Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Minimización de los tiempos ociosos de las máquinas en el taller de flujo de permutación

Autor: Juan Raúl Castaño Huertas

Tutor: Paz Pérez González

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

Sevilla, 2020

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Minimización de los tiempos ociosos de las máquinas en el taller de flujo de permutación

Autor:
Juan Raúl Castaño Huertas

Tutor:
Paz Pérez González
Profesor Titular

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

Sevilla, 2020

Trabajo de Fin de Grado: Minimización de los tiempos ociosos de las máquinas en el taller de flujo de permutación

Autor: Juan Raúl Castaño Huertas
Tutor: Paz Pérez González

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mi abuelo, por estar conmigo durante todo el camino.

Juan Raúl Castaño Huertas
Sevilla, 2020

Resumen

En este Trabajo de Fin de Grado se aborda el problema de la programación de operaciones en un entorno de taller de flujo regular con permutación (*permutation flowshop scheduling*) proponiendo un método de resolución en el caso de la minimización de tiempo ocioso de las máquinas. Para ello, se hará una revisión de los conceptos fundamentales que involucra el problema, así como de algunos de los métodos utilizados para resolverlo para otros objetivos. Finalmente, se adaptan algunos de los métodos ya existentes y se desarrollan nuevas propuestas, analizando tanto la calidad de las soluciones obtenidas como los tiempos de ejecución de los algoritmos utilizados.

Abstract

The objective of this Final Degree Proyect is to solve the permutation flowshop scheduing problem with the objective of minimizing the total core idle times. In order to achieve it, a review of the fundamental concepts involving the problem will be made, as well as some of the most used algorithms to solve the problem with other objectives. Finally, some of the existing methods in the literature are adapted, and some new proposals are devoloped, analyzing both the quality of the solutions obtained and the CPU times.

Índice

Índice de Figuras

Índice de Tablas

1 Introducción y objetivos

En este capítulo se pretende introducir al lector a la temática de la programación de operaciones, así como ponerle en contexto en cuanto al objetivo del problema que se pretende resolver y su justificación. También se explican tanto la estructura que sigue el propio documento como el contenido en cada una de sus partes.
Asimismo, a medida que se avance en la lectura, se irá introduciendo terminología en inglés, siendo esta identificada por aparecer en cursiva, y que será explicada a medida que vaya apareciendo. Por último, se hará uso de siglas, pertenecientes a sus respectivas expresiones en inglés, habiendo sido todas y cada una de ellas definidas con anterioridad.

La programación de operaciones (*operation scheduling*) ocupa un papel muy importante en la industria, ya que puede suponer importantes mejoras en el desempeño de una empresa, sea cual sea su campo de producción (sector automovilístico, productos tecnológicos, etc.). Aumentos en el rendimiento, tales como el aumento de la productividad a través de la disminución de los tiempos de proceso totales, la disminución de los tiempos ociosos de las máquinas entre trabajos o de los retrasos en los trabajos son algunas de las ventajas que se pueden lograr con una eficiente programación de las operaciones de una empresa.

Por otra parte, el entorno de taller de flujo uniforme (*flowshop*), es uno de los tipos de problema más extendidos en la actualidad, debido a que es el que más se asemeja a la configuración real de la mayor parte de los sistemas productivos de la industria.

1.1 Objetivos del proyecto

En este trabajo se tiene como principal objetivo analizar la eficiencia de distintos algoritmos, cuyo propósito es minimizar la suma ponderada entre el máximo tiempo de terminación de una serie de trabajos en flujo uniforme con permutación (*permutation flowshop scheduling*) y la suma de los tiempos ociosos de las máquinas entre trabajos (*core idle time*). Para ello, se propone un nuevo algoritmo, que será comparado con algunos de los mejores algoritmos presentes en la actualidad: el propuesto por Víctor Fernández – Viagas y José Manuel Framiñán (Fernández – Viagas, Framiñán, 2014), Weibo Liu, Yan Jin y Mark Price (Weibo Liu, Yan Jin, Mark Price, 2016) y el algoritmo en que ambos métodos están basados, propuesto por Muhammad Nawaz, E. Emory Enscore e Inyong Ham (Nawaz, Enscore, Ham, 1983).

