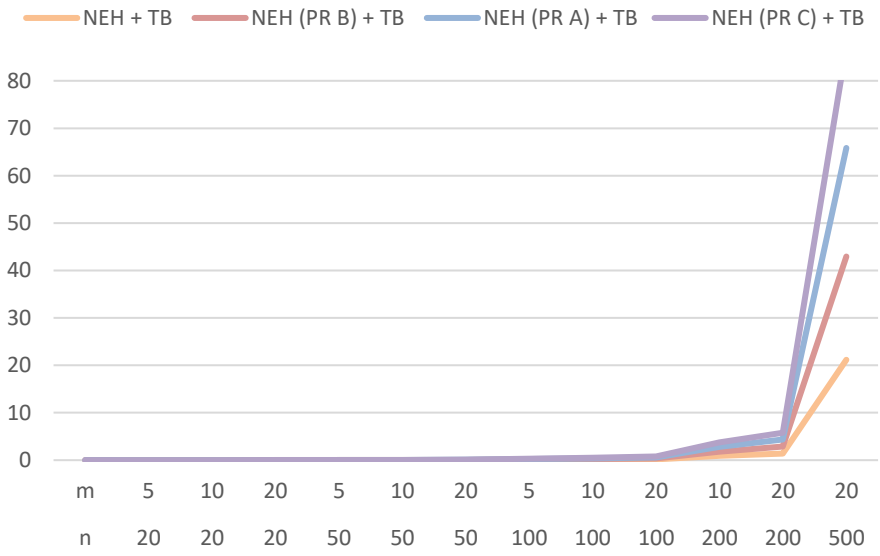

Figura 11. Tiempos de ejecución de cada variante según la secuencia inicial y regla de desempate.

En este análisis se observan tiempos de ejecución mayores, ya que aplicar la regla de desempate implica mayor tiempo de CPU, siendo el caso de NEH (PR A) + TB el que mayor tiempo emplea. A diferencia del caso anterior, donde no se empleaba ninguna técnica de desempate, el mejor de todos los métodos es el NEH (PR B) + TB, alcanzando el mejor resultado en más del 55% de los casos estudiados y con menor valor medio de ARPD. Le precede el NEH (PR A) + TB, que abarca el 33% de las mejores soluciones del conjunto total de problemas. En este caso, resulta más interesante emplear NEH (PR A) + TB en problemas con dimensiones mayores (es decir, instancias con más de 200 trabajos) y NEH (PR B) +TB en problemas de dimensiones menores. Por otro lado, para los problemas de dimensiones elevadas destaca NEH+TB.

Si analizamos los valores medios de la función objetivo para los casos en los que se emplea cada *priority rule* con la regla de desempate de *total flowtime* se obtienen los resultados mostrados en Tabla 10 y Figura 12.

Tabla 10. Representación de los valores medios de la función objetivo para cada variante según la secuencia inicial y regla de desempate.

n	m	NEH +TB	NEH (PR B) +TB	NEH (PR A) + TB	NEH (PR C) + TB
20	5	6142,9	6929,9	7135,0	6142,9
20	10	45679,7	46622,0	50089,2	45679,7
20	20	200415,1	199736,2	199716,3	200415,1
50	5	18122,2	18052,7	18562,8	18122,2
50	10	59411,1	65528,6	66983,9	59411,1
50	20	302345,2	299298,3	304242,2	302345,2
100	5	13207,0	15380,0	16876,6	13207,0
100	10	122798,6	128140,5	129376,4	122798,6
100	20	413574,9	422315,6	449003,4	413574,9
200	10	110432,3	121417,6	142715,1	110432,3
200	20	552315,3	597978,1	636118,0	552315,3
500	20	714276,8	814803,8	877841,6	714276,8
Promedio		213226,8	228016,9	241555,0	213226,8

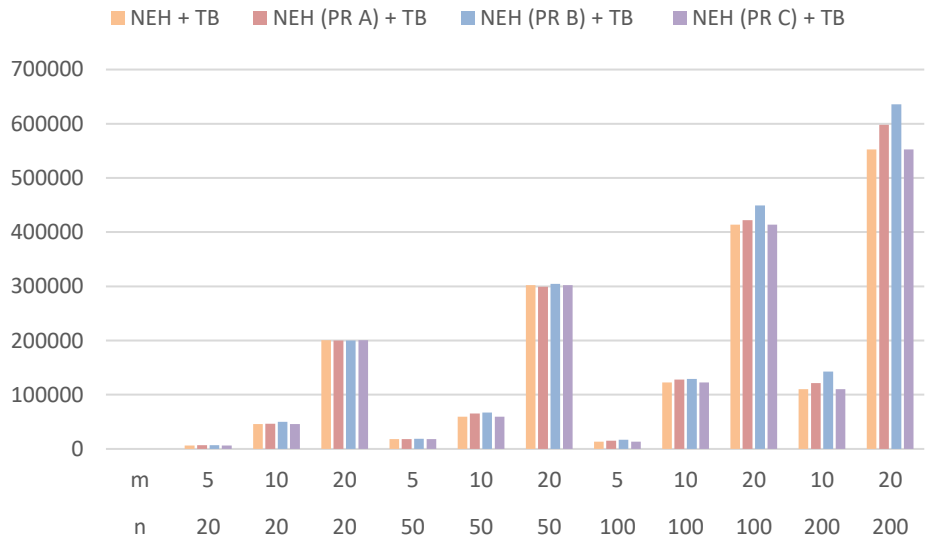

Figura 12. Representación de cada valor medio de la función objetivo para cada variante según secuencia inicial y regla de desempate.

En este caso se obtienen los mejores valores medios para el caso de NEH+TB y NEH (PR C) + TB en el 83.33% de los casos. Entre ambos métodos resulta más interesante aplicar NEH+TB ya que su tiempo de CPU es menor. Sin embargo, para instancias de dimensiones reducidas resulta conveniente usar NEH (PR B) +TB ya que sus valores de ARPD son menores y obtiene valores medios semejantes al caso de NEH+TB.

4.4.3 Comparativa total

En este apartado se estudia la influencia de la regla de desempate en el problema. Tal y como se observa en la Tabla 11, la regla de desempate resulta ventajosa en cada uno de los casos, obteniendo siempre valores de ARPD promedio menores en el caso de aplicar la regla de desempate que en el caso donde no se aplica. Sin embargo, en el caso de NEH y NEH + TB se obtienen el mismo valor promedio de ARPD, por lo tanto, resulta más conveniente en este caso no aplicar la regla de desempate ya que se alcanzan los mismos resultados, pero con un tiempo de ejecución mayor. Además, en la Tabla 12 se observan los valores medios de la función objetivo obtenidos en cada una de las variantes de la NEH según la secuencia inicial empleada y regla de desempate, siendo en todos los casos inferior si se aplica la regla de desempate, a excepción del caso NEH +TB, donde no se obtiene mejor valor promedio de la función objetivo si se aplica.

Tabla 11. ARPD de cada variante incluyendo regla de desempate y sin incluir regla de desempate.

n	m	ARPD NEH	ARPD NEH (PRA)	ARPD NEH (PRB)	ARPD NEH (PRC)	ARPD NEH+ TB	ARPD NEH (PRA) + TB	ARPD NEH (PRB) + TB	ARPD NEH (PRC) + TB
20	5	10,895	130,168	4,002	13,884	10,634	126,411	3,971	10,634
20	10	1,394	1,534	1,607	1,577	1,394	1,545	1,582	1,394
20	20	0,485	0,569	0,441	0,788	0,480	0,569	0,440	0,725
50	5	3,542	2,365	2,103	3,807	3,588	2,385	1,997	3,588
50	10	0,991	0,881	0,632	1,348	1,035	0,932	0,647	0,966
50	20	0,406	0,330	0,496	0,479	0,388	0,319	0,495	0,388
100	5	12,673	3,359	11,646	22,168	12,546	3,200	11,016	12,063
100	10	0,871	0,727	0,798	1,169	0,839	0,685	0,752	0,839
100	20	0,777	0,677	0,704	1,021	0,764	0,694	0,699	0,764
200	10	5,394	1,677	2,393	7,871	5,117	1,647	2,180	4,972
200	20	7,606	0,324	0,416	0,777	8,276	0,284	0,381	0,450
500	20	0,272	0,248	0,415	0,750	0,260	0,233	0,412	0,227
Promedio		3,775	11,905	2,138	4,637	3,777	11,575	2,048	3,084

Tabla 12. Representación de los valores medios de la función objetivo para cada variante según la secuencia inicial con regla de desempate y sin regla de desempate.

n	m	NEH	NEH (PRB)	NEH (PRA)	NEH (PRC)	NEH +TB	NEH (PRB) +TB	NEH (PRA) + TB	NEH (PRC) + TB
20	5	6280,7	6973,2	7345,4	7858,6	6142,9	6929,9	7135	6142,9
20	10	45673,7	47066,6	49868,9	49171,7	45679,7	46622	50089,2	45679,7
20	20	206406,1	206242,7	206427	213192,4	200415,1	199736,2	199716,3	200415,1
50	5	17939,2	18692,7	18452,9	18985,9	18122,2	18052,7	18562,8	18122,2
50	10	58101,6	64906,3	65227,7	70958,8	59411,1	65528,6	66983,9	59411,1
50	20	306195,2	299499,9	306799,7	322283,9	302345,2	299298,3	304242,2	302345,2
100	5	13331,5	16187,2	17512,9	23422,8	13207	15380	16876,6	13207
100	10	124933,7	131514,7	132603,1	144831,9	122798,6	128140,5	129376,4	122798,6
100	20	416817,2	423572,6	444352	473862,9	413574,9	422315,6	449003,4	413574,9
200	10	115431,2	129529,6	144346,7	164037,2	110432,3	121417,6	142715,1	110432,3
200	20	512395,7	612835,4	655960,2	676974,4	552315,3	597978,1	636118	552315,3
500	20	720860,4	816923,4	888595,5	1018740,1	714276,8	814803,8	877841,6	714276,8

Promedio	212030,5	231162	244791,0	265360,1	213226,8	228016,9	241555,0	213226,8

4.4.4 Problema sin pesos

A pesar de que el objetivo que persigue dicho estudio consiste en la reducción del tiempo ocioso y el coste asociado del mismo a las máquinas de una planta de producción, en este apartado se estudia el impacto que supone añadir dichos costes a la función objetivo de minimización. Como se ha comentado anteriormente, el tiempo que una máquina se mantiene en estado de *standby* mientras espera al siguiente trabajo que debe procesar se genera un coste y consumo energético que se debe minimizar, por su importante consecuencia medio ambiental y económica. El motivo por el que se decide estudiar este caso sin tener en cuenta los costes son para ver que de forma influyen en la función objetivo los pesos asociados a cada máquina, la calidad de las soluciones obtenidas, el factor aleatorio de los mismos, etc.

Tabla 13. Representación de cada ARPD del algoritmo NEH para el problema sin pesos.

n	m	NEH sin pesos	NEH sin pesos + TB
20	5	2,49	2,49
20	10	0,80	0,81
20	20	0,37	0,37
50	5	1,82	1,81
50	10	0,79	0,34
50	20	0,18	0,18
100	5	23,63	23,44
100	10	0,43	0,47
100	20	0,19	0,11
200	10	1,80	2,05
200	20	0,13	0,16
500	20	0,15	0,31
	Promedio	2,73	2,71

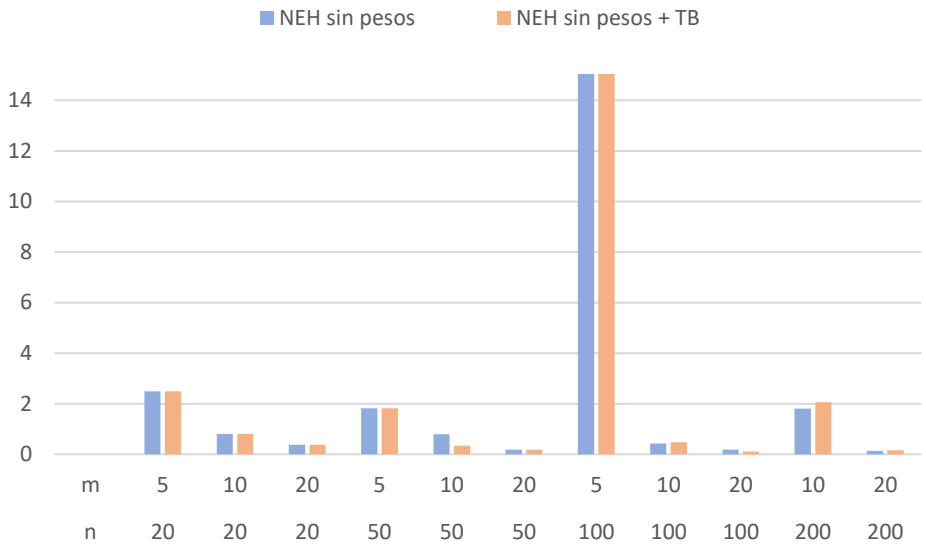

Figura 13. Representación de cada ARPD del algoritmo NEH sin pesos, con regla de desempate y sin regla de desempate.

Tabla 14. Tiempos de ejecución del algoritmo NEH sin pesos, con regla de desempate y sin regla de desempate.

n	m	NEH sin pesos	NEH sin pesos + TB
20	5	0,00	0,00
20	10	0,00	0,00
20	20	0,00	0,00
50	5	0,01	0,01
50	10	0,01	0,02
50	20	0,02	0,03
100	5	0,04	0,07
100	10	0,09	0,12
100	20	0,17	0,20
200	10	0,60	0,92
200	20	1,23	1,43
500	20	17,37	22,49
Promedio		1,63	2,11

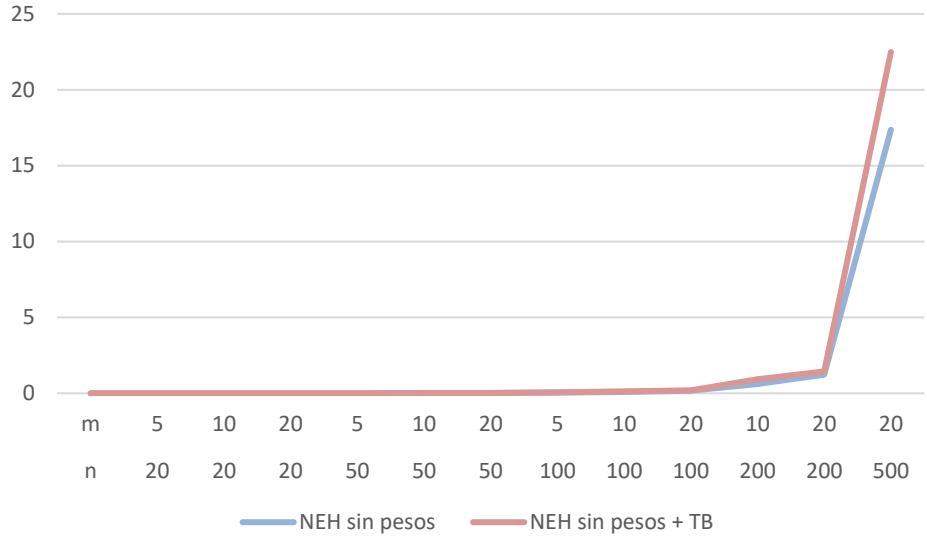

Figura 14. Tiempo de ejecución del algoritmo NEH sin pesos, con regla de desempate y sin regla de desempate.