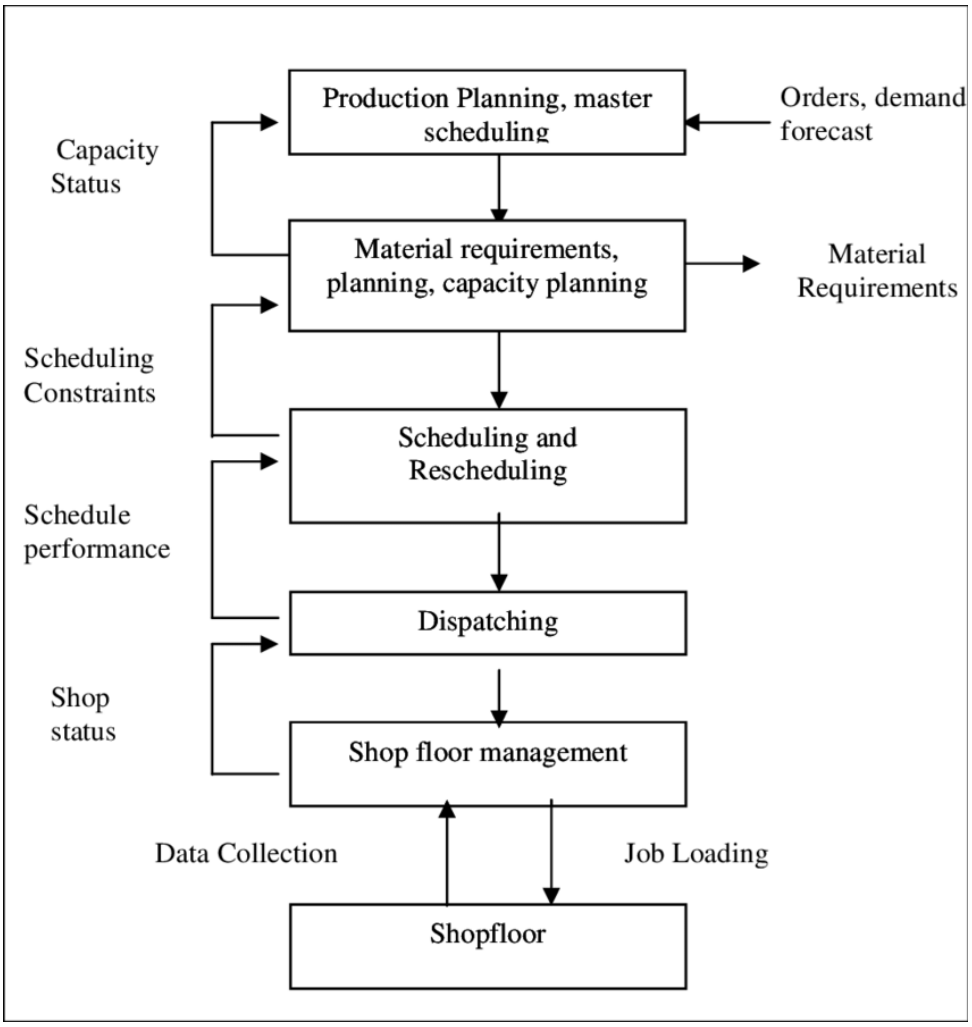


Figura 1.1 Diagrama de flujo de información en un sistema de fabricación.

Para comprobar la eficiencia de los métodos anteriormente mencionados se compararán la calidad de las soluciones obtenidas mediante la desviación porcentual relativa promedio (*Average relative percentage deviation*) o ARPD, así como los tiempos de ejecución de cada uno de ellos.

1.2 Justificación

Mediante la correcta programación de operaciones se puede lograr una disminución significativa en el tiempo total de proceso, lo que se traduce en un ahorro de costes para la empresa. Además, atendiendo al criterio de la disminución de los tiempos ociosos de las máquinas, se consigue aumentar la eficiencia de cada una de las máquinas por separado.

El estudio de este tipo de problemas atiende a la necesidad de disminuir el tiempo y los recursos necesarios para resolverlos de forma óptima, ya que muchos de los problemas de tipo *flowshop* pertenecen a la clasificación de no polinomiales o NP-*hard*, que como será explicado posteriormente, necesitan de una basta cantidad de tiempo y recursos para encontrar una solución óptima al problema a tratar. Es por eso por lo que se hace uso de algoritmos que permitan encontrar una solución lo suficientemente próxima a la solución óptima del problema en una cantidad de tiempo razonable.

1.3 Estructura del documento

El presente documento está estructurado en 6 capítulos, a lo largo de los cuales se desarrollarán los siguientes puntos:

En el primer capítulo se hace una breve introducción del problema a tratar en este trabajo y los objetivos que se pretenden, pasando por la justificación del mismo. Por último, se explica la estructura que sigue el documento.

En el capítulo segundo se describe el problema, tratando aspectos tales como la notación, el entorno, las restricciones a los que está sujeto y el objetivo a optimizar. Previamente a la presentación de dichas particularidades se hará una introducción a la programación de la producción, con el fin de ayudar al lector a entender de una mejor manera el contexto en el que se sitúa el problema.

En el tercer capítulo se realiza una descripción de los algoritmos que han sido puestos en práctica para la comparación de las distintas soluciones al problema, explicando los pasos que sigue cada uno de ellos.

En el cuarto capítulo se introduce *Matlab*, el *software* utilizado para implementar los 4 algoritmos utilizados para los experimentos, calcular y comparar las soluciones obtenidas. También se describen las instancias utilizadas para comparar dichos algoritmos, así como los indicadores usados para evaluar las soluciones.

En el capítulo quinto se muestran y analizan los resultados obtenidos.

Por último, en el sexto capítulo se exponen las conclusiones extraídas de los resultados mostrados en el capítulo 5.

2 Descripción del problema

En este capítulo se describirá el problema objetivo de este trabajo. Tal y como se ha mencionado en el capítulo 1, se realizará una introducción previa a la programación de la producción, para posteriormente abordar las distintas particularidades del problema.

2.1 Programación de la producción

La programación de la producción (*manufacturing scheduling*) es la rama de la organización de la producción (*production management*) que se encarga de asignar una serie de trabajos o recursos en el tiempo con el fin de hacer dicho proceso de producción más eficiente, ya sea disminuyendo los tiempos totales de proceso, la utilización de recursos o los costes para la empresa.

Dicha asignación da lugar a un programa (*production schedule*), que identifica cada trabajo (*job*) e indica cuándo debe ser procesado en cada máquina (*machine*). Una de las herramientas más utilizadas para representar dichos programas de forma gráfica es el diagrama de Gantt, desarrollado por Henry Laurence Gantt entre 1910 y 1915, donde se representan las máquinas o las operaciones que componen un trabajo (vertical) frente al tiempo (horizontal). A partir de este diagrama pueden identificarse tanto las fechas de comienzo y fin de cada trabajo como su duración.

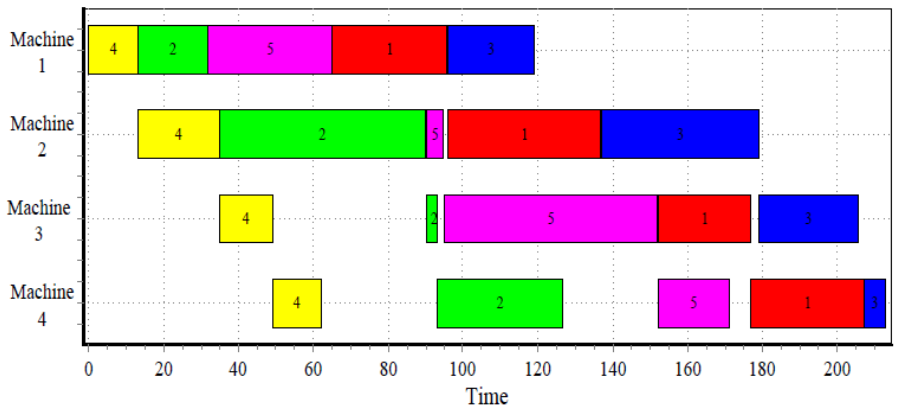


Figura 2.1 Diagrama de Gantt [4].

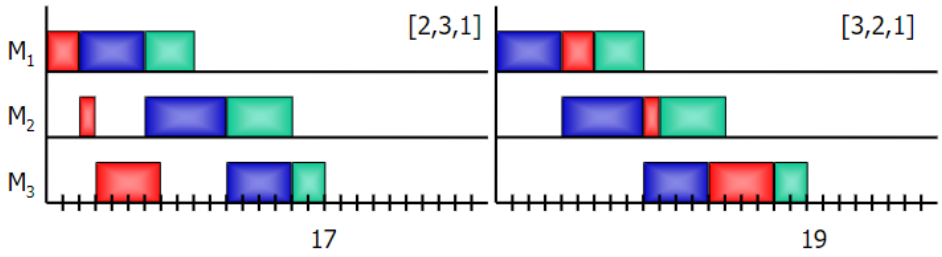


Figura 2.2 Diagrama de Gantt de dos secuencias diferentes [5].

La programación de la producción es catalogada como un proceso de toma de decisiones, y es ampliamente utilizado en el ámbito industrial, tanto en producción y ensamblaje como en la planificación de proyectos. Su cometido es el de encontrar al menos un programa admisible (*feasible schedule*) que logre optimizar cierto objetivo, atendiendo al mismo tiempo a las restricciones que envuelven al problema en cuestión.

En el caso de que se hubieran encontrado más de un programa admisible, se deberá elegir aquel cuya solución mejor satisfaga el objetivo del problema.

2.2 Algoritmos

El método utilizado para obtener un programa admisible recibe el nombre de algoritmo. Un algoritmo es una sucesión finita de operaciones ordenadas que tiene como objetivo la obtención de una solución a un problema. Dicha solución puede ser exacta o aproximada.

Se entiende como solución exacta aquella que proporciona un resultado óptimo a un problema, mientras que una solución aproximada ofrecerá un resultado que, aun a pesar de no ser el óptimo, se le acerca de una forma (más o menos) razonable. Los algoritmos de aproximación resultan especialmente interesantes en aquellos casos en los que calcular la solución óptima de un problema requiere una gran cantidad de recursos. Es el caso del problema de *flowshop* tratado en el presente trabajo, del que se hablará en la sección 2.3 de este capítulo.

2.3 Conceptos básicos

A continuación, se definen algunos de los elementos básicos que componen un problema de programación de la producción, junto con la notación que se va a usar para designarlas en este trabajo:

- Máquina (*machine*): "Recurso productivo con capacidad para realizar operaciones de transformación/transporte de material" [4]. Hay un total de m máquinas, designadas por el subíndice i.
- Trabajo (*job*): "Producto que es objeto de una operación en alguna de las máquinas de la fábrica" [4]. Hay un total de n trabajos, designados por el subíndice j.
- Ruta (*route*): La ruta indica el orden que debe seguir un trabajo a la hora de procesarse. Es un vector de m componentes, que contiene los números asociados a cada una de las m máquinas. Existen un total de n rutas, una por cada trabajo, designadas mediante la notación R_j.
  De este modo, tomando como ejemplo la ruta R_2=(3,1,2), el trabajo 2 deberá procesarse primero en la máquina 3, después en la 1 y por último en la 2.
- Secuencia (*sequence*): Es un vector formado por n componentes. Contiene los números asociados a cada uno de los n trabajos que, por orden, deben procesarse en cada máquina.

Puede haber un total de m máquinas o únicamente una, en caso de que todos los trabajos sean procesados en el mismo orden. La secuencia se designa mediante la notación S_i.

- Tiempo de proceso (*processing time*): El tiempo de proceso p_{ij} indica el tiempo que el trabajo j tarda en procesarse en la máquina i.

p_{ij}	job (j)			
machine (i)	1	2	3	4
1	3	3	3	2
2	3	2	4	0
3	2	3	1	3

Figura 2.3 Ejemplo de diferentes tiempos de proceso para cada trabajo en las distintas máquinas [4].

- Fecha de llegada (*release date*): Instante de tiempo en el cual el trabajo j pasa a estar disponible para procesarse, representado por r_j.
- Fecha de entrega (*due date*): La fecha de entrega d_j es el instante de tiempo en el que, como máximo, un trabajo debe acabar de procesarse.
- Peso (*weight*): Sirve como método de ponderación para indicar la importancia de un trabajo. Se designa mediante la notación w_j.
- Tiempo de terminación (*completion time*): Instante de tiempo en el que el trabajo j termina de procesarse en la última máquina. Representado por C_j.
- Tiempo de flujo (*flowtime*): Tiempo total que el trabajo permanece en el entorno, desde que está disponible hasta que termina de procesarse en la última máquina. Se define como $F_j = C_j - r_j$.
- Retraso (*lateness*): Retraso del trabajo j, definido como $L_j = C_j - d_j$.

2.4 Entorno, restricciones y objetivo

Para poder definir adecuadamente un problema de programación de la producción, primero es necesario atender a 3 conceptos fundamentales: el entorno, las restricciones y el objetivo. Para ello se introduce la notación $\alpha|\beta|\gamma$, propuesta en 1917 por Graham, Lawler, Lenstra y Rinnooy Kann (R.L. Graham, E.L. Lawler, J.K Lenstra, A.H.G. Rinnooy Kan, 1979), donde las variables α, β y γ representan el entorno, las restricciones y el objetivo del modelo, respectivamente.