En el caso en el que no se tienen en cuenta los pesos se obtienen ARPD menores debido a que al no contar con el factor aleatorio de los pesos las soluciones son más semejantes. Para dimensiones pequeñas resulta mejor opción el análisis sin regla de desempate y en los casos donde hay un elevado número de máquinas si es interesante emplearla, aunque en general es más interesante emplear la regla de desempate ya que el valor promedio de ARPD es menor en este caso. En cuanto a los tiempos de ejecución son bastante similares tal y como se observan en la Tabla 14. Por otro lado, si se analiza el valor medio de las soluciones obtenidas, tal y como se muestra en la Tabla 15 y Figura 15 se obtienen los siguientes resultados:

Tabla 15. Valores medios de la función objetivo con el algoritmo NEH sin pesos con regla de desempate y sin regla de desempate

n	m	NEH sin pesos	NEH sin pesos + TFT
20	5	125,6	125,6
20	10	917,2	919,5
20	20	4255,1	4248,8
50	5	423,1	422,1
50	10	1295,1	1331,2
50	20	6415,5	6404,1
100	5	320,2	317,7
100	10	2360,2	2333,0
100	20	8893,3	8942,4
200	10	2637,8	2571,9
200	20	11273,7	11128,4
500	20	15050,3	14769,9
Promedio		4497,3	4459,6

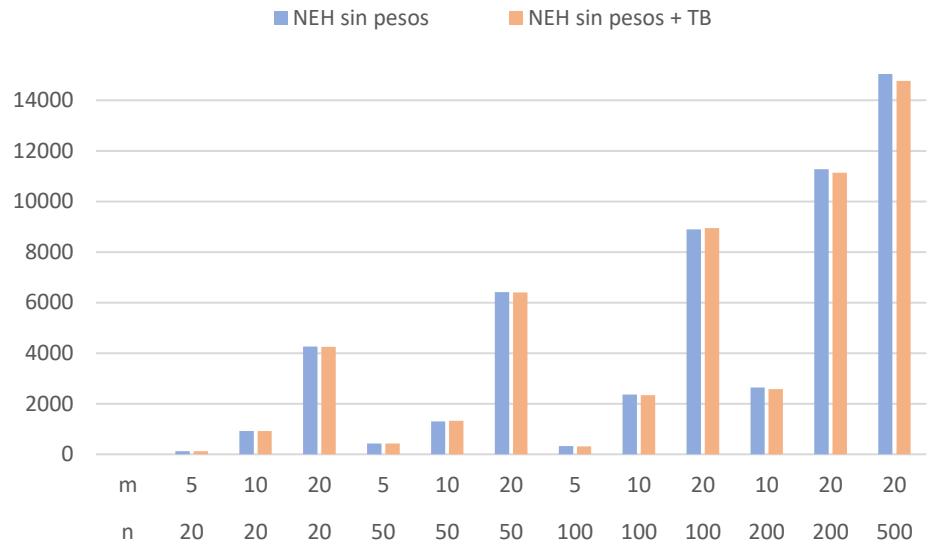

Figura 15. Representación de los valores medios de la función objetivo con el algortimo NEH sin pesos con regla de desempate y sin regla de desempate.

En cuanto a los valores medios obtenidos es más conveniente la aplicación de la técnica de desempate en aquellos problemas donde hay más de 200 trabajos, ya que reduce el valor de la función objetivo, el tiempo de ejecución es similar y los ARPD similares.

5 CONCLUSIONES

En este último capítulo se comentan las conclusiones finales tras la aplicación en un entorno de taller de flujo regular o *flowshop* de una modificación de la NEH y las diferentes propuestas de secuencias iniciales y técnicas de desempate. Para el estudio se han seguido los siguientes pasos:

- En primer lugar, se han definido los conceptos básicos en los que se basa la programación de operaciones. De esta forma se contextualiza el problema y se introduce el tema al lector, facilitándole en todo momento la comprensión del estudio de este trabajo fin de grado.
- Posteriormente, se han descrito las características por los que se rige el problema estudiado, así como las reglas de desempate y *priority rules* que se han empleado. Además, se explica la herramienta y *software* utilizado.
- A continuación, se analizan y expresan los resultados obtenidos en cada experimento.

Como conclusiones tras el análisis de los resultados obtenidos se puede añadir lo siguiente:

- NEH (PR C) es la que mejor valor de ARPD ofrece, debido a que tiene en cuenta un número muy bajo de máquinas para generar la secuencia inicial. Sin embargo, es la que mayor tiempo de ejecución emplea. En cuanto a los valores medios obtenidos el caso de NEH (PR B) es mucho mejor, teniendo además tiempos de CPU muy bajos.
- NEH (PR B) es la que mejor resultados ofrece de manera general. Es cierto que sus valores de ARPD no son lo más acertados en todos los casos, pero ofrece buenos tiempos de ejecución y sus valores medios son bastante aceptables.
- En caso de usar regla de desempate *total flowtime* resulta más interesante aplicar NEH (PR A) + TB. A continuación, le precede NEH (PR B) + TB, la cual también cuenta con buenos valores de ARPD y tiempos de ejecución. Sin embargo, no resulta conveniente emplearla para el caso de NEH (PR C) + TB ya que no se obtiene ningún tipo de mejora en la función objetivo.
- En el caso en el que no se tienen en cuenta los pesos, se obtienen ARPD menores debido a que al no contar con el factor aleatorio de los costes asociados a cada máquina las soluciones son mucho más semejantes. Para dimensiones pequeñas resulta mejor opción el análisis sin regla de desempate y en los casos donde hay un elevado número de máquinas si es interesante emplear la regla de *total flowtime*. En relación con el tiempo de ejecución son muy similares ambos casos, con regla de desempate de *total flowtime* y sin ella.

A modo resumen se puede afirmar:

- Para el estudio de un entorno tipo taller de permutación (*flowshop*) con función objetivo de minimización de tiempo ocioso total sin costes asociados a las mismas, el mejor resultado se obtiene con la aplicación del algoritmo NEH, *priority rule* LPT y regla de desempate de *total flowtime.*
- En caso de tener en cuenta los costes y consumo energético, el que ofrece mejores resultados a modo general es el algoritmo NEH, *priority rule* PR B y regla de desempate de *total flowtime*. Aunque no domine en todos los casos, ofrece generalmente buenos tiempos de ejecución, valores medios y ARPD.

ANEXO I: CÓDIGO

PR A

```
/*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
   int maquinas_significativas=((machines*5)/10);
   //printf("maquinas significativas: %d\n", maquinas_significativas);

   VECTOR_INT pesos_ordenados=DIM_VECTOR_INT(machines);
   VECTOR_INT wj_copia=DIM_VECTOR_INT(machines);
   copy_vector(wj, wj_copia, machines);
   sort_vector(wj_copia, pesos_ordenados, machines, 'D');
   //print_vector(pesos_ordenados, machines);

   VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
   int i=0;
   int j=0;
   int k=0;
   setval_vector(tiempos, jobs, 0);

   for(j=0; j<jobs; j++)
   {
      for(i=0; i<maquinas_significativas; i++)
      {
         tiempos[j]+=pt[pesos_ordenados[i]][j];
      }
   }

   //printf("\nEl vector de tiempos es:\n");
   //print_vector(tiempos,jobs);

   /*Genero secuencia inicial*/
   VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
   sort_vector(tiempos,sec_inicial_neh, jobs,'D');
```

```
//printf("\n La secuencia inicial para NEH es:\n");
//print_vector(sec_inicial_neh, jobs);
```

PR B

```
/*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
int maquinas_significativas=((machines*8)/10);
//printf("maquinas significativas: %d\n", maquinas_significativas);

VECTOR_INT pesos_ordenados=DIM_VECTOR_INT(machines);
VECTOR_INT wj_copia=DIM_VECTOR_INT(machines);
copy_vector(wj, wj_copia, machines);
sort_vector(wj_copia, pesos_ordenados, machines, 'D');
//print_vector(pesos_ordenados, machines);

VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
int i=0;
int j=0;
int k=0;
setval_vector(tiempos, jobs, 0);

for(j=0; j<jobs; j++)
{
   for(i=0; i<maquinas_significativas; i++)
   {
      tiempos[j]+=pt[pesos_ordenados[i]][j];
   }
}

///printf("\nEl vector de tiempos es:\n");
///print_vector(tiempos,jobs);

/*Genero secuencia inicial*/
VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
sort_vector(tiempos,sec_inicial_neh, jobs,'D');

//printf("\n La secuencia inicial para NEH es:\n");
//print_vector(sec_inicial_neh, jobs);
```

PR C

```
/*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
   int maquinas_significativas=((machines*2)/10);
   //printf("maquinas significativas: %d\n", maquinas_significativas);

   VECTOR_INT pesos_ordenados=DIM_VECTOR_INT(machines);
   VECTOR_INT wj_copia=DIM_VECTOR_INT(machines);
   copy_vector(wj, wj_copia, machines);
   sort_vector(wj_copia, pesos_ordenados, machines, 'D');
   //print_vector(pesos_ordenados, machines);

   VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
   int i=0;
   int j=0;
   int k=0;
   setval_vector(tiempos, jobs, 0);

   for(j=0; j<jobs; j++)
   {
      for(i=0; i<maquinas_significativas; i++)
      {
         tiempos[j]+=pt[pesos_ordenados[i]][j];
      }
   }

   ///printf("\nEl vector de tiempos es:\n");
   ///print_vector(tiempos,jobs);

   /*Genero secuencia inicial*/
   VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
   sort_vector(tiempos,sec_inicial_neh, jobs,'D');

   //printf("\n La secuencia inicial para NEH es:\n");
   //print_vector(sec_inicial_neh, jobs);
```

NEH

```
/*Algoritmo NEH con regla de despacho LPT y función objetivo minimización de tiempo ocioso**/
#include <schedule.h>
#include <time.h>

MAT_INT loadGeneralData(char *data, int *jobs, int *machines);
VECTOR_INT loadwj(char *datapesos, int jobs);
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt, VECTOR_INT wj);
int funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ, VECTOR_INT wj);
void output(char *data, char *solution, int n, int m, int s, int t);

int main(int argc,char *argv[])
{
   int machines;
   int jobs;

   MAT_INT pij=loadGeneralData(argv[1], &machines, &jobs);
   VECTOR_INT wi=loadwj(argv[2], machines);
   //printf("El vector de wj para este fichero es: \n");
   //print_vector(wj, machines);
   //print_matrix(pij, machines, jobs);
   //printf("El numero de trabajos es: %d\n", jobs);
   //printf("El numero de maquinas es : %d\n", machines);

   VECTOR_INT solucion_final=DIM_VECTOR_INT(2);
   solucion_final=funcion_final(jobs, machines, pij, wi);
   //printf("La solucion es : %d\n", solucion_final);

   output(argv[1], argv[3], jobs, machines, solucion_final[0], solucion_final[1]);

   free_vector(wj);
   free_matrix(pij, machines);

  return 0;
}
```

Función para cargar los datos

```
MAT_INT loadGeneralData(char *data, int *machines, int *jobs)
{
FILE *input;
input=fopen(data, "r"); //Se abre con acceso lectura

if(input==NULL)
{
   printf("No se puede abrir el fichero\n");
   system("pause");
   exit(EXIT_FAILURE);
}

int temp_data;
fscanf(input, "%d\n", &temp_data);
*(machines)=temp_data;
fscanf(input, "%d\n", &temp_data);
*(jobs)=temp_data;

//printf("jobs: %d  machines: %d\n",*(jobs),*(machines));

MAT_INT pt=DIM_MAT_INT(*(machines),*(jobs));
int i,j;
for(j=0; j<*(jobs); j++)
{
   for(i=0; i<*(machines); i++)
   {
      fscanf(input, "%d\n", &temp_data);
      pt[i][j]=temp_data;
   }
}
//print_matrix(pt, *(machines), *(jobs));

fclose(input);

return pt;
}
```

Función para cargar los pesos

```
VECTOR_INT loadwj(char *datapesos, int machines)
{
FILE *input;
input=fopen(datapesos, "r"); //Se abre con acceso lectura
if(input==NULL)
{
   printf("No se puede abrir el fichero\n");
   system("pause");
   exit(EXIT_FAILURE);
}
//printf("jobs: %d  machines: %d\n",*(jobs),*(machines));
int temp_datapesos;

VECTOR_INT pesos=DIM_VECTOR_INT(machines);
int k=0;
for(k=0; k<machines; k++)
{
      fscanf(input, "%d\n", &temp_datapesos);
      pesos[k]=temp_datapesos;
}

fclose(input);

return pesos;
}
```