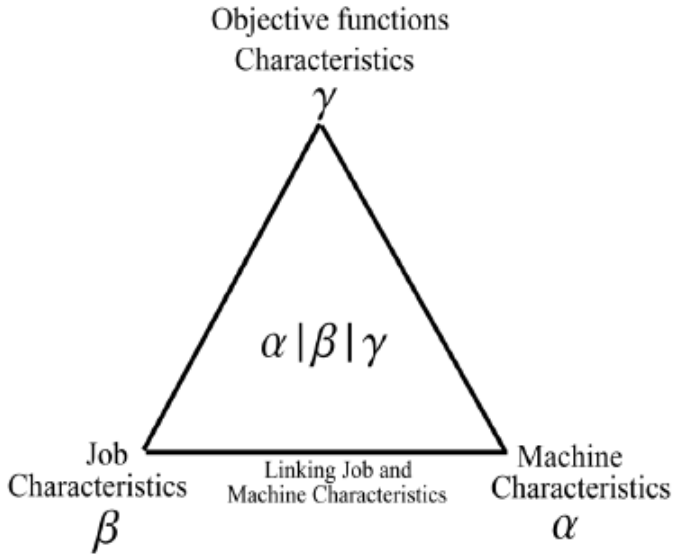


Figura 2.4 Clasificación de los modelos de programación de la producción [4].

2.4.1 Entorno

El entorno, designado por la variable α, define las máquinas involucradas en el proceso y sus características. A continuación se presentan los diferentes entornos, haciendo una breve descripción de cada uno de ellos:

- Una máquina o *single machine* (α=1): Los trabajos son procesados por una única máquina.
- Máquinas en paralelo o *parallel machines*: Entorno formado por *m* máquinas, donde cada trabajo puede procesarse en una de ellas. Existen a su vez 3 tipos de entorno:
  - *Identical parallel machines* (α=P): Todas las máquinas son iguales entre sí.
  - *Uniform parallel machines* (α=Q): Formado por máquinas con diferentes velocidades, lo que origina que haya distintos tiempos de proceso para cada trabajo según la máquina en la que se procese, variando estos de forma proporcional a la velocidad de la máquina.
  - *Unrelated parallel machines* (α=R): Todas las máquinas son distintas entre sí, haciendo que cada trabajo tenga un tiempo de proceso distinto según la máquina a la que se asigne.
- Taller de flujo uniforme o *flowshop* (α=F): Todos los trabajos tienen la misma ruta, es decir, deben pasar por las distintas máquinas en el mismo orden. Un caso particular de este tipo de entorno es el taller de flujo uniforme con permutación o *permutation flowshop*, en el que todas las máquinas tienen la misma secuencia.

- Taller de trabajos o *jobshop* (α=J): Cada uno de los trabajos tiene una ruta diferente, esto es, cada trabajo pasará por las distintas máquinas siguiendo un orden particular.
- Taller abierto o *openshop* (α=O): Entorno de taller en el que no existe una ruta predeterminada.

A la hora de definir el entorno en un problema, también es necesario especificar el número de máquinas fijadas. Para ello se suelen utilizar los subíndices α_1 y α_2, que sirven para hacer referencia al tipo de entorno en sí (α_1) y al número de máquinas fijadas (α_2). Por ejemplo, para el caso de un taller de flujo uniforme con 3 máquinas se utilizará la notación $\alpha = F3$.

2.4.2 Restricciones

En segundo lugar se encuentran las restricciones del modelo, definidas por la letra β. Algunas de las restricciones más comunes en problemas de programación de operaciones son las siguientes:

- Interrupciones ($\beta = pmtn$): Los trabajos pueden interrumpirse una vez se han empezado a procesar en una máquina. Se pueden diferenciar 3 tipos de interrupciones:
  - *Non-resumable* ($\beta = pmtn - non - resumable$): Se pierde todo el progreso del trabajo interrumpido, teniendo que reiniciar el proceso por completo tras la interrupción.
  - *Semi-resumable* ($\beta = pmtn - semi - resumable$): Sólo se pierde una parte del progreso tras la interrupción.
  - *Resumable* ($\beta = pmtn - resumable$): La interrupción del trabajo no influye en su progreso.
- Fechas de llegada ($\beta = r_j$): Indican el instante en el que cada trabajo está disponible para ser procesado.
- Fechas de entrega ($\beta = d_j$): Indican el instante en el que, como máximo, cada trabajo debe ser entregado para que no conlleve ningún retraso.
- Tiempos de *setup* ($\beta = s_{ij}$): Son pausas que se realizan antes de procesar un trabajo. Los tiempos de *setup* pueden ser independientes de la secuencia ($\beta = s_{ijk}$), dependientes de la secuencia (=sijk) o independientes de la máquina ($\beta = s_{jk}$).
- Procesado en lotes ($\beta = batch$): Cada máquina puede procesar lotes de hasta b trabajos.
  - Lotes en paralelo ($\beta = p - batch(b)$): Los trabajos del lote se procesan de manera simultánea, siendo el tiempo de procesado del lote igual al mayor de los tiempos de proceso de los trabajos que lo componen.
  - Lotes en serie ($\beta = s - batch(b)$): El tiempo de procesado del lote es igual a la suma de los tiempos de proceso de los trabajos que lo componen.
- Relaciones de precedencia ($\beta = prec$): No pueden procesarse nuevos trabajos hasta que sus predecesores hayan terminado.
- Permutación ($\beta = prmu$): Restricción particular de los entornos de tipo flowshop, donde la secuencia es la misma para todos los trabajos (S=(1,2,3,...n)).

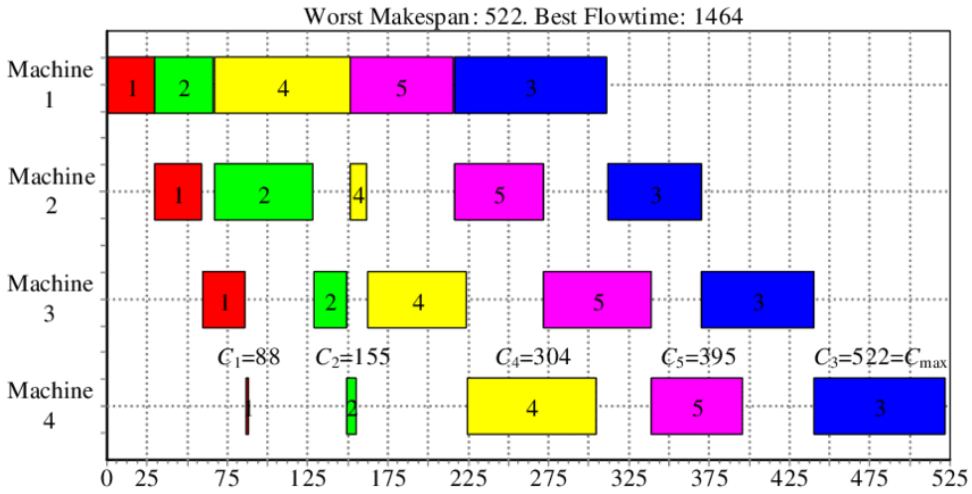


Figura 2.5 Diagrama de Gantt de un problema de *flowshop* con restricción de permutación ($\beta = prmu$) [4].

- Tiempos ociosos no permitidos ($\beta = no-idle$): No se permiten tiempos ociosos en las máquinas, es decir, una vez se ha comenzado a procesar el primer trabajo, esta no puede detenerse hasta haber terminado de procesarlos todos.
- Espera no permitida ($\beta = no-wait$): La espera entre máquinas para un trabajo no está permitida.
- Almacenes de una máquina o *buffer* ($\beta = b_i$): Indican la cantidad de trabajos que pueden esperar en una máquina para ser procesados.

Para el caso en que no exista ninguna restricción se deberán realizar las siguientes suposiciones:

- Todos los trabajos están disponibles en el instante t=0.
- Las máquinas están disponibles en todo momento.
- El *buffer* entre las máquinas es infinito.
- El tiempo de transporte de los trabajos entre las máquinas es despreciable.
- Los trabajos no pueden sufrir interrupciones.
- Cada máquina puede procesar sólo un trabajo de forma simultánea.
- Cada trabajo únicamente puede ser procesado en una máquina.

2.4.3 Objetivo

Por último, el objetivo, definido por la letra , hace referencia a la función objetivo que se quiere optimizar. Algunos de los objetivos más comunes en los problemas de programación de la producción son los siguientes:

- *Makespan* o *Maximum completion time* ($C_{max} = max\ C_j$): El objetivo que se persigue es minimizar el mayor de los tiempos de terminación de los trabajos, es decir, del último.
- *Total completion* time ($\sum C_j$): Se busca minimizar la suma de los tiempos de terminación de todos los trabajos.
- *Maximum flowtime* ($max\ F_j$): Se busca minimizar el mayor valor del tiempo que un trabajo permanece en el entorno, desde que entra en la primera máquina hasta que termina de procesarse en la última.
- *Total flowtime* ($\sum F_j$): El objetivo es minimizar la suma de los tiempos que cada trabajo permanece en el entorno.
- *Maximum lateness* ($max\ L_j$): Minimizar el mayor de los retrasos de los trabajos.

- *Total lateness* ($\sum L_j$): Minimizar la suma de los retrasos de todos los trabajos. Cabe mencionar que si un trabajo acaba de procesarse antes de su fecha de entrega, entonces el valor asociado al retraso de dicho trabajo será negativo.
- *Maximum tardiness* ($max\ T_j$): Similar al objetivo *maximum lateness*, con la diferencia de que los valores negativos se sustituyen por cero.
- *Total tardiness* ($\sum T_j$): Al igual que en el *total lateness*, se busca minimizar la suma de los retrasos, con la diferencia de que en el *total tardiness* los valores negativos se sustituyen por cero.
- *Maximum earliness* ($max\ E_j$): Minimizar el mayor de los adelantos de los trabajos. Los adelantos con valores negativos se sustituyen por cero.
- *Total earliness* ($\sum E_j$): Minimizar la suma de los adelantos de todos los trabajos. Si un trabajo acaba de procesarse después de su fecha de entrega, entonces su adelanto será cero.
- *Number of tardy jobs* ($\sum U_j$): Se busca minimizar el número de trabajos terminados después de su fecha de entrega.

2.5 Caracterización y notación del problema

En base a todo lo expuesto durante el presente capítulo, el problema sobre el que versa este trabajo queda definido mediante la notación $\alpha|\beta|\gamma$ como:

$$Fm|prmu|C_{max}, \sum CIT_i$$

- Entorno ($\alpha = Fm$): se trata de un entorno de tipo taller de flujo uniforme o *flowshop* ($\alpha_1 = F$), compuesto por m máquinas ($\alpha_2 = m$), y n trabajos. Como se ha explicado en la sección **??**, el entorno *flowshop* se caracteriza porque todos los trabajos tienen la misma ruta.
- Restricciones ($\beta = prmu$): la restricción de permutación constituye un caso particular del entorno *flowshop*, donde la secuencia es la misma para todos los trabajos (S=(1,2,3,...n)).
- Objetivo ($\gamma = C_{max}, \sum CIT_i$): El objetivo del problema es minimizar el máximo de los tiempos de terminación de los trabajos (*makespan*) y el *core idle-time*. El *core idle-time* se define como el tiempo ocioso de una máquina entre dos trabajos. En la literatura se pueden encontrar hasta 3 definiciones diferentes para el *idle-time*, considerando o no los tiempos previos al primer trabajo de cada máquina (*front delay*) y los tiempos entre el último trabajo de cada máquina y el *completion time* del último trabajo (*back delay*) en cada caso. En este trabajo se tomará el concepto de *idle-time* como el conjunto entre el *front delay* y el *core idle-time*, sin tener en cuenta el *back delay*.

Figura 2.6 *Front delay*, *core idle-time* (IT) y *back delay*[1].