Algoritmo NEH

```
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt, VECTOR_INT wj)
{
   /*Se contabiliza el tiempo que va a tardar*/
   srand((int)time(0));
   clock_t t_ini,t_fin;
   double secs;
   t_ini=clock();

   /*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
   VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
```

```
int i=0;
int j=0;
int k=0;
setval_vector(tiempos, jobs, 0);

for(j=0; j<jobs; j++)
{
   for(i=0; i<machines; i++)
   {
      tiempos[j]+=pt[i][j];
   }
}

printf("\nEl vector de tiempos es:\n");
print_vector(tiempos,jobs);

/*Genero secuencia inicial*/
VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
sort_vector(tiempos,sec_inicial_neh, jobs,'D');

///printf("\n La secuencia inicial para NEH es:\n");
///print_vector(sec_inicial_neh, jobs);

/*Funcion objetivo para la secuencia inicial*/
int fo=funcion_objetivo(jobs, machines, pt, sec_inicial_neh, wj);
///printf("La funcion objetivo para la secuencia inicial del NEH vale: %d\n", fo);

/*Calculo de la funcion objetivo con neh*/
VECTOR_INT secuencia_parcial=DIM_VECTOR_INT(jobs+1);
setval_vector(secuencia_parcial, jobs+1, -1);
VECTOR_INT best_secuencia=DIM_VECTOR_INT(jobs+1);
setval_vector(best_secuencia, jobs+1, -1);

copy_vector(sec_inicial_neh, best_secuencia, jobs); //copio solo la parte primera o la que me conviene
int job_to_insert;
int best_fo;
VECTOR_INT solucion=DIM_VECTOR_INT(2);
```

```
for(j=1; j<jobs; j++)
{
   copy_vector(best_secuencia, secuencia_parcial, j+1);
   job_to_insert=sec_inicial_neh[j]; //si se inicializa a 0 seria en j+1 y  j si se inicializa a 1
   //printf("\nEl numero a insertar en la siguiente posicion es: %d\n", job_to_insert);

   best_fo=10000000; //porque cada vez que analizo con el bucle de abajo tengo diferentes valores para bet_fo por el tamñao del vecto

   for(k=0; k<j+1; k++)
   {
      insert_vector(secuencia_parcial, j+1, job_to_insert, k);
      //printf("El vector tras la insercion:\n");
      //print_vector(secuencia_parcial, j+1);
      int fo_neh=funcion_objetivo(j+1, machines, pt, secuencia_parcial, wj);
      //printf("La funcion objetivo para la secuencia anterior vale: %d\n", fo_neh);
      if(fo_neh<best_fo)
      {
         best_fo=fo_neh;
         copy_vector(secuencia_parcial, best_secuencia, j+1);
      }
      //printf("\nY la mejor funcion objetivo hasta el momento vale: %d\n", best_fo);
      //printf("Y la mejor secuencia es:\n");
      //print_vector(best_secuencia, j+1);
      int new_lenght=extract_vector(secuencia_parcial, j+1, k);
      //printf("El vector tras la extracion: \n");
      //print_vector(secuencia_parcial, new_lenght);
   }
}

t_fin=clock();

secs=(double)(t_fin-t_ini)/CLOCKS_PER_SEC;
printf("%.16g\n", secs*1000);

solucion[0]=best_fo;
solucion[1]=secs*1000;
```

```
    free_vector(best_secuencia);
    free_vector(secuencia_parcial);
    free_vector(sec_inicial_neh);
    free_vector(tiempos);
    free_vector(wj);

    return solucion;
}
```

Función para el cálculo de la función objetivo

```
int funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ, VECTOR_INT w)
{
    /*Genero los cj de cada trabajo en cada maquina*/
    MAT_INT cj=DIM_MAT_INT(machines, jobs);
    VECTOR_INT tiempo_ocioso=DIM_VECTOR_INT(machines);
    setval_vector(tiempo_ocioso, machines, 0);
    setval_matrix(cj, machines, jobs, 0);

    cj[0][0]=p[0][sequ[0]];

    int i=0;
    int j=0;

    for(j=1; j<jobs; j++)
    {
        cj[0][j]=cj[0][j-1]+p[0][sequ[j]];
        //print_matrix(cj, machines, jobs);
    }

    int primer_trabajo=sequ[0];
    for(i=1; i<machines; i++)
    {
        cj[i][0]=cj[i-1][0]+p[i][primer_trabajo];
        //print_matrix(cj, machines, jobs);
    }
```

```
for(i=1; i<machines; i++)
{
   for(j=1; j<jobs; j++)
   {
      if(cj[i-1][j]>cj[i][j-1])
      {
         cj[i][j]=cj[i-1][j]+p[i][sequ[j]];
         //print_matrix(cj,machines, jobs);
      }
      else
      {
         cj[i][j]=cj[i][j-1]+p[i][sequ[j]];
         //print_matrix(cj, machines, jobs);
      }
   }
}
//print_matrix(cj, machines, jobs);

/*Calculo el tiempo ocioso*/
for(i=1; i<machines; i++)
{
   for(j=0; j<jobs-1; j++)
   {
      if(cj[i][j]>cj[i-1][j+1])
      {
         tiempo_ocioso[i]+=0;

      }
      else
      {
         tiempo_ocioso[i]+=(cj[i-1][j+1]-cj[i][j]);
      }
   }
}
//printf("El vector de tiempo ocioso por maquinas es: \n");
//print_vector(tiempo_ocioso, machines);

/*Calculo funcion objetivo como la sum de wj*tiempo_ocioso para la secuencia inicial*/
```

```
    VECTOR_INT funcion_objetivo=DIM_VECTOR_INT(machines);
    for(i=0; i<machines; i++)
    {
       funcion_objetivo[i]=w[i]*tiempo_ocioso[i];
    }

    int fo_total=0;
    for(i=0; i<machines; i++)
    {
       fo_total=(fo_total+funcion_objetivo[i]);
    }

    free_vector(tiempo_ocioso);
    free_vector(funcion_objetivo);
    free_matrix(cj, machines);

    return fo_total;

}
```

Función para escritura de la solución en fichero

```
void output(char *data, char *solution, int n, int m, int s, int t)
{
    FILE *salida;
    salida=fopen(solution, "a");

    fprintf(salida,"%s   %d    %d    %d    %d\n", data, n, m, s, t);

    /*int i=0;2
    for(i=0; i<n; i++)
    {
       fprintf(salida, "%d ", sec[i]);
    }*/

    fclose(salida);
}
```

NEH con regla de desempate TB

```
#include <schedule.h>
#include <time.h>

MAT_INT loadGeneralData(char *data, int *jobs, int *machines);
VECTOR_INT loadwj(char *datapesos, int jobs);
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt, VECTOR_INT wj);
VECTOR_INT funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ, VECTOR_INT wj);
void output(char *data, char *solution, int n, int m, int s, int t);
```

Función objetivo para cargar los datos

```
int main(int argc,char *argv[])
{
   int machines;
   int jobs;

   MAT_INT pij=loadGeneralData(argv[1], &machines, &jobs);
   VECTOR_INT wj=loadwj(argv[2], machines);
   //printf("El vector de wj para este fichero es: \n");
   //print_vector(wj, machines);

   ///print_matrix(pij, machines, jobs);
   //printf("El numero de trabajos es: %d\n", jobs);
   //printf("El numero de maquinas es : %d\n", machines);

   VECTOR_INT solucion_final=funcion_final(jobs, machines, pij, wj);
   ///printf("La solucion es : %d\n", solucion_final);

   output(argv[1], argv[3], jobs, machines, solucion_final[0], solucion_final[1]);

   free_vector(wj);
  free_matrix(pij, machines);
  return 0;
}

MAT_INT loadGeneralData(char *data, int *machines, int *jobs)
```

```
{
FILE *input;
input=fopen(data, "r"); //abrimos con acceso de lectura

if(input==NULL)
{
   printf("No se puede abrir el fichero\n");
   system("pause");
   exit(EXIT_FAILURE);
}

int temp_data;
fscanf(input, "%d\n", &temp_data);
*(machines)=temp_data;
fscanf(input, "%d\n", &temp_data);
*(jobs)=temp_data;

//printf("jobs: %d  machines: %d\n",*(jobs),*(machines));

MAT_INT pt=DIM_MAT_INT(*(machines),*(jobs));
int i,j;
for(j=0; j<*(jobs); j++)
{
   for(i=0; i<*(machines); i++)
   {
      fscanf(input, "%d\n", &temp_data);
      pt[i][j]=temp_data;
   }
}

//print_matrix(pt, *(machines), *(jobs));

fclose(input);

return pt;
}
```

Función para cargar los pesos

```
VECTOR_INT loadwj(char *datapesos, int machines)
{
FILE *input;
input=fopen(datapesos, "r"); //ABRIMOS CON ACCESO DE LECTURA

if(input==NULL)
{
   printf("No se puede abrir el fichero\n");
   system("pause");
   exit(EXIT_FAILURE);
}

//printf("jobs: %d  machines: %d\n",*(jobs),*(machines));
int temp_datapesos;

VECTOR_INT pesos=DIM_VECTOR_INT(machines);
int k=0;
for(k=0; k<machines; k++)
{
      fscanf(input, "%d\n", &temp_datapesos);
      pesos[k]=temp_datapesos;
}

fclose(input);

return pesos;
}
```

Función Algoritmo NEH y regla de desempate TB

```
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt, VECTOR_INT wj)
{
   /*SE CONTABILIZA EL TIEMPO QUE VA A TARDAR*/
   srand((int)time(0));
   clock_t t_ini,t_fin;
   double secs;
   t_ini=clock();
```

```
/*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
int maquinas_significativas=((machines*5)/10);
//printf("maquinas significativas: %d\n", maquinas_significativas);

VECTOR_INT pesos_ordenados=DIM_VECTOR_INT(machines);
VECTOR_INT wj_copia=DIM_VECTOR_INT(machines);
copy_vector(wj, wj_copia, machines);
sort_vector(wj_copia, pesos_ordenados, machines, 'D');
//print_vector(pesos_ordenados, machines);

VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
int i=0;
int j=0;
int k=0;
setval_vector(tiempos, jobs, 0);

for(j=0; j<jobs; j++)
{
   for(i=0; i<maquinas_significativas; i++)
   {
      tiempos[j]+=pt[pesos_ordenados[i]][j];
   }
}

///printf("\nEl vector de tiempos es:\n");
///print_vector(tiempos,jobs);

/*Genero secuencia inicial*/
VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
sort_vector(tiempos,sec_inicial_neh, jobs,'D');

///printf("\n La secuencia inicial para NEH es:\n");
///print_vector(sec_inicial_neh, jobs);

/*Funcion objetivo para la secuencia inicial*/
int fo=funcion_objetivo(jobs, machines, pt, sec_inicial_neh, wj);
///printf("La funcion objetivo para la secuencia inicial del NEH vale: %d\n", fo);
```

```
/*Cálculo de la funcion objetivo con neh*/
VECTOR_INT secuencia_parcial=DIM_VECTOR_INT(jobs+1);
setval_vector(secuencia_parcial, jobs+1, -1);
VECTOR_INT best_secuencia=DIM_VECTOR_INT(jobs+1);
setval_vector(best_secuencia, jobs+1, -1);

copy_vector(sec_inicial_neh, best_secuencia, jobs); //copio solo la parte primera o la que me conviene
int job_to_insert;
int best_fo;
VECTOR_INT solucion=DIM_VECTOR_INT(2);
VECTOR_INT funcionobjetivo_neh=DIM_VECTOR_INT(2);
VECTOR_INT funcionobjetivo_bestsecuencia=DIM_VECTOR_INT(2);

for(j=1; j<jobs; j++)
{
   copy_vector(best_secuencia, secuencia_parcial, j+1);
   job_to_insert=sec_inicial_neh[j]; //si se inicializa a 0 seria en j+1 y  j si se inicializa a 1
   ///printf("\nEl numero a insertar en la siguiente posicion es: %d\n", job_to_insert);

   best_fo=10000000; //porque cada vez que analizo con el bucle de abajo tengo diferentes valores para bet_fo por el tamñao del vecto

   for(k=0; k<j+1; k++)
   {
      insert_vector(secuencia_parcial, j+1, job_to_insert, k);
      ///printf("El vector tras la insercion:\n");
      ///print_vector(secuencia_parcial, j+1);
      funcionobjetivo_neh=funcion_objetivo(j+1, machines, pt, secuencia_parcial, wj);
      int fo_neh=funcionobjetivo_neh[0];
      ///printf("La funcion objetivo para la secuencia anterior vale: %d\n", fo_neh);
      if(fo_neh<best_fo)
      {
         best_fo=fo_neh;
         copy_vector(secuencia_parcial, best_secuencia, j+1);
      }
      else
      {
         if(fo_neh==best_fo)
```

```
        {
            funcionobjetivo_bestsecuencia=funcion_objetivo(j+1, machines, pt, best_secuencia, wj);
            int tft_bestsecuencia=funcionobjetivo_bestsecuencia[1];
            ///printf("Como son iguales las fo, el TFT de la mejor secuencia vale: %d\n", tft_bestsecuencia);
            ///printf("Y el valor del TFT  de la solucion del NEH vale: %d\n", funcionobjetivo_neh[1]);

            if(funcionobjetivo_neh[1]<=tft_bestsecuencia)
            {
               best_fo=fo_neh;
               copy_vector(secuencia_parcial, best_secuencia, j+1);
            }
         }
      }

      ///printf("\nY la mejor funcion objetivo hasta el momento vale: %d\n", best_fo);
      ///printf("Y la mejor secuencia es:\n");
      ///print_vector(best_secuencia, j+1);
      int new_lenght=extract_vector(secuencia_parcial, j+1, k);
      //printf("El vector tras la extracion: \n");
      //print_vector(secuencia_parcial, new_lenght);
   }
}
t_fin=clock();

secs=(double)(t_fin-t_ini)/CLOCKS_PER_SEC;
//printf("%.16g\n", secs*1000);

solucion[0]=best_fo;
solucion[1]=secs*1000;

free_vector(best_secuencia);
free_vector(secuencia_parcial);
free_vector(sec_inicial_neh);
free_vector(tiempos);
free_vector(wj);
free_vector(funcionobjetivo_bestsecuencia);
free_vector(funcionobjetivo_neh);
```

```
  return solucion;
}
```

Función para calcular la función objetivo

```
VECTOR_INT funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ, VECTOR_INT w)
{
  VECTOR_INT solucion=DIM_VECTOR_INT(2);

  /*Genero los cj de cada trabajo en cada maquina*/
  MAT_INT cj=DIM_MAT_INT(machines, jobs);
  VECTOR_INT tiempo_ocioso=DIM_VECTOR_INT(machines);

  setval_vector(tiempo_ocioso, machines, 0);
  setval_matrix(cj, machines, jobs, 0);

  cj[0][0]=p[0][sequ[0]];

  int i=0;
  int j=0;
  int TFT=0;

  for(j=1; j<jobs; j++)
  {
    cj[0][j]=cj[0][j-1]+p[0][sequ[j]];
    TFT=TFT+cj[0][j];
    //print_matrix(cj, machines, jobs);
  }

  int primer_trabajo=sequ[0];
  for(i=1; i<machines; i++)
  {
    cj[i][0]=cj[i-1][0]+p[i][primer_trabajo];
    TFT=TFT+cj[i][0];
    //print_matrix(cj, machines, jobs);
  }

  for(i=1; i<machines; i++)
  {
```

```
   for(j=1; j<jobs; j++)
   {
      if(cj[i-1][j]>cj[i][j-1])
      {
         cj[i][j]=cj[i-1][j]+p[i][sequ[j]];
         TFT=TFT+cj[i][j];
         //print_matrix(cj,machines, jobs);
      }
      else
      {
         cj[i][j]=cj[i][j-1]+p[i][sequ[j]];
         TFT=TFT+cj[i][j];
         //print_matrix(cj, machines, jobs);
      }
   }
}
//print_matrix(cj, machines, jobs);

/*Calculo el tiempo ocioso*/
for(i=1; i<machines; i++)
{
   for(j=0; j<jobs-1; j++)
   {
      if(cj[i][j]>cj[i-1][j+1])
      {
         tiempo_ocioso[i]+=0;

      }
      else
      {
         tiempo_ocioso[i]+=(cj[i-1][j+1]-cj[i][j]);
      }
   }
}
//printf("El vector de tiempo ocioso por maquinas es: \n");
//print_vector(tiempo_ocioso, machines);

/*Cálculo funcion objetivo como la sum de wj*tiempo_ocioso para la secuencia inicial*/
```

```
    VECTOR_INT funcion_objetivo=DIM_VECTOR_INT(machines);
    for(i=0; i<machines; i++)
    {
        funcion_objetivo[i]=w[i]*tiempo_ocioso[i];
    }

    int fo_total=0;
    for(i=0; i<machines; i++)
    {
        fo_total=(fo_total+funcion_objetivo[i]);
    }

    solucion[0]=fo_total;
    solucion[1]=TFT;

    free_vector(tiempo_ocioso);
    free_vector(funcion_objetivo);
    free_matrix(cj, machines);

    return solucion;
}
```