Por otra parte, la función objetivo se puede expresar como:

$$min\ f = w * C_{max} + (1 - w) * \sum_{i=1}^{m} CIT_i$$

Siendo w el peso o ponderación de C_{max}.

Tabla 2.1 Notación utilizada en el documento.

Notación	Descripción
m	Número de máqinas
n	Número de trabajos
i	Índice para las máquinas, $i \in (1, 2, ...m)$
j	Índice para los trabajos, $j \in (1, 2, \ldots n)$
p_{ij}	Tiempo de proceso del trabajo j en la máquina i
C_j	*Completion time* del trabajo j
C_{ij}	*Completion time* del trabajo j en la máquina i

Por último, se exponen una serie de hipótesis adicionalmente consideradas para este problema, las cuales son las siguientes:

- Todos los trabajos están disponibles en el instante $t = 0$.
- Las máquinas están siempre disponibles.
- Una máquina no puede procesar más de un trabajo de forma simultánea, así como un trabajo no puede ser procesado en dos o más máquinas diferentes al mismo tiempo.
- No hay fechas de entrega, es decir, $d_j = 0$. Tampoco habrá, por tanto, ningún tipo de penalización asociada a estas.
- Todos los tiempos de proceso son conocidos.
- Los tiempos de *setup* de los trabajos están considerados dentro de los tiempos de proceso.
- Los tiempos de transporte de los trabajos pueden considerarse despreciables.
- El *buffer* entre las máquinas se considera infinito.

3 Descripción de los algoritmos

En este capítulo se presentan los algoritmos utilizados para la obtención y el posterior análisis de los resultados del problema, detallando los pasos que sigue cada uno de ellos. En primer lugar se introducen los tres algoritmos implementados para la comparación de los resultados, muy reconocidos y utilizados para abordar problemas similares al tratado en este trabajo. Finalmente, se describirá el nuevo algoritmo propuesto.

3.1 NEH

Creado en 1982 por Muhammad Nawaz, E. Emory Enscore e Inyong Ham (Nawaz, Enscore, Ham, 1983), el NEH es la base de los mejores algoritmos que existen en la actualidad para resolver problemas de *permutation flowshop* con el objetivo de *makespan*.

El algoritmo NEH está basado en la idea de que aquellos trabajos que requieren mayores tiempos de proceso deben procesarse antes que los que tienen tiempos de proceso menores. Para ello, ordena los trabajos en función de la suma de sus tiempos de proceso y forma la secuencia mediante la inserción de los trabajos de uno en uno. Los pasos de los que se compone esta heurística son los siguientes:

Para empezar, es necesario definir una regla de prioridad o *priority rule* (PR) con la que establecer un orden inicial para los trabajos: en este caso la PR_{NEH} se basa en ordenar de mayor a menor la suma de los tiempos de proceso de cada trabajo:

- Primero: Se calcula la suma de los tiempos de proceso de los trabajos en cada una de las máquinas: $P_j = \sum_{i=1}^{m} p_{ij}$
- Segundo: Se ordenan los trabajos de mayor a menor en función de sus P_j, dando como resultado un vector con los trabajos ya ordenados.

Una vez se ha determinado el orden inicial, se procede a realizar la inserción de los trabajos en la secuencia parcial:

- Tercero: Se seleccionan los dos primeros trabajos y se determina la secuencia parcial óptima. Ésta será la de menor función objetivo, que en este caso coincidirá con la de menor *makespan*. Al final de este paso se fija la variable j con el valor j=3.
- Cuarto: Se selecciona el *j-ésimo* trabajo del vector obtenido en el segundo paso y se inserta en todas las posiciones posibles, sin alterar el orden de la subsecuencia anteriormente fijada. Se evalúan todas las posibles secuencias y se elige aquella de menor valor objetivo.
- Quinto: Repetir el cuarto paso con j=j+1 hasta que se hayan insertado todos los trabajos, es decir, hasta que j=n.

Input: instance data
Output: Π, C_{max}
begin
  $P_j = \sum_{i=1}^{m} p_{ij}$;
  $\Pi = \varnothing$, $J = \{1,\ldots,n\}$ verifying $P_1 \geq P_2 \geq \ldots P_n$;
  Let Π be the best permutation between $[1,2]$ and $[2,1]$;
  for $k = 3$ *to* n **do**
    Insert the job k in the position of Π yielding the lowest makespan value;
  return Π, C_{max}

Figura 3.1 Pseudocódigo del algoritmo NEH [4].

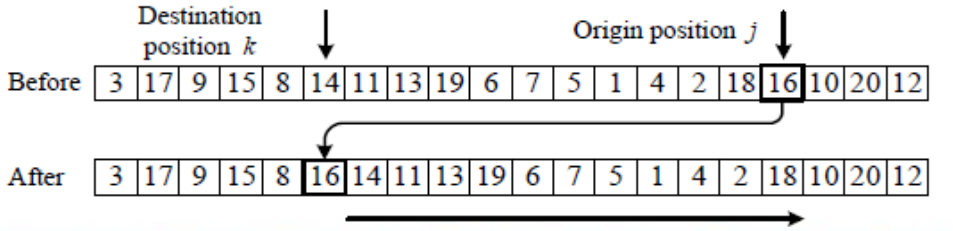


Figura 3.2 Ejemplo del método de inserción, en el que un trabajo es intercambiado de la posición j a la posición k de una secuencia [4].

3.2 NEH$_{FF}$

El algoritmo NEH$_{FF}$ es una variante del NEH desarrollada en 2014 por Víctor Fernández – Viagas y José M. Framiñán (Fernández – Viagas, Framiñán, 2014) que, mediante la implementación de la regla de desempate TB$_{FF}$, se ha consolidado como uno de los mejores algoritmos usados para resolver problemas del tipo $Fm|prmu|C_{max}$ hasta la fecha.