Función para escritura de la solución en ficheros

```
void output(char *data, char *solution, int n, int m, int s, int t)
{
    FILE *salida;
    salida=fopen(solution, "a");

    fprintf(salida,"%s   %d    %d    %d    %d\n", data, n, m, s, t);

    /*int i=0;2
    for(i=0; i<n; i++)
    {
        fprintf(salida, "%d ", sec[i]);
    }*/

    fclose(salida);
}
```

```
#include <schedule.h>
#include <time.h>

MAT_INT loadGeneralData(char *data, int *jobs, int *machines);
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt);
int funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ);
void output(char *data, char *solution, int n, int m, int s, int t);

int main(int argc,char *argv[])
{
   int machines;
   int jobs;

   MAT_INT pij=loadGeneralData(argv[1], &machines, &jobs);
   ///print_matrix(pij, machines, jobs);
   //printf("El numero de trabajos es: %d\n", jobs);
   //printf("El numero de maquinas es : %d\n", machines);

   VECTOR_INT solucion_final=DIM_VECTOR_INT(2);
   solucion_final=funcion_final(jobs, machines, pij);
   ///printf("La solucion es : %d\n", solucion_final);

   output(argv[1], argv[2], jobs, machines, solucion_final[0], solucion_final[1]);

   free_matrix(pij, machines);

  return 0;
}
```

Función para cargar los datos

```
MAT_INT loadGeneralData(char *data, int *machines, int *jobs)
{
FILE *input;
input=fopen(data, "r"); //ABRIMOS CON ACCESO DE LECTURA

if(input==NULL)
```

```
{
   printf("No se puede abrir el fichero\n");
   system("pause");
   exit(EXIT_FAILURE);
}

int temp_data;
fscanf(input, "%d\n", &temp_data);
*(machines)=temp_data;
fscanf(input, "%d\n", &temp_data);
*(jobs)=temp_data;

//printf("jobs: %d  machines: %d\n",*(jobs),*(machines));

MAT_INT pt=DIM_MAT_INT(*(machines),*(jobs));
int i,j;
for(j=0; j<*(jobs); j++)
{
   for(i=0; i<*(machines); i++)
   {
      fscanf(input, "%d\n", &temp_data);
      pt[i][j]=temp_data;
   }
}

//print_matrix(pt, *(machines), *(jobs));

fclose(input);

return pt;

}
```

Algoritmo NEH sin pesos y regla de despacho LPT

```
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt)
{
   /*SE CONTABILIZA EL TIEMPO QUE VA A TARDAR*/
   srand((int)time(0));
```

```
clock_t t_ini,t_fin;
double secs;
t_ini=clock();

/*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
int i=0;
int j=0;
int k=0;
setval_vector(tiempos, jobs, 0);

for(j=0; j<jobs; j++)
{
   for(i=0; i<machines; i++)
   {
      tiempos[j]+=pt[i][j];
   }
}

///printf("\nEl vector de tiempos es:\n");
///print_vector(tiempos,jobs);

/*Genero secuencia inicial*/
VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
sort_vector(tiempos,sec_inicial_neh, jobs,'D');

///printf("\n La secuencia inicial para NEH es:\n");
///print_vector(sec_inicial_neh, jobs);

/*Funcion objetivo para la secuencia inicial*/
int fo=funcion_objetivo(jobs, machines, pt, sec_inicial_neh);
///printf("La funcion objetivo para la secuencia inicial del NEH vale: %d\n", fo);

/*Calculo de la funcion objetivo con neh*/
VECTOR_INT secuencia_parcial=DIM_VECTOR_INT(jobs+1);
setval_vector(secuencia_parcial, jobs+1, -1);
VECTOR_INT best_secuencia=DIM_VECTOR_INT(jobs+1);
setval_vector(best_secuencia, jobs+1, -1);
```

```
copy_vector(sec_inicial_neh, best_secuencia, jobs); //copio solo la parte primera o la que me conviene
int job_to_insert;
int best_fo;
VECTOR_INT solucion=DIM_VECTOR_INT(2);

for(j=1; j<jobs; j++)
{
   copy_vector(best_secuencia, secuencia_parcial, j+1);
   job_to_insert=sec_inicial_neh[j]; //si se inicializa a 0 seria en j+1 y  j si se inicializa a 1
   ///printf("\nEl numero a insertar en la siguiente posicion es: %d\n", job_to_insert);

   best_fo=10000000; //porque cada vez que analizo con el bucle de abajo tengo diferentes valores para bet_fo por el tamñao del vecto

   for(k=0; k<j+1; k++)
   {
      insert_vector(secuencia_parcial, j+1, job_to_insert, k);
      ///printf("El vector tras la insercion:\n");
      ///print_vector(secuencia_parcial, j+1);
      int fo_neh=funcion_objetivo(j+1, machines, pt, secuencia_parcial);
      ///printf("La funcion objetivo para la secuencia anterior vale: %d\n", fo_neh);
      if(fo_neh<best_fo)
      {
         best_fo=fo_neh;
         copy_vector(secuencia_parcial, best_secuencia, j+1);
      }
      ///printf("\nY la mejor funcion objetivo hasta el momento vale: %d\n", best_fo);
      ///printf("Y la mejor secuencia es:\n");
      ///print_vector(best_secuencia, j+1);
      int new_lenght=extract_vector(secuencia_parcial, j+1, k);
      //printf("El vector tras la extracion: \n");
      //print_vector(secuencia_parcial, new_lenght);
   }
}
t_fin=clock();

secs=(double)(t_fin-t_ini)/CLOCKS_PER_SEC;
```

```
    ///printf("%.16g\n", secs*1000);

    solucion[0]=best_fo;
    solucion[1]=secs*1000;

    free_vector(best_secuencia);
    free_vector(secuencia_parcial);
    free_vector(sec_inicial_neh);
    free_vector(tiempos);


    return solucion;
}
```

Función para cálculo de la función objetivo

```
int funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ)
{
    /*Genero los cj de cada trabajo en cada maquina*/
    MAT_INT cj=DIM_MAT_INT(machines, jobs);
    VECTOR_INT tiempo_ocioso=DIM_VECTOR_INT(machines);

    setval_vector(tiempo_ocioso, machines, 0);
    setval_matrix(cj, machines, jobs, 0);

    cj[0][0]=p[0][sequ[0]];

    int i=0;
    int j=0;

    for(j=1; j<jobs; j++)
    {
        cj[0][j]=cj[0][j-1]+p[0][sequ[j]];
        //print_matrix(cj, machines, jobs);
    }

    int primer_trabajo=sequ[0];
    for(i=1; i<machines; i++)
```

```
{
   cj[i][0]=cj[i-1][0]+p[i][primer_trabajo];
   //print_matrix(cj, machines, jobs);
}

for(i=1; i<machines; i++)
{
   for(j=1; j<jobs; j++)
   {
      if(cj[i-1][j]>cj[i][j-1])
      {
         cj[i][j]=cj[i-1][j]+p[i][sequ[j]];
         //print_matrix(cj,machines, jobs);
      }
      else
      {
         cj[i][j]=cj[i][j-1]+p[i][sequ[j]];
         //print_matrix(cj, machines, jobs);
      }
   }
}
//print_matrix(cj, machines, jobs);

/*Calculo el tiempo ocioso*/
for(i=1; i<machines; i++)
{
   for(j=0; j<jobs-1; j++)
   {
      if(cj[i][j]>cj[i-1][j+1])
      {
         tiempo_ocioso[i]+=0;

      }
      else
      {
         tiempo_ocioso[i]+=(cj[i-1][j+1]-cj[i][j]);
      }
   }
```

```
    }
    //printf("El vector de tiempo ocioso por maquinas es: \n");
    //print_vector(tiempo_ocioso, machines);

    /*Calculo funcion objetivo como la sum de wj*tiempo_ocioso para la secuencia inicial*/
    int fo_total=0;
    for(i=0; i<machines; i++)
    {
       fo_total=(fo_total+tiempo_ocioso[i]);
    }

    free_vector(tiempo_ocioso);
    free_matrix(cj, machines);

    return fo_total;
}
```

Función para escritura de la solución en fichero

```
void output(char *data, char *solution, int n, int m, int s, int t)
{
    FILE *salida;
    salida=fopen(solution, "a");

    fprintf(salida,"%s   %d    %d    %d    %d\n", data, n, m, s, t);

    /*int i=0;2
    for(i=0; i<n; i++)
    {
       fprintf(salida, "%d ", sec[i]);
    }*/

    fclose(salida);
}


#include <schedule.h>
#include <time.h>
```

```
MAT_INT loadGeneralData(char *data, int *jobs, int *machines);
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt);
int funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ);
void output(char *data, char *solution, int n, int m, int s, int t);

int main(int argc,char *argv[])

{
   int machines;
   int jobs;

   MAT_INT pij=loadGeneralData(argv[1], &machines, &jobs);

   //print_matrix(pij, machines, jobs);
   //printf("El numero de trabajos es: %d\n", jobs);
   //printf("El numero de maquinas es : %d\n", machines);

   VECTOR_INT solucion_final=DIM_VECTOR_INT(2);
   solucion_final=funcion_final(jobs, machines, pij);
   //printf("La solucion es : %d\n", solucion_final);

   output(argv[1], argv[2], jobs, machines, solucion_final[0], solucion_final[1]);

   free_matrix(pij, machines);



  return 0;
}
```

Función para cargar datos

```
MAT_INT loadGeneralData(char *data, int *machines, int *jobs)
{
FILE *input;
input=fopen(data, "r"); //ABRIMOS CON ACCESO DE LECTURA

if(input==NULL)
```

```
{
   printf("No se puede abrir el fichero\n");
   system("pause");
   exit(EXIT_FAILURE);
}

int temp_data;
fscanf(input, "%d\n", &temp_data);
*(machines)=temp_data;
fscanf(input, "%d\n", &temp_data);
*(jobs)=temp_data;

//printf("jobs: %d  machines: %d\n",*(jobs),*(machines));

MAT_INT pt=DIM_MAT_INT(*(machines),*(jobs));
int i,j;
for(j=0; j<*(jobs); j++)
{
   for(i=0; i<*(machines); i++)
   {
      fscanf(input, "%d\n", &temp_data);
      pt[i][j]=temp_data;
   }
}

//print_matrix(pt, *(machines), *(jobs));

fclose(input);

return pt;
}
```

Algoritmo NEH sin pesos y regla de desempate TFT

```
VECTOR_INT funcion_final(int jobs, int machines, MAT_INT pt)
{
   /*Se contabiliza el tiempo que va a tardar*/
   srand((int)time(0));
```

```
clock_t t_ini,t_fin;
double secs;
t_ini=clock();

/*Sumo los pt de cada trabajo en todas las maquinas y los guardo en un vector para despues ordenarlos*/
VECTOR_INT tiempos=DIM_VECTOR_INT(jobs);
int i=0;
int j=0;
int k=0;
setval_vector(tiempos, jobs, 0);



for(j=0; j<jobs; j++)
{
   for(i=0; i<machines; i++)
   {
      tiempos[j]+=pt[i][j];
   }
}

//printf("\nEl vector de tiempos es:\n");
//print_vector(tiempos,jobs);

/*Genero secuencia inicial*/
VECTOR_INT sec_inicial_neh=DIM_VECTOR_INT(jobs);
sort_vector(tiempos,sec_inicial_neh, jobs,'D');

//printf("\n La secuencia inicial para NEH es:\n");
//print_vector(sec_inicial_neh, jobs);

/*Funcion objetivo para la secuencia inicial*/
int fo=funcion_objetivo(jobs, machines, pt, sec_inicial_neh);
//printf("La funcion objetivo para la secuencia inicial del NEH vale: %d\n", fo);

/*Calculo de la funcion objetivo con neh*/
VECTOR_INT secuencia_parcial=DIM_VECTOR_INT(jobs+1);
setval_vector(secuencia_parcial, jobs+1, -1);
VECTOR_INT best_secuencia=DIM_VECTOR_INT(jobs+1);
```

```
setval_vector(best_secuencia, jobs+1, -1);

copy_vector(sec_inicial_neh, best_secuencia, jobs); //copio solo la parte primera o la que me conviene
int job_to_insert;
int best_fo;
VECTOR_INT solucion=DIM_VECTOR_INT(2);
VECTOR_INT funcionobjetivo_neh=DIM_VECTOR_INT(2);
VECTOR_INT funcionobjetivo_bestsecuencia=DIM_VECTOR_INT(2);

for(j=1; j<jobs; j++)
{
   copy_vector(best_secuencia, secuencia_parcial, j+1);
   job_to_insert=sec_inicial_neh[j]; //si se inicializa a 0 seria en j+1 y  j si se inicializa a 1
   //printf("\nEl numero a insertar en la siguiente posicion es: %d\n", job_to_insert);

   best_fo=10000000; //porque cada vez que analizo con el bucle de abajo tengo diferentes valores para bet_fo por el tamñao del vecto

   for(k=0; k<j+1; k++)
   {
      insert_vector(secuencia_parcial, j+1, job_to_insert, k);
      //printf("El vector tras la insercion:\n");
      //print_vector(secuencia_parcial, j+1);
      funcionobjetivo_neh=funcion_objetivo(j+1, machines, pt, secuencia_parcial);
      int fo_neh=funcionobjetivo_neh[0];
      //printf("La funcion objetivo para la secuencia anterior vale: %d\n", fo_neh);
      if(fo_neh<best_fo)
      {
         best_fo=fo_neh;
         copy_vector(secuencia_parcial, best_secuencia, j+1);
      }
      else
      {
         if(fo_neh==best_fo)
         {
            funcionobjetivo_bestsecuencia=funcion_objetivo(j+1, machines, pt, best_secuencia);
            int tft_bestsecuencia=funcionobjetivo_bestsecuencia[1];
            //printf("Como son iguales las fo, el TFT de la mejor secuencia vale: %d\n", tft_bestsecuencia);
```

```
            //printf("Y el valor del TFT  de la solucion del NEH vale: %d\n", funcionobjetivo_neh[1]);

            if(funcionobjetivo_neh[1]<=tft_bestsecuencia)
            {
               best_fo=fo_neh;
               copy_vector(secuencia_parcial, best_secuencia, j+1);
            }
          }
        }

        //printf("\nY la mejor funcion objetivo hasta el momento vale: %d\n", best_fo);
        //printf("Y la mejor secuencia es:\n");
        //print_vector(best_secuencia, j+1);
        int new_lenght=extract_vector(secuencia_parcial, j+1, k);
        //printf("El vector tras la extracion: \n");
        //print_vector(secuencia_parcial, new_lenght);
      }
    }
    t_fin=clock();

    secs=(double)(t_fin-t_ini)/CLOCKS_PER_SEC;
    //printf("%.16g\n", secs*1000);

    solucion[0]=best_fo;
    solucion[1]=secs*1000;

    free_vector(best_secuencia);
    free_vector(secuencia_parcial);
    free_vector(sec_inicial_neh);
    free_vector(tiempos);
    free_vector(funcionobjetivo_bestsecuencia);
    free_vector(funcionobjetivo_neh);

    return solucion;

}
```

Función para el cálculo de la función objetivo

```
int funcion_objetivo(int jobs, int machines, MAT_INT p, VECTOR_INT sequ)
{
   VECTOR_INT solucion=DIM_VECTOR_INT(2);

   /*Genero los cj de cada trabajo en cada maquina*/
   MAT_INT cj=DIM_MAT_INT(machines, jobs);
   VECTOR_INT tiempo_ocioso=DIM_VECTOR_INT(machines);

   setval_vector(tiempo_ocioso, machines, 0);
   setval_matrix(cj, machines, jobs, 0);

   cj[0][0]=p[0][sequ[0]];

   int i=0;
   int j=0;
   int TFT=0;

   for(j=1; j<jobs; j++)
   {
      cj[0][j]=cj[0][j-1]+p[0][sequ[j]];
      TFT=TFT+cj[0][j];
      //print_matrix(cj, machines, jobs);
   }

   int primer_trabajo=sequ[0];
   for(i=1; i<machines; i++)
   {
      cj[i][0]=cj[i-1][0]+p[i][primer_trabajo];
      TFT=TFT+cj[i][0];
      //print_matrix(cj, machines, jobs);
   }

   for(i=1; i<machines; i++)
   {
      for(j=1; j<jobs; j++)
      {
         if(cj[i-1][j]>cj[i][j-1])
         {
```

```
            cj[i][j]=cj[i-1][j]+p[i][sequ[j]];
            TFT=TFT+cj[i][j];
            //print_matrix(cj,machines, jobs);
        }
        else
        {
            cj[i][j]=cj[i][j-1]+p[i][sequ[j]];
            TFT=TFT+cj[i][j];
            //print_matrix(cj, machines, jobs);
        }
    }
}
//print_matrix(cj, machines, jobs);

/*Calculo el tiempo ocioso*/
for(i=1; i<machines; i++)
{
    for(j=0; j<jobs-1; j++)
    {
        if(cj[i][j]>cj[i-1][j+1])
        {
            tiempo_ocioso[i]+=0;

        }
        else
        {
            tiempo_ocioso[i]+=(cj[i-1][j+1]-cj[i][j]);
        }
    }
}
//printf("El vector de tiempo ocioso por maquinas es: \n");
//print_vector(tiempo_ocioso, machines);

/*Calculo funcion objetivo como la sum de wj*tiempo_ocioso para la secuencia inicial*/
int fo_total=0;
for(i=0; i<machines; i++)
{
    fo_total=(fo_total+tiempo_ocioso[i]);
```

```
    }

    solucion[0]=fo_total;
    solucion[1]=TFT;

    free_vector(tiempo_ocioso);
    free_matrix(cj, machines);

    return solucion;

}
```

Función para escritura de la solución en ficheros

```
void output(char *data, char *solution, int n, int m, int s, int t)
{
    FILE *salida;
    salida=fopen(solution, "a");

    fprintf(salida,"%s   %d    %d    %d    %d\n", data, n, m, s, t);

    /*int i=0;2
    for(i=0; i<n; i++)
    {
        fprintf(salida, "%d ", sec[i]);
    }*/

    fclose(salida);
}
```

ANEXO II: TABLAS DE RESULTADOS

Tabla 16. Resultados función objetivo para cada variante según la secuencia inicial.

Instancia	Trabajos	Máquinas	NEH	NEH (PR A)	NEH (PR B)	NEH (PR C)
TA001	20	5	9869	11794	6979	7599
TA002	20	5	1759	160	2494	4322
TA003	20	5	13082	15134	13082	13880
TA004	20	5	1646	2657	4654	3275
TA005	20	5	15680	16779	16023	17671
TA006	20	5	3918	3952	3918	5454
TA007	20	5	1394	1779	1394	2550
TA008	20	5	1584	839	2017	1584
TA009	20	5	12123	14061	14526	16513
TA010	20	5	1752	6299	4645	5738
TA011	20	10	46361	54728	48380	52227
TA012	20	10	24912	36296	24635	30955
TA013	20	10	79751	86011	76241	89792
TA014	20	10	55968	61878	62204	66130
TA015	20	10	43929	43869	43047	47464
TA016	20	10	61412	61184	53464	56194
TA017	20	10	19079	19682	18055	23139
TA018	20	10	41650	41277	48369	33270
TA019	20	10	39918	42688	38628	50091
TA020	20	10	43757	51076	57643	42455
TA021	20	20	253018	266823	264068	314346
TA022	20	20	148159	154217	153184	143378
TA023	20	20	193643	198742	202828	205714
TA024	20	20	218813	223813	215151	203451
TA025	20	20	204734	230216	217707	240250
TA026	20	20	139010	131595	143142	119224
TA027	20	20	168794	157462	161200	179504
TA028	20	20	296224	285199	266705	288902
TA029	20	20	206789	186879	201199	196821
TA030	20	20	234877	229324	237243	240334
TA031	50	5	46002	45401	45281	46669
TA032	50	5	16016	16026	16224	15925
TA033	50	5	21852	22596	22176	21264
TA034	50	5	6024	6024	6024	6216
TA035	50	5	52269	46959	53241	48822
TA036	50	5	7983	8778	6688	12863

TA037	50	5	8233	8260	8110	8435
TA038	50	5	9575	16909	9989	14150
TA039	50	5	7488	8092	10514	9277
TA040	50	5	3950	5484	8680	6238
TA041	50	10	88922	92509	97400	95182
TA042	50	10	42386	64656	51166	86608
TA043	50	10	62345	61818	60666	69822
TA044	50	10	52156	55690	54216	61228
TA045	50	10	37653	47601	42345	56346
TA046	50	10	29188	35723	39777	32260
TA047	50	10	72110	82825	80463	75857
TA048	50	10	93658	104922	109989	100428
TA049	50	10	63169	58464	60589	76118
TA050	50	10	39429	48069	52452	55739
TA051	50	20	322608	292213	319830	320004
TA052	50	20	302104	283426	292980	329846
TA053	50	20	296092	287262	273736	278916
TA054	50	20	340239	339226	322658	352373
TA055	50	20	217839	243249	200226	242768
TA056	50	20	302841	313697	311844	332455
TA057	50	20	346968	375461	362760	372397
TA058	50	20	364778	363250	357740	367057
TA059	50	20	294748	275484	281047	343556
TA060	50	20	273735	294729	272178	283467
TA061	100	5	18842	25321	21682	28285
TA062	100	5	7400	9163	6497	13745
TA063	100	5	4294	6954	4313	16416
TA064	100	5	9314	17442	11960	22594
TA065	100	5	45471	44618	44620	44498
TA066	100	5	6245	14721	23924	12050
TA067	100	5	31692	35342	32892	57942
TA068	100	5	975	9200	1280	11247
TA069	100	5	6786	7354	11870	17946
TA070	100	5	2296	5014	2834	9505
TA071	100	10	103404	122087	111108	136315
TA072	100	10	95635	100391	92472	109037
TA073	100	10	173525	169657	170197	188285
TA074	100	10	82639	100548	96666	118271
TA075	100	10	132340	135962	129123	164299
TA076	100	10	81800	88448	93977	94523
TA077	100	10	160598	179898	181518	185781
TA078	100	10	189238	185810	196452	207670
TA079	100	10	159416	166191	166472	156064
TA080	100	10	70742	77039	77162	88074
TA081	100	20	352139	341385	348803	407815
TA082	100	20	240159	264991	256094	294038

TA083	100	20	433808	423647	422367	435500
TA084	100	20	446491	485524	468150	508980
TA085	100	20	455461	493305	496872	521587
TA086	100	20	298781	333591	285183	312964
TA087	100	20	563537	623641	592358	643480
TA088	100	20	522322	565791	543675	625620
TA089	100	20	365879	428620	377260	444416
TA090	100	20	489595	483025	444964	544229
TA091	200	10	249576	284890	249789	299451
TA092	200	10	55990	79830	55571	92742
TA093	200	10	199843	213740	212138	223365
TA094	200	10	64117	106360	79621	143104
TA095	200	10	205979	202958	206515	283183
TA096	200	10	88705	145304	110677	108627
TA097	200	10	18053	53925	55685	74334
TA098	200	10	116938	135439	135801	137344
TA099	200	10	43433	84491	51189	96309
TA100	200	10	111678	136530	138310	181913
TA101	200	20	578802	676308	627776	657180
TA102	200	20	411332	532126	473077	646972
TA103	200	20	493638	584103	586495	597930
TA104	200	20	402133	510368	481011	468205
TA105	200	20	717974	826868	742043	856608
TA106	200	20	59541	639781	646526	661585
TA107	200	20	665100	767186	713337	771367
TA108	200	20	678910	790473	748882	800930
TA109	200	20	565380	643351	570228	701614
TA110	200	20	551147	589038	538979	607353
TA111	500	20	639448	781446	694459	1008411
TA112	500	20	717664	910913	804609	933133
TA113	500	20	742759	912571	823477	1032683
TA114	500	20	719373	712031	825765	926180
TA115	500	20	783999	864763	786588	1105990
TA116	500	20	582825	778802	780382	956191
TA117	500	20	1045432	1252047	1183154	1333576
TA118	500	20	800122	1105401	863742	1067688
TA119	500	20	566670	721877	657332	901467
TA120	500	20	610312	846104	749726	922082

Tabla 17. RPD de cada variante según la secuencia inicial.

Instancia	Trabajos	Máquinas	NEH	NEH (PR A)	NEH (PR B)	NEH (PR C)
TA001	20	5	6,079	72,713	4,006	3,797

TA002	20	5	0,261	0,000	0,789	1,729
TA003	20	5	8,385	93,588	8,385	7,763
TA004	20	5	0,181	15,606	2,339	1,068
TA005	20	5	10,248	103,869	10,494	10,156
TA006	20	5	1,811	23,700	1,811	2,443
TA007	20	5	0,000	10,119	0,000	0,610
TA008	20	5	0,136	4,244	0,447	0,000
TA009	20	5	7,697	86,881	9,420	9,425
TA010	20	5	0,257	38,369	2,332	2,622
TA011	20	10	1,430	1,781	1,680	1,257
TA012	20	10	0,306	0,844	0,364	0,338
TA013	20	10	3,180	3,370	3,223	2,881
TA014	20	10	1,933	2,144	2,445	1,858
TA015	20	10	1,302	1,229	1,384	1,051
TA016	20	10	2,219	2,109	1,961	1,429
TA017	20	10	0,000	0,000	0,000	0,000
TA018	20	10	1,183	1,097	1,679	0,438
TA019	20	10	1,092	1,169	1,139	1,165
TA020	20	10	1,293	1,595	2,193	0,835
TA021	20	20	0,820	1,028	0,845	1,637
TA022	20	20	0,066	0,172	0,070	0,203
TA023	20	20	0,393	0,510	0,417	0,725
TA024	20	20	0,574	0,701	0,503	0,706
TA025	20	20	0,473	0,749	0,521	1,015
TA026	20	20	0,000	0,000	0,000	0,000
TA027	20	20	0,214	0,197	0,126	0,506
TA028	20	20	1,131	1,167	0,863	1,423
TA029	20	20	0,488	0,420	0,406	0,651
TA030	20	20	0,690	0,743	0,657	1,016
TA031	50	5	10,646	7,279	6,517	6,508
TA032	50	5	3,055	1,922	1,693	1,562
TA033	50	5	4,532	3,120	2,681	2,421
TA034	50	5	0,525	0,098	0,000	0,000
TA035	50	5	12,233	7,563	7,838	6,854
TA036	50	5	1,021	0,601	0,110	1,069
TA037	50	5	1,084	0,506	0,346	0,357
TA038	50	5	1,424	2,083	0,658	1,276
TA039	50	5	0,896	0,476	0,745	0,492
TA040	50	5	0,000	0,000	0,441	0,004
TA041	50	10	2,047	1,590	1,449	1,950
TA042	50	10	0,452	0,810	0,286	1,685
TA043	50	10	1,136	0,730	0,525	1,164
TA044	50	10	0,787	0,559	0,363	0,898
TA045	50	10	0,290	0,333	0,065	0,747
TA046	50	10	0,000	0,000	0,000	0,000
TA047	50	10	1,471	1,319	1,023	1,351

TA048	50	10	2,209	1,937	1,765	2,113
TA049	50	10	1,164	0,637	0,523	1,360
TA050	50	10	0,351	0,346	0,319	0,728
TA051	50	20	0,481	0,201	0,597	0,318
TA052	50	20	0,387	0,165	0,463	0,359
TA053	50	20	0,359	0,181	0,367	0,149
TA054	50	20	0,562	0,395	0,611	0,451
TA055	50	20	0,000	0,000	0,000	0,000
TA056	50	20	0,390	0,290	0,557	0,369
TA057	50	20	0,593	0,544	0,812	0,534
TA058	50	20	0,675	0,493	0,787	0,512
TA059	50	20	0,353	0,133	0,404	0,415
TA060	50	20	0,257	0,212	0,359	0,168
TA061	100	5	18,325	4,050	15,939	1,976
TA062	100	5	6,590	0,827	4,076	0,446
TA063	100	5	3,404	0,387	2,370	0,727
TA064	100	5	8,553	2,479	8,344	1,377
TA065	100	5	45,637	7,899	33,859	3,682
TA066	100	5	5,405	1,936	17,691	0,268
TA067	100	5	31,505	6,049	24,697	5,096
TA068	100	5	0,000	0,835	0,000	0,183
TA069	100	5	5,960	0,467	8,273	0,888
TA070	100	5	1,355	0,000	1,214	0,000
TA071	100	10	0,462	0,585	0,440	0,548
TA072	100	10	0,352	0,303	0,198	0,238
TA073	100	10	1,453	1,202	1,206	1,138
TA074	100	10	0,168	0,305	0,253	0,343
TA075	100	10	0,871	0,765	0,673	0,865
TA076	100	10	0,156	0,148	0,218	0,073
TA077	100	10	1,270	1,335	1,352	1,109
TA078	100	10	1,675	1,412	1,546	1,358
TA079	100	10	1,253	1,157	1,157	0,772
TA080	100	10	0,000	0,000	0,000	0,000
TA081	100	20	0,466	0,288	0,362	0,387
TA082	100	20	0,000	0,000	0,000	0,000
TA083	100	20	0,806	0,599	0,649	0,481
TA084	100	20	0,859	0,832	0,828	0,731
TA085	100	20	0,896	0,862	0,940	0,774
TA086	100	20	0,244	0,259	0,114	0,064
TA087	100	20	1,347	1,353	1,313	1,188
TA088	100	20	1,175	1,135	1,123	1,128
TA089	100	20	0,523	0,617	0,473	0,511
TA090	100	20	1,039	0,823	0,738	0,851
TA091	200	10	12,825	4,283	3,880	3,028
TA092	200	10	2,101	0,480	0,086	0,248
TA093	200	10	10,070	2,964	3,144	2,005
TA094	200	10	2,552	0,972	0,555	0,925