Las reglas de desempate, o *tie-breaking rules*, son herramientas de decisión que sirven para resolver los desempates que pueden darse a la hora de elegir una secuencia parcial óptima durante la fase de inserción. La TB$_{FF}$ está basada en la disminución de los *idle-time*, resolviendo los desempates a favor de la secuencia parcial cuya suma entre el *front delay* y el *core idle-time* sea la de menor valor. El cálculo de esta suma se realiza de la siguiente manera:

- Primero se calculan los *core idle-time* de cada máquina: $CIT_i = \sum_{j=2}^{n} max(C_{ij} - p_{ij} - C_{i,j-1}, 0)$
- En segundo lugar se calculan los *front delay* de cada máquina, a los que se les asignará la notación de FIT (*front idle-time*): $FIT_i = \sum_{k=1}^{i-1} p_{i-k,1}$, siendo $FIT_1 = 0$.
- A continuación, se suman los CIT y los FIT para obtener los *idle-time* de cada máquina: $IT_i = \sum_{i=1}^{m} FIT_i + CIT_i$
- Por último, se suman todos los IT_i ($\sum_{i=1}^{m} IT_i$), dando como resultado el *idle-time* total de la secuencia.

Una vez introducida la regla de desempate, se pasa a enumerar los pasos que ha de seguir el algoritmo:

En primer lugar, se ejecuta la *priority rule* del algoritmo NEH:

- Primero: Se calculan los P_j de todos los trabajos.
- Segundo: Se ordenan los trabajos de mayor a menor según sus P_j.

Una vez se tiene la secuencia inicial, es hora de pasar a la inserción. Para ello, se partirá de la secuencia parcial formada por los 2 primeros trabajos, en la que se irán insertando el resto de trabajos siguiendo la *tie-breaking rule* TB$_{FF}$:

- Tercero: Se seleccionan los dos primeros trabajos y se determina la secuencia parcial óptima.
- Cuarto: Para el resto de los trabajos (de $j = 3$ a $j = n$), se inserta el *j-ésimo* trabajo en todas las posiciones posibles y se retiene aquella secuencia cuya función objetivo sea la de menor valor. En caso de que haya más de una secuencia con el mismo valor de la función objetivo, se aplicará el método TB$_{FF}$, siendo seleccionada aquella secuencia cuyo IT asociado sea el de menor valor.
- Quinto: Repetir el cuarto paso hasta que todos los trabajos hayan sido insertados ($j = n$).

3.3 NEH$_{LJP}$

Creado en 2016 por Weibo Liu, Yan Jin y Mark Price (Weibo Liu, Yan Jin y Mark Price, 2016), el NEH$_{LJP}$ es uno de los algoritmos más recientes que, basándose en la heurística del NEH, ha sido diseñado para resolver el problema de *permutation flowshop* con los objetivos de *makespan* y *core idle-time* ($Fm|prmu|Cmax, \sum CIT_i$). Para ello, sus autores proponen una nueva regla de prioridad, basada en indicadores estadísticos, así como una regla de desempate.

3.3.1 PR$_{LJP}$

La *priority rule* que se propone en este algoritmo consiste en ordenar de mayor a menor los trabajos en función de:

$$\left(AVG_j + MAD_j + abs(SKE_j)^{\frac{1}{3}} + \frac{1}{KUR_j{}^{\frac{1}{4}}} \right)$$

A continuación se definen cada uno de los términos de la expresión anterior:

- AVG: El *average processing time* se corresponde con la media de los *processing time* del trabajo j en todas las máquinas: $AVG_j = \frac{1}{m} \sum_{i=1}^{m} p_{ij}$
- MAD: Este término hace referencia a la desviación absoluta media (*mean absolute deviation*) de los *processing time* del trabajo j: $MAD_j = \frac{1}{m} \sum_{i=1}^{m} |p_{ij} - AVG_j|$
  La desviación absoluta se utiliza como una medida de la dispersión, lo que permite conocer cómo se distribuyen los valores de una población con respecto a la media. En la figura 3.3 se puede observar cómo dos poblaciones, a pesar de compartir la misma media, presentan distribuciones muy diferentes. Por ello, estudiar la dispersión resulta de gran interés a la hora de trabajar con conjuntos de datos.

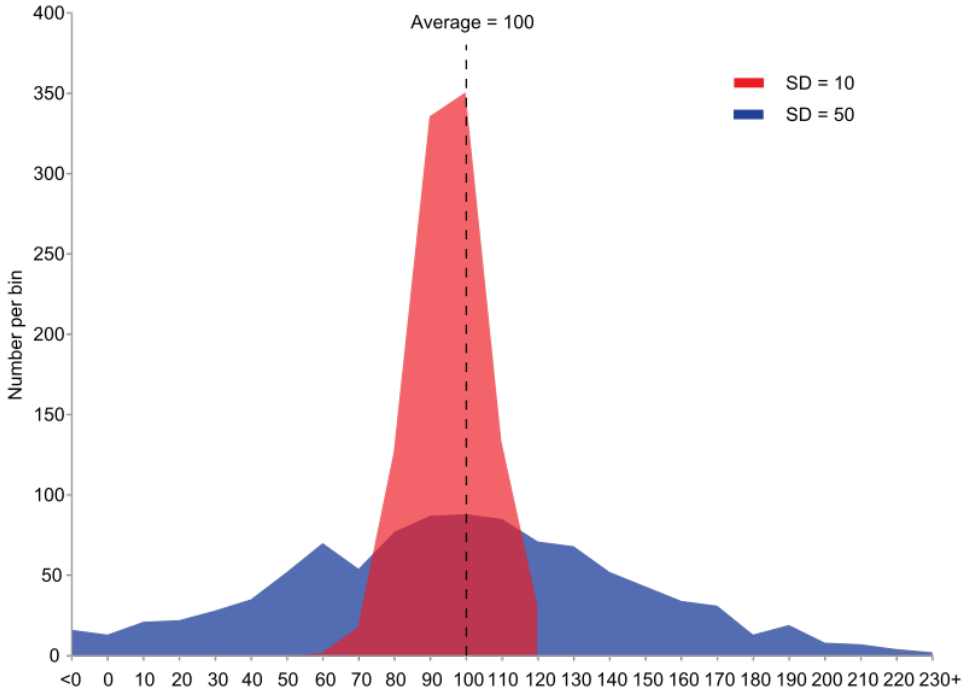


Figura 3.3 Diferencia entre dos poblaciones de igual media y diferente dispersión.