TA095	200	10	10,410	2,764	3,034	2,810
TA096	200	10	3,914	1,695	1,162	0,461
TA097	200	10	0,000	0,000	0,088	0,000
TA098	200	10	5,477	1,512	1,653	0,848
TA099	200	10	1,406	0,567	0,000	0,296
TA100	200	10	5,186	1,532	1,702	1,447
TA101	200	20	8,721	0,325	0,327	0,404
TA102	200	20	5,908	0,043	0,000	0,382
TA103	200	20	7,291	0,144	0,240	0,277
TA104	200	20	5,754	0,000	0,017	0,000
TA105	200	20	11,058	0,620	0,569	0,830
TA106	200	20	0,000	0,254	0,367	0,413
TA107	200	20	10,170	0,503	0,508	0,647
TA108	200	20	10,402	0,549	0,583	0,711
TA109	200	20	8,496	0,261	0,205	0,499
TA110	200	20	8,257	0,154	0,139	0,297
TA111	500	20	0,128	0,097	0,056	0,119
TA112	500	20	0,266	0,279	0,224	0,035
TA113	500	20	0,311	0,282	0,253	0,146
TA114	500	20	0,269	0,000	0,256	0,027
TA115	500	20	0,384	0,215	0,197	0,227
TA116	500	20	0,029	0,094	0,187	0,061
TA117	500	20	0,845	0,758	0,800	0,479
TA118	500	20	0,412	0,552	0,314	0,184
TA119	500	20	0,000	0,014	0,000	0,000
TA120	500	20	0,077	0,188	0,141	0,023

Tabla 18 .Tiempos de ejecución de cada variante según la secuencia inicial

Instancia	Trabajos	Máquinas	Tiempo CPU NEH (s)	Tiempo CPU NEH (PR A) (s)	Tiempo CPU NEH (PR B) (s)	Tiempo CPU NEH (PR C) (s)
TA001	20	5	0,001	0,001	0,001	0,001
TA002	20	5	0,001	0,001	0,001	0,001
TA003	20	5	0,001	0,001	0,001	0,001
TA004	20	5	0,001	0,001	0,001	0,001
TA005	20	5	0,001	0,001	0,001	0,001
TA006	20	5	0,001	0,001	0,001	0,001
TA007	20	5	0,001	0,001	0,001	0,001
TA008	20	5	0,001	0,001	0,001	0,001
TA009	20	5	0,001	0,001	0,001	0,001

TA010	20	5	0,001	0,001	0,001	0,001
TA011	20	10	0,001	0,002	0,002	0,001
TA012	20	10	0,001	0,001	0,001	0,002
TA013	20	10	0,002	0,001	0,001	0,001
TA014	20	10	0,002	0,001	0,002	0,001
TA015	20	10	0,001	0,002	0,001	0,001
TA016	20	10	0,001	0,001	0,001	0,001
TA017	20	10	0,001	0,001	0,001	0,001
TA018	20	10	0,001	0,001	0,001	0,001
TA019	20	10	0,001	0,002	0,001	0,002
TA020	20	10	0,001	0,002	0,002	0,001
TA021	20	20	0,001	0,002	0,002	0,002
TA022	20	20	0,003	0,002	0,002	0,002
TA023	20	20	0,002	0,002	0,002	0,002
TA024	20	20	0,002	0,002	0,002	0,002
TA025	20	20	0,001	0,002	0,001	0,002
TA026	20	20	0,002	0,002	0,002	0,002
TA027	20	20	0,001	0,002	0,002	0,001
TA028	20	20	0,001	0,002	0,002	0,002
TA029	20	20	0,002	0,002	0,002	0,002
TA030	20	20	0,002	0,002	0,002	0,002
TA031	50	5	0,006	0,007	0,007	0,006
TA032	50	5	0,006	0,006	0,007	0,007
TA033	50	5	0,006	0,010	0,007	0,006
TA034	50	5	0,006	0,008	0,007	0,007
TA035	50	5	0,006	0,007	0,007	0,007
TA036	50	5	0,007	0,008	0,008	0,006
TA037	50	5	0,007	0,007	0,007	0,007
TA038	50	5	0,007	0,007	0,006	0,007
TA039	50	5	0,008	0,007	0,007	0,007
TA040	50	5	0,006	0,007	0,008	0,007
TA041	50	10	0,014	0,012	0,014	0,014
TA042	50	10	0,014	0,014	0,014	0,016
TA043	50	10	0,012	0,012	0,012	0,012
TA044	50	10	0,012	0,012	0,012	0,014
TA045	50	10	0,014	0,012	0,012	0,012
TA046	50	10	0,014	0,012	0,014	0,012
TA047	50	10	0,012	0,014	0,014	0,012
TA048	50	10	0,014	0,016	0,012	0,014
TA049	50	10	0,014	0,014	0,012	0,012
TA050	50	10	0,014	0,014	0,014	0,012
TA051	50	20	0,025	0,022	0,022	0,025
TA052	50	20	0,029	0,024	0,022	0,030
TA053	50	20	0,022	0,022	0,022	0,021
TA054	50	20	0,022	0,024	0,022	0,022
TA055	50	20	0,022	0,022	0,024	0,029
TA056	50	20	0,022	0,022	0,025	0,022

TA057	50	20	0,022	0,025	0,022	0,024
TA058	50	20	0,022	0,022	0,022	0,024
TA059	50	20	0,022	0,024	0,022	0,024
TA060	50	20	0,024	0,025	0,022	0,024
TA061	100	5	0,040	0,042	0,042	0,042
TA062	100	5	0,038	0,041	0,041	0,04
TA063	100	5	0,038	0,042	0,038	0,043
TA064	100	5	0,042	0,042	0,042	0,04
TA065	100	5	0,040	0,040	0,041	0,038
TA066	100	5	0,038	0,038	0,038	0,04
TA067	100	5	0,041	0,042	0,05	0,043
TA068	100	5	0,038	0,042	0,038	0,041
TA069	100	5	0,038	0,041	0,040	0,042
TA070	100	5	0,042	0,042	0,040	0,043
TA071	100	10	0,085	0,088	0,085	0,097
TA072	100	10	0,087	0,086	0,085	0,085
TA073	100	10	0,084	0,085	0,09	0,092
TA074	100	10	0,081	0,085	0,088	0,088
TA075	100	10	0,082	0,086	0,086	0,089
TA076	100	10	0,083	0,087	0,094	0,097
TA077	100	10	0,095	0,089	0,089	0,097
TA078	100	10	0,085	0,088	0,083	0,090
TA079	100	10	0,084	0,088	0,083	0,087
TA080	100	10	0,084	0,087	0,086	0,101
TA081	100	20	0,167	0,169	0,168	0,168
TA082	100	20	0,154	0,170	0,162	0,169
TA083	100	20	0,16	0,167	0,163	0,171
TA084	100	20	0,17	0,174	0,170	0,166
TA085	100	20	0,164	0,176	0,174	0,173
TA086	100	20	0,164	0,172	0,170	0,173
TA087	100	20	0,17	0,17	0,175	0,171
TA088	100	20	0,163	0,167	0,171	0,173
TA089	100	20	0,164	0,174	0,167	0,171
TA090	100	20	0,171	0,180	0,169	0,172
TA091	200	10	0,604	0,628	0,628	0,621
TA092	200	10	0,591	0,574	0,594	0,612
TA093	200	10	0,591	0,597	0,611	0,620
TA094	200	10	0,575	0,615	0,600	0,619
TA095	200	10	0,605	0,613	0,595	0,643
TA096	200	10	0,589	0,642	0,603	0,595
TA097	200	10	0,555	0,605	0,592	0,616
TA098	200	10	0,590	0,610	0,590	0,612
TA099	200	10	0,564	0,604	0,581	0,616
TA100	200	10	0,584	0,613	0,598	0,652
TA101	200	20	1,181	1,223	1,223	1,383
TA102	200	20	1,169	1,235	1,199	1,25

TA103	200	20	1,205	1,235	1,209	1,241
TA104	200	20	1,173	1,26	1,219	1,241
TA105	200	20	1,177	1,241	1,209	1,241
TA106	200	20	1,207	1,215	1,221	1,205
TA107	200	20	1,175	1,199	1,201	1,241
TA108	200	20	1,181	1,237	1,197	1,266
TA109	200	20	1,175	1,241	1,209	1,300
TA110	200	20	1,185	1,241	1,213	1,276
TA111	500	20	16,823	17,786	17,082	18,542
TA112	500	20	16,696	17,204	17,274	17,282
TA113	500	20	16,853	17,568	17,099	18,381
TA114	500	20	16,989	17,131	16,841	17,490
TA115	500	20	16,888	17,615	17,106	18,420
TA116	500	20	16,832	17,42	17,201	18,185
TA117	500	20	17,103	17,382	17,423	18,350
TA118	500	20	17,120	18,306	17,684	18,152
TA119	500	20	16,596	17,196	16,922	17,964
TA120	500	20	17,184	18,277	17,797	18,948

Tabla 19 .Valores función objetivo de cada variante según la secuencia inicial y regla de desempate.

Instancia	Trabajos	Máquinas	NEH+TB	NEH (PR A) +TB	NEH (PR B) +TB	NEH (PR C) +TB
TA001	20	5	9869	11794	6979	9869
TA002	20	5	1759	56	2210	1759
TA003	20	5	13082	15134	13082	13082
TA004	20	5	528	2657	4654	528
TA005	20	5	15680	16347	16023	15680
TA006	20	5	3918	3952	3918	3918
TA007	20	5	1394	1394	1394	1394
TA008	20	5	1324	724	1868	1324
TA009	20	5	12123	12993	14526	12123
TA010	20	5	1752	6299	4645	1752
TA011	20	10	46361	54728	48380	46361
TA012	20	10	24912	36296	21435	24912
TA013	20	10	79751	86011	76241	79751
TA014	20	10	55968	62638	62204	55968
TA015	20	10	43989	43869	43047	43989
TA016	20	10	61412	62698	53464	61412
TA017	20	10	19079	19682	18055	19079
TA018	20	10	41650	41206	47123	41650
TA019	20	10	39918	42688	38628	39918
TA020	20	10	43757	51076	57643	43757
TA021	20	20	253018	266823	264068	253018
TA022	20	20	148159	154217	153184	148159

TA023	20	20	193643	198742	202828	193643
TA024	20	20	218346	223813	214418	218346
TA025	20	20	204734	230216	217707	204734
TA026	20	20	139010	131595	143142	139010
TA027	20	20	161045	157462	161200	161045
TA028	20	20	297133	285199	266705	297133
TA029	20	20	206789	186879	201199	206789
TA030	20	20	234877	229324	237243	234877
TA031	50	5	46002	45904	45281	46002
TA032	50	5	16016	16016	16224	16016
TA033	50	5	21852	22140	22224	21852
TA034	50	5	6024	6120	6024	6024
TA035	50	5	52269	46959	45765	52269
TA036	50	5	9823	9196	6688	9823
TA037	50	5	8233	8197	8233	8233
TA038	50	5	9575	16909	10867	9575
TA039	50	5	7478	8703	10541	7478
TA040	50	5	3950	5484	8680	3950
TA041	50	10	102129	97065	111988	102129
TA042	50	10	48359	57776	53775	48359
TA043	50	10	62436	62720	59688	62436
TA044	50	10	50800	62278	51853	50800
TA045	50	10	31355	46834	42345	31355
TA046	50	10	30219	34672	39777	30219
TA047	50	10	69862	85457	80541	69862
TA048	50	10	94563	108606	105922	94563
TA049	50	10	63311	71116	60589	63311
TA050	50	10	41077	43315	48808	41077
TA051	50	20	322608	292213	319830	322608
TA052	50	20	302104	283426	292980	302104
TA053	50	20	289724	287262	273736	289724
TA054	50	20	340239	339226	322658	340239
TA055	50	20	217839	230617	200226	217839
TA056	50	20	284199	298767	311844	284199
TA057	50	20	350823	374415	362760	350823
TA058	50	20	364778	363250	357740	364778
TA059	50	20	291751	275484	279031	291751
TA060	50	20	259387	297762	272178	259387
TA061	100	5	18842	25088	20009	18842
TA062	100	5	6497	9022	6497	6497
TA063	100	5	3952	7277	4294	3952
TA064	100	5	9314	18632	10232	9314
TA065	100	5	45471	44618	44368	45471
TA066	100	5	6245	13157	23924	6245
TA067	100	5	31692	34786	32892	31692
TA068	100	5	1011	4018	1749	1011

TA069	100	5	6786	6728	6902	6786
TA070	100	5	2260	5440	2933	2260
TA071	100	10	105092	116594	109880	105092
TA072	100	10	86057	92136	91562	86057
TA073	100	10	173525	172905	170389	173525
TA074	100	10	89701	85700	97512	89701
TA075	100	10	132680	132481	127116	132680
TA076	100	10	85921	81099	95920	85921
TA077	100	10	158578	171110	163911	158578
TA078	100	10	175680	199751	192042	175680
TA079	100	10	153977	165189	159919	153977
TA080	100	10	66775	76799	73154	66775
TA081	100	20	336070	359644	340308	336070
TA082	100	20	234512	264991	248627	234512
TA083	100	20	420206	415649	432214	420206
TA084	100	20	447315	488543	486426	447315
TA085	100	20	435795	506935	487117	435795
TA086	100	20	316954	326574	323842	316954
TA087	100	20	563537	655245	567394	563537
TA088	100	20	524430	551106	534518	524430
TA089	100	20	373625	448115	373197	373625
TA090	100	20	483305	473232	429513	483305
TA091	200	10	250688	252215	252859	250688
TA092	200	10	54213	85406	49184	54213
TA093	200	10	197070	210935	210222	197070
TA094	200	10	65368	120767	70017	65368
TA095	200	10	178252	239893	207045	178252
TA096	200	10	77368	124340	103555	77368
TA097	200	10	18491	78813	38178	18491
TA098	200	10	115124	119264	114044	115124
TA099	200	10	45486	65181	40775	45486
TA100	200	10	102263	130337	128297	102263
TA101	200	20	568861	676254	611904	568861
TA102	200	20	380969	500251	485058	380969
TA103	200	20	453436	614807	530865	453436
TA104	200	20	402041	495576	432855	402041
TA105	200	20	730712	828117	738533	730712
TA106	200	20	573002	581935	605172	573002
TA107	200	20	692928	771784	737899	692928
TA108	200	20	691954	749084	716853	691954
TA109	200	20	531402	630160	603709	531402
TA110	200	20	497848	513212	516933	497848
TA111	500	20	638483	765614	577177	638483
TA112	500	20	684005	876174	829067	684005
TA113	500	20	715744	938490	771400	715744
TA114	500	20	657679	734581	779881	657679
TA115	500	20	759709	776448	833914	759709

TA116	500	20	658751	767505	772304	658751
TA117	500	20	1012326	1229684	1145879	1012326
TA118	500	20	832236	1108678	1003917	832236
TA119	500	20	582247	752886	725241	582247
TA120	500	20	601588	828356	709258	601588

Tabla 20. RPD de cada variante según secuencia inicial y regla de desempate

Instancia	Trabajos	Máquinas	RPD NEH	RPD A	RPD B	RPC
TA001	20	5	17,691	209,607	4,006	17,691
TA002	20	5	2,331	0,000	0,585	2,331
TA003	20	5	23,777	269,250	8,385	23,777
TA004	20	5	0,000	46,446	2,339	0,000
TA005	20	5	28,697	290,911	10,494	28,697
TA006	20	5	6,420	69,571	1,811	6,420
TA007	20	5	1,640	23,893	0,000	1,640
TA008	20	5	1,508	11,929	0,340	1,508
TA009	20	5	21,960	231,018	9,420	21,960
TA010	20	5	2,318	111,482	2,332	2,318
TA011	20	10	1,430	1,781	1,680	1,430
TA012	20	10	0,306	0,844	0,187	0,306
TA013	20	10	3,180	3,370	3,223	3,180
TA014	20	10	1,933	2,183	2,445	1,933
TA015	20	10	1,306	1,229	1,384	1,306
TA016	20	10	2,219	2,186	1,961	2,219
TA017	20	10	0,000	0,000	0,000	0,000
TA018	20	10	1,183	1,094	1,610	1,183
TA019	20	10	1,092	1,169	1,139	1,092
TA020	20	10	1,293	1,595	2,193	1,293
TA021	20	20	0,820	1,028	0,845	0,820
TA022	20	20	0,066	0,172	0,070	0,066
TA023	20	20	0,393	0,510	0,417	0,393
TA024	20	20	0,571	0,701	0,498	0,571
TA025	20	20	0,473	0,749	0,521	0,473
TA026	20	20	0,000	0,000	0,000	0,000
TA027	20	20	0,159	0,197	0,126	0,159
TA028	20	20	1,137	1,167	0,863	1,137
TA029	20	20	0,488	0,420	0,406	0,488
TA030	20	20	0,690	0,743	0,657	0,690
TA031	50	5	10,646	7,371	6,517	10,646
TA032	50	5	3,055	1,920	1,693	3,055
TA033	50	5	4,532	3,037	2,689	4,532
TA034	50	5	0,525	0,116	0,000	0,525
TA035	50	5	12,233	7,563	6,597	12,233
TA036	50	5	1,487	0,677	0,110	1,487

TA037	50	5	1,084	0,495	0,367	1,084
TA038	50	5	1,424	2,083	0,804	1,424
TA039	50	5	0,893	0,587	0,750	0,893
TA040	50	5	0,000	0,000	0,441	0,000
TA041	50	10	2,380	1,800	1,815	2,380
TA042	50	10	0,600	0,666	0,352	0,600
TA043	50	10	1,066	0,809	0,501	1,066
TA044	50	10	0,681	0,796	0,304	0,681
TA045	50	10	0,038	0,351	0,065	0,038
TA046	50	10	0,000	0,000	0,000	0,000
TA047	50	10	1,312	1,465	1,025	1,312
TA048	50	10	2,129	2,132	1,663	2,129
TA049	50	10	1,095	1,051	0,523	1,095
TA050	50	10	0,359	0,249	0,227	0,359
TA051	50	20	0,481	0,267	0,597	0,481
TA052	50	20	0,387	0,229	0,463	0,387
TA053	50	20	0,330	0,246	0,367	0,330
TA054	50	20	0,562	0,471	0,611	0,562
TA055	50	20	0,000	0,000	0,000	0,000
TA056	50	20	0,305	0,296	0,557	0,305
TA057	50	20	0,610	0,624	0,812	0,610
TA058	50	20	0,675	0,575	0,787	0,675
TA059	50	20	0,339	0,195	0,394	0,339
TA060	50	20	0,191	0,291	0,359	0,191
TA061	100	5	17,637	5,244	10,440	17,637
TA062	100	5	5,426	1,245	2,715	5,426
TA063	100	5	2,909	0,811	1,455	2,909
TA064	100	5	8,213	3,637	4,850	8,213
TA065	100	5	43,976	10,105	24,368	43,976
TA066	100	5	5,177	2,275	12,679	5,177
TA067	100	5	30,347	7,658	17,806	30,347
TA068	100	5	0,000	0,000	0,000	0,000
TA069	100	5	5,712	0,674	2,946	5,712
TA070	100	5	1,235	0,354	0,677	1,235
TA071	100	10	0,574	0,518	0,502	0,574
TA072	100	10	0,289	0,200	0,252	0,289
TA073	100	10	1,599	1,251	1,329	1,599
TA074	100	10	0,343	0,116	0,333	0,343
TA075	100	10	0,987	0,725	0,738	0,987
TA076	100	10	0,287	0,056	0,311	0,287
TA077	100	10	1,375	1,228	1,241	1,375
TA078	100	10	1,631	1,601	1,625	1,631
TA079	100	10	1,306	1,151	1,186	1,306
TA080	100	10	0,000	0,000	0,000	0,000
TA081	100	20	0,433	0,357	0,369	0,433
TA082	100	20	0,000	0,000	0,000	0,000
TA083	100	20	0,792	0,569	0,738	0,792

TA084	100	20	0,907	0,844	0,956	0,907
TA085	100	20	0,858	0,913	0,959	0,858
TA086	100	20	0,352	0,232	0,303	0,352
TA087	100	20	1,403	1,473	1,282	1,403
TA088	100	20	1,236	1,080	1,150	1,236
TA089	100	20	0,593	0,691	0,501	0,593
TA090	100	20	1,061	0,786	0,728	1,061
TA091	200	10	12,557	2,869	5,623	12,557
TA092	200	10	1,932	0,310	0,288	1,932
TA093	200	10	9,658	2,236	4,506	9,658
TA094	200	10	2,535	0,853	0,834	2,535
TA095	200	10	8,640	2,680	4,423	8,640
TA096	200	10	3,184	0,908	1,712	3,184
TA097	200	10	0,000	0,209	0,000	0,000
TA098	200	10	5,226	0,830	1,987	5,226
TA099	200	10	1,460	0,000	0,068	1,460
TA100	200	10	4,530	1,000	2,360	4,530
TA101	200	20	0,493	0,365	0,414	0,493
TA102	200	20	0,000	0,009	0,121	0,000
TA103	200	20	0,190	0,241	0,226	0,190
TA104	200	20	0,055	0,000	0,000	0,055
TA105	200	20	0,918	0,671	0,706	0,918
TA106	200	20	0,504	0,174	0,398	0,504
TA107	200	20	0,819	0,557	0,705	0,819
TA108	200	20	0,816	0,512	0,656	0,816
TA109	200	20	0,395	0,272	0,395	0,395
TA110	200	20	0,307	0,036	0,194	0,307
TA111	500	20	0,097	0,042	0,000	0,097
TA112	500	20	0,175	0,193	0,436	0,175
TA113	500	20	0,229	0,278	0,337	0,229
TA114	500	20	0,130	0,000	0,351	0,130
TA115	500	20	0,305	0,057	0,445	0,305
TA116	500	20	0,131	0,045	0,338	0,131
TA117	500	20	0,739	0,674	0,985	0,739
TA118	500	20	0,429	0,509	0,739	0,429
TA119	500	20	0,000	0,025	0,257	0,000
TA120	500	20	0,033	0,128	0,229	0,033

Tabla 21. Tiempos de ejecución de cada variante según secuencia inicial y regla de desempate.

Instancia	Trabajos	Máquinas	NEH+TB	NEH (PR A) + TB	NEH (PR B) + TB	NEH (PR C) + TB
TA001	20	5	0,003	0,001	0,001	0,001
TA002	20	5	0,003	0,001	0,001	0,001

TA003	20	5	0,001	0,001	0,001	0,002
TA004	20	5	0,002	0,002	0,001	0,001
TA005	20	5	0,001	0,002	0,001	0,001
TA006	20	5	0,001	0,002	0,001	0,001
TA007	20	5	0,001	0,001	0,001	0,002
TA008	20	5	0,001	0,001	0,001	0,001
TA009	20	5	0,001	0,001	0,001	0,001
TA010	20	5	0,001	0,001	0,001	0,001
TA011	20	10	0,002	0,001	0,001	0,001
TA012	20	10	0,001	0,001	0,002	0,002
TA013	20	10	0,002	0,001	0,001	0,002
TA014	20	10	0,001	0,001	0,002	0,002
TA015	20	10	0,001	0,001	0,001	0,001
TA016	20	10	0,001	0,002	0,002	0,002
TA017	20	10	0,002	0,001	0,001	0,001
TA018	20	10	0,002	0,001	0,001	0,002
TA019	20	10	0,001	0,001	0,001	0,001
TA020	20	10	0,001	0,002	0,001	0,001
TA021	20	20	0,002	0,002	0,002	0,002
TA022	20	20	0,002	0,002	0,003	0,002
TA023	20	20	0,002	0,002	0,002	0,002
TA024	20	20	0,003	0,002	0,002	0,002
TA025	20	20	0,002	0,002	0,003	0,003
TA026	20	20	0,002	0,002	0,002	0,002
TA027	20	20	0,002	0,002	0,002	0,002
TA028	20	20	0,003	0,002	0,002	0,003
TA029	20	20	0,002	0,003	0,002	0,002
TA030	20	20	0,002	0,003	0,003	0,002
TA031	50	5	0,012	0,012	0,012	0,012
TA032	50	5	0,012	0,010	0,010	0,012
TA033	50	5	0,010	0,010	0,010	0,010
TA034	50	5	0,010	0,010	0,010	0,010
TA035	50	5	0,012	0,010	0,010	0,012
TA036	50	5	0,012	0,010	0,010	0,012
TA037	50	5	0,010	0,010	0,012	0,012
TA038	50	5	0,012	0,010	0,010	0,014
TA039	50	5	0,012	0,008	0,010	0,014
TA040	50	5	0,010	0,010	0,010	0,012
TA041	50	10	0,016	0,017	0,017	0,016
TA042	50	10	0,014	0,016	0,017	0,016
TA043	50	10	0,017	0,016	0,017	0,017
TA044	50	10	0,016	0,016	0,017	0,016
TA045	50	10	0,014	0,014	0,016	0,016
TA046	50	10	0,017	0,017	0,017	0,016
TA047	50	10	0,016	0,017	0,017	0,017
TA048	50	10	0,017	0,016	0,017	0,017
TA049	50	10	0,016	0,018	0,017	0,017

TA050	50	10	0,014	0,018	0,016	0,014
TA051	50	20	0,025	0,025	0,026	0,025
TA052	50	20	0,026	0,025	0,025	0,028
TA053	50	20	0,025	0,025	0,025	0,025
TA054	50	20	0,025	0,025	0,025	0,025
TA055	50	20	0,025	0,025	0,025	0,025
TA056	50	20	0,029	0,025	0,025	0,029
TA057	50	20	0,028	0,028	0,025	0,026
TA058	50	20	0,025	0,025	0,026	0,025
TA059	50	20	0,029	0,025	0,025	0,025
TA060	50	20	0,025	0,026	0,026	0,025
TA061	100	5	0,075	0,084	0,079	0,076
TA062	100	5	0,077	0,079	0,073	0,075
TA063	100	5	0,075	0,081	0,074	0,074
TA064	100	5	0,080	0,083	0,079	0,079
TA065	100	5	0,080	0,075	0,084	0,075
TA066	100	5	0,069	0,084	0,076	0,070
TA067	100	5	0,082	0,075	0,079	0,080
TA068	100	5	0,064	0,075	0,063	0,065
TA069	100	5	0,076	0,079	0,076	0,079
TA070	100	5	0,079	0,08	0,076	0,081
TA071	100	10	0,120	0,120	0,112	0,116
TA072	100	10	0,117	0,108	0,122	0,119
TA073	100	10	0,137	0,126	0,140	0,138
TA074	100	10	0,117	0,138	0,119	0,116
TA075	100	10	0,113	0,130	0,127	0,116
TA076	100	10	0,123	0,148	0,137	0,121
TA077	100	10	0,159	0,153	0,163	0,157
TA078	100	10	0,118	0,118	0,121	0,115
TA079	100	10	0,110	0,113	0,105	0,113
TA080	100	10	0,128	0,113	0,125	0,129
TA081	100	20	0,203	0,198	0,194	0,198
TA082	100	20	0,180	0,191	0,184	0,186
TA083	100	20	0,188	0,186	0,190	0,192
TA084	100	20	0,203	0,199	0,196	0,196
TA085	100	20	0,204	0,196	0,196	0,203
TA086	100	20	0,200	0,194	0,201	0,199
TA087	100	20	0,194	0,188	0,190	0,191
TA088	100	20	0,190	0,192	0,196	0,189
TA089	100	20	0,199	0,195	0,192	0,190
TA090	100	20	0,192	0,199	0,195	0,195
TA091	200	10	1,052	0,981	1,042	1,040
TA092	200	10	0,878	0,945	0,851	0,895
TA093	200	10	1,050	1,098	1,056	1,038
TA094	200	10	0,816	0,979	0,898	0,820
TA095	200	10	0,856	1,034	0,897	0,854

TA096	200	10	0,86	1,082	0,967	0,881
TA097	200	10	0,813	0,916	0,852	0,819
TA098	200	10	0,875	0,972	0,831	0,887
TA099	200	10	0,853	1,006	0,949	0,856
TA100	200	10	0,915	0,932	0,957	0,929
TA101	200	20	1,453	1,504	1,445	1,449
TA102	200	20	1,374	1,437	1,433	1,387
TA103	200	20	1,379	1,423	1,407	1,393
TA104	200	20	1,443	1,536	1,459	1,443
TA105	200	20	1,407	1,427	1,423	1,411
TA106	200	20	1,413	1,467	1,441	1,415
TA107	200	20	1,375	1,362	1,481	1,391
TA108	200	20	1,435	1,580	1,600	1,449
TA109	200	20	1,375	1,469	1,457	1,391
TA110	200	20	1,441	1,465	1,437	1,455
TA111	500	20	21,028	22,339	21,018	20,871
TA112	500	20	21,001	21,312	21,417	20,955
TA113	500	20	20,445	23,327	21,647	20,565
TA114	500	20	20,939	23,192	21,053	21,216
TA115	500	20	21,391	22,955	21,370	21,708
TA116	500	20	21,19	22,685	21,670	21,449
TA117	500	20	21,294	23,364	22,245	21,437
TA118	500	20	21,963	22,524	22,280	22,039
TA119	500	20	20,012	23,073	21,431	20,243
TA120	500	20	22,399	24,236	23,592	22,492

Tabla 22. Comparativa conjunta de la regla de desempate en cada una de las variantes de la NEH según secuencia inicial empleada.

RPD NEH	RPD NEH (PR A)	RPD NEH (PR B)	RPD NEH (PR C)	RPD NEH + TB	RPD NEH (PR A) + TB	RPD NEH (PR B) + TB	RPD NEH (PR C) + TB
17,691	209,607	4,006	13,392	17,691	209,607	4,006	17,691
2,331	1,857	0,789	7,186	2,331	0,000	0,585	2,331
23,777	269,250	8,385	25,288	23,777	269,250	8,385	23,777
2,117	46,446	2,339	5,203	0,000	46,446	2,339	0,000
28,697	298,625	10,494	32,468	28,697	290,911	10,494	28,697
6,420	69,571	1,811	9,330	6,420	69,571	1,811	6,420
1,640	30,768	0,000	3,830	1,640	23,893	0,000	1,640
2,000	13,982	0,447	2,000	1,508	11,929	0,340	1,508
21,960	250,089	9,420	30,275	21,960	231,018	9,420	21,960
2,318	111,482	2,332	9,867	2,318	111,482	2,332	2,318
1,430	1,781	1,680	1,737	1,430	1,781	1,680	1,430
0,306	0,844	0,364	0,622	0,306	0,844	0,187	0,306
3,180	3,370	3,223	3,706	3,180	3,370	3,223	3,180
1,933	2,144	2,445	2,466	1,933	2,183	2,445	1,933

1,302	1,229	1,384	1,488	1,306	1,229	1,384	1,306
2,219	2,109	1,961	1,945	2,219	2,186	1,961	2,219
0,000	0,000	0,000	0,213	0,000	0,000	0,000	0,000
1,183	1,097	1,679	0,744	1,183	1,094	1,610	1,183
1,092	1,169	1,139	1,625	1,092	1,169	1,139	1,092
1,293	1,595	2,193	1,225	1,293	1,595	2,193	1,293
0,820	1,028	0,845	1,637	0,820	1,028	0,845	1,122
0,066	0,172	0,070	0,203	0,066	0,172	0,070	0,243
0,393	0,510	0,417	0,725	0,393	0,510	0,417	0,624
0,574	0,701	0,503	0,706	0,571	0,701	0,498	0,831
0,473	0,749	0,521	1,015	0,473	0,749	0,521	0,717
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,166
0,214	0,197	0,126	0,506	0,159	0,197	0,126	0,351
1,131	1,167	0,863	1,423	1,137	1,167	0,863	1,492
0,488	0,420	0,406	0,651	0,488	0,420	0,406	0,734
0,690	0,743	0,657	1,016	0,690	0,743	0,657	0,970
10,646	7,279	6,517	10,815	10,646	7,371	6,517	10,646
3,055	1,922	1,693	3,032	3,055	1,920	1,693	3,055
4,532	3,120	2,681	4,383	4,532	3,037	2,689	4,532
0,525	0,098	0,000	0,574	0,525	0,116	0,000	0,525
12,233	7,563	7,838	11,360	12,233	7,563	6,597	12,233
1,021	0,601	0,110	2,256	1,487	0,677	0,110	1,487
1,084	0,506	0,346	1,135	1,084	0,495	0,367	1,084
1,424	2,083	0,658	2,582	1,424	2,083	0,804	1,424
0,896	0,476	0,745	1,349	0,893	0,587	0,750	0,893
0,000	0,000	0,441	0,579	0,000	0,000	0,441	0,000
2,047	1,668	1,449	2,150	2,499	1,800	1,815	2,380
0,452	0,865	0,286	1,866	0,657	0,666	0,352	0,600
1,136	0,783	0,525	1,311	1,139	0,809	0,501	1,066
0,787	0,606	0,363	1,026	0,740	0,796	0,304	0,681
0,290	0,373	0,065	0,865	0,074	0,351	0,065	0,038
0,000	0,030	0,000	0,068	0,035	0,000	0,000	0,000
1,471	1,389	1,023	1,510	1,394	1,465	1,025	1,312
2,209	2,026	1,765	2,323	2,240	2,132	1,663	2,129
1,164	0,686	0,523	1,519	1,169	1,051	0,523	1,095
0,351	0,386	0,319	0,845	0,407	0,249	0,227	0,359
0,481	0,267	0,597	0,469	0,481	0,267	0,597	0,481
0,387	0,229	0,463	0,514	0,387	0,229	0,463	0,387
0,359	0,246	0,367	0,280	0,330	0,246	0,367	0,330
0,562	0,471	0,611	0,618	0,562	0,471	0,611	0,562
0,000	0,055	0,000	0,114	0,000	0,000	0,000	0,000
0,390	0,360	0,557	0,526	0,305	0,296	0,557	0,305
0,593	0,628	0,812	0,710	0,610	0,624	0,812	0,610
0,675	0,575	0,787	0,685	0,675	0,575	0,787	0,675
0,353	0,195	0,404	0,577	0,339	0,195	0,394	0,339
0,257	0,278	0,359	0,301	0,191	0,291	0,359	0,191

18,325	5,302	15,939	26,977	18,325	5,244	14,632	17,637
6,590	1,280	4,076	12,595	5,664	1,245	4,076	5,426
3,404	0,731	2,370	15,237	3,053	0,811	2,355	2,909
8,553	3,341	8,344	21,348	8,553	3,637	6,994	8,213
45,637	10,105	33,859	43,014	45,637	10,105	33,663	43,976
5,405	2,664	17,691	10,919	5,405	2,275	17,691	5,177
31,505	7,796	24,697	56,312	31,505	7,658	24,697	30,347
0,000	1,290	0,000	10,125	0,037	0,000	0,366	0,000
5,960	0,830	8,273	16,751	5,960	0,674	4,392	5,712
1,355	0,248	1,214	8,402	1,318	0,354	1,291	1,235
0,549	0,590	0,519	1,041	0,574	0,518	0,502	0,574
0,432	0,307	0,264	0,633	0,289	0,200	0,252	0,289
1,599	1,209	1,327	1,820	1,599	1,251	1,329	1,599
0,238	0,309	0,321	0,771	0,343	0,116	0,333	0,343
0,982	0,770	0,765	1,460	0,987	0,725	0,738	0,987
0,225	0,152	0,285	0,416	0,287	0,056	0,311	0,287
1,405	1,342	1,481	1,782	1,375	1,228	1,241	1,375
1,834	1,419	1,685	2,110	1,631	1,601	1,625	1,631
1,387	1,164	1,276	1,337	1,306	1,151	1,186	1,306
0,059	0,003	0,055	0,319	0,000	0,000	0,000	0,000
0,502	0,288	0,403	0,739	0,433	0,357	0,369	0,433
0,024	0,000	0,030	0,254	0,000	0,000	0,000	0,000
0,850	0,599	0,699	0,857	0,792	0,569	0,738	0,792
0,904	0,832	0,883	1,170	0,907	0,844	0,956	0,907
0,942	0,862	0,998	1,224	0,858	0,913	0,959	0,858
0,274	0,259	0,147	0,335	0,352	0,232	0,303	0,352
1,403	1,353	1,383	1,744	1,403	1,473	1,282	1,403
1,227	1,135	1,187	1,668	1,236	1,080	1,150	1,236
0,560	0,617	0,517	0,895	0,593	0,691	0,501	0,593
1,088	0,823	0,790	1,321	1,061	0,786	0,728	1,061
12,825	4,283	5,543	15,194	12,886	3,677	5,623	12,557
2,101	0,480	0,456	4,016	2,003	0,584	0,288	1,932
10,070	2,964	4,557	11,080	9,916	2,912	4,506	9,658
2,552	0,972	1,086	6,739	2,621	1,240	0,834	2,535
10,410	2,764	4,409	14,315	8,874	3,449	4,423	8,640
3,914	1,695	1,899	4,875	3,286	1,306	1,712	3,184
0,000	0,000	0,459	3,020	0,024	0,462	0,000	0,000
5,477	1,512	2,557	6,428	5,377	1,212	1,987	5,226
1,406	0,567	0,341	4,208	1,520	0,209	0,068	1,460
5,186	1,532	2,623	8,838	4,665	1,417	2,360	4,530
8,721	0,365	0,450	0,725	8,554	0,365	0,414	0,493
5,908	0,074	0,093	0,698	5,398	0,009	0,121	0,000
7,291	0,179	0,355	0,569	6,616	0,241	0,226	0,190
5,754	0,030	0,111	0,229	5,752	0,000	0,000	0,055
11,058	0,668	0,714	1,248	11,272	0,671	0,706	0,918
0,000	0,291	0,494	0,737	8,624	0,174	0,398	0,504
10,170	0,548	0,648	1,025	10,638	0,557	0,705	0,819

10,402	0,595	0,730	1,102	10,621	0,512	0,656	0,816
8,496	0,298	0,317	0,842	7,925	0,272	0,395	0,395
8,257	0,189	0,245	0,594	7,361	0,036	0,194	0,307
0,128	0,097	0,203	0,732	0,127	0,075	0,000	0,097
0,266	0,279	0,394	0,603	0,207	0,231	0,436	0,175
0,311	0,282	0,427	0,774	0,263	0,318	0,337	0,229
0,269	0,000	0,431	0,591	0,161	0,032	0,351	0,130
0,384	0,215	0,363	0,900	0,341	0,090	0,445	0,305
0,029	0,094	0,352	0,642	0,162	0,078	0,338	0,131
0,845	0,758	1,050	1,290	0,786	0,727	0,985	0,739
0,412	0,552	0,496	0,834	0,469	0,557	0,739	0,429
0,000	0,014	0,139	0,548	0,027	0,057	0,257	0,000
0,077	0,188	0,299	0,584	0,062	0,163	0,229	0,033

BIBLIOGRAFÍA

Bierer, Annett, and Uwe Götze. 2011. *A Target Costing-Based Approach for Design to Energy Efficiency*. *Glocalized Solutions for Sustainability in Manufacturing - Proceedings of the 18th CIRP International Conference on Life Cycle Engineering*. https://doi.org/10.1007/978-3-642-19692-8_110.

Escuela Superior de Ingenieros de la Universidad de Sevilla. 2007. "Técnicas y Herrramientas Para La Gestión de Proyectos." *Métodos y Técnicas Para La Gestión de Proyectos Software*, 1–29. http://bibing.us.es/proyectos/abreproy/70193/fichero/4.+TÉCNICAS+Y+HERRAMIENTAS+PARA+LA+GESTIÓN+DE+PROYECTOS.pdf.

Fang, Kan, Nelson A. Uhan, Fu Zhao, and John W. Sutherland. 2013. "Flow Shop Scheduling with Peak Power Consumption Constraints." *Annals of Operations Research* 206 (1): 115–45. https://doi.org/10.1007/s10479-012-1294-z.

Fernandez-Viagas, Victor, and Jose M. Framinan. 2014. "On Insertion Tie-Breaking Rules in Heuristics for the Permutation Flowshop Scheduling Problem." *Computers and Operations Research* 45: 60–67. https://doi.org/10.1016/j.cor.2013.12.012.

Framinan, J. M., R. Leisten, and C. Rajendran. 2003. "Different Initial Sequences for the Heuristic of Nawaz, Enscore and Ham to Minimize Makespan, Idletime or Flowtime in the Static Permutation Flowshop Sequencing Problem." *International Journal of Production Research* 41 (1): 121–48. https://doi.org/10.1080/00207540210161650.

Framinan, Jose M., Rainer Leisten, and Rubén Ruiz García. 2014. *Manufacturing Scheduling Systems*. *Manufacturing Scheduling Systems*. https://doi.org/10.1007/978-1-4471-6272-8.

Framiñan Torres, Jose Manuel, Paz Pérez González, and Víctor Fernández-Viagas Escudero. 2020. *Programación de Operaciones*. Sevilla.

Graham, R L, E L Lawler, J. K. Lenstra, and A H G Rinnooy Kan. 1979. "Optimization and Heuristic in Deterministic Sequencing and Scheduling: A Survey." *Annals of Discrete Mathematics* 5: 287–326. https://ac.els-cdn.com/S016750600870356X/1-s2.0-S016750600870356X-main.pdf?_tid=cbf345a3-808d-42df-9560-bf8a52069d0e&acdnat=1550949881_e9837bfdf6316caefaf71ae2f3779b10.

Liu, Weibo, Yan Jin, and Mark Price. 2017. "A New Improved NEH Heuristic for Permutation Flowshop Scheduling Problems." *International Journal of Production Economics* 193 (June 2016): 21–30. https://doi.org/10.1016/j.ijpe.2017.06.026.

Medero, Juan Manuel Rodríguez. 2012. "Metodología de Diseño Del Layout," 39–71.

Nawaz, Muhammad, E. Emory Enscore, and Inyong Ham. 1983. "A Heuristic Algorithm for the M-Machine, n-Job Flow-Shop Sequencing Problem." *Omega* 11 (1): 91–95. https://doi.org/10.1016/0305-0483(83)90088-9.

Öztop, Hande, M. Fatih Tasgetiren, Deniz Türsel Eliiyi, Quan Ke Pan, and Levent Kandiller. 2020. "An Energy-Efficient Permutation Flowshop Scheduling Problem." *Expert Systems with Applications* 150. https://doi.org/10.1016/j.eswa.2020.113279.

Pinedo, Michael L. 2012. *Scheduling. Theory, Algorithms, and Systems*. Fourth Edi.

Taillard, E. 1993. "Benchmarks for Basic Scheduling Problems." *European Journal of Operational Research* 64 (2): 278–85. https://doi.org/10.1016/0377-2217(93)90182-M.