- SKE: La asimetría, o *skewness*, es un indicador estadístico que sirve para conocer la tendencia de una población a desviarse de la media. Se define de la siguiente manera:

$$SKE_j = \frac{\frac{1}{m}\sum_{i=1}^{m}(p_{ij} - AVG_j)^3}{\left(\sqrt{\frac{1}{m}\sum_{i=1}^{m}(p_{ij} - AVG_j)^2}\right)^3}$$

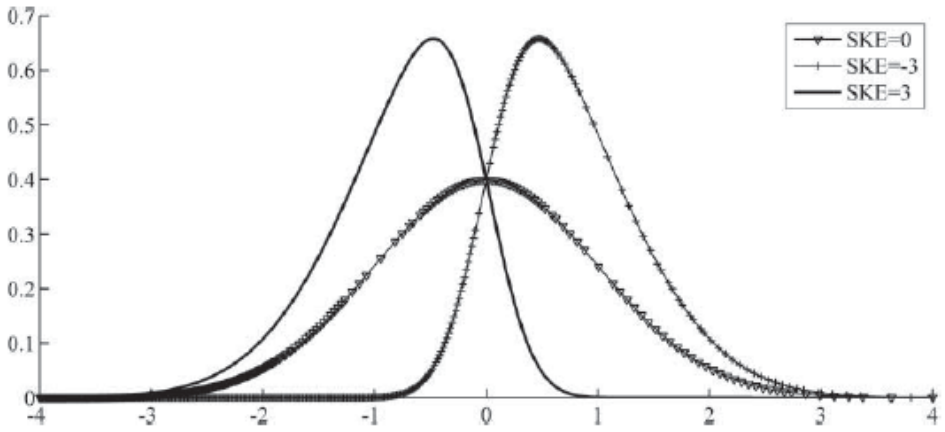


Figura 3.4 Efecto del *skewness* en una función [1].

- KUR: La curtosis (en inglés *kurtosis*) indica la concentración de los valores de una población con respecto a la media. Su expresión matemática se puede escribir como:

$$KUR_j = \frac{\frac{1}{m}\sum_{i=1}^{m}(p_{ij} - AVG_j)^4}{\left(\sqrt{\frac{1}{m}\sum_{i=1}^{m}(p_{ij} - AVG_j)^2}\right)^2}$$

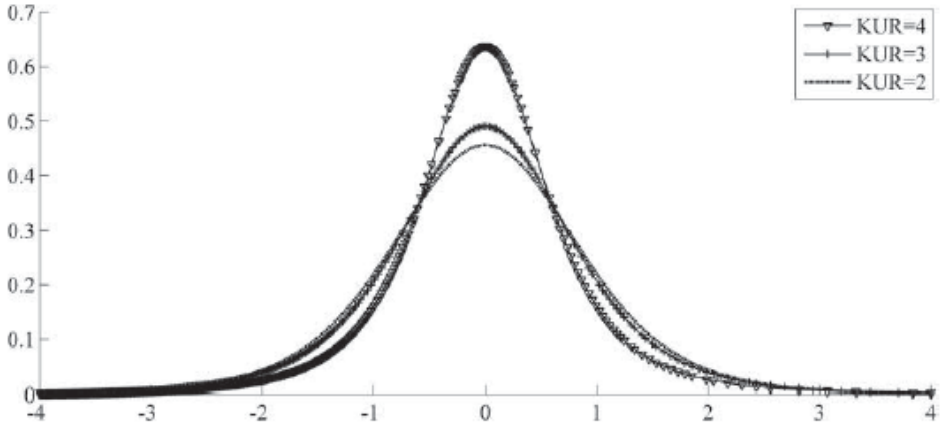


Figura 3.5 Efecto de la curtosis en una función [1].

3.3.2 TB$_{LJP}$

La regla de desempate del algoritmo NEH$_{LJP}$ está basada en los tiempos de flujo dinámicos o *dynamic flowtimes*, partiendo de la idea de que "minimizar los tiempos de flujo conduce a una disminución del *makespan*, así como a un aumento de los tiempos ociosos de las máquinas, y viceversa" [11].

Se entiende por *dynamic flowtime* la cantidad de tiempo transcurrido entre el instante en que un trabajo comienza a procesarse en la primera máquina y el instante en que termina de procesarse en la última (no confundir este concepto con el *flowtime*, explicado en la sección 4.2. Se expresa de la siguiente forma: $\bar{f}_j = C_{mj} - C_{1,j-1}$, siendo C_{ij} el instante de tiempo en que el trabajo j termina de procesarse en la máquina i.

Atendiendo a lo explicado anteriormente, la TB$_{LJP}$ consiste en elegir aquella secuencia que proporcione el mayor valor para la expresión dada por:

$$\frac{\bar{f} - \sqrt{\frac{1}{n-1}\sum_{j}^{n}\left(f_j - \bar{f}\right)^2}}{C_{max}}$$

donde $\bar{f} = \frac{1}{n}\sum_{j=1}^{n} f_j$ representa la media de los *dynamic flowtime* de cada trabajo.

3.3.3 Algoritmo NEH$_{LJP}$

Atendiendo a todo lo expuesto anteriormente, el algoritmo NEH$_{LJP}$ se ejecuta siguiendo los siguientes pasos:

- Primero: Se calcula el valor de $\left(AVG_j + MAD_j + abs(SKE_j)^{\frac{1}{3}} + \frac{1}{KUR_j{}^{\frac{1}{4}}}\right)$ para cada uno.

- Segundo: Se ordenan los trabajos de mayor a menor en función de los valores calculados en el paso primero.
- Tercero: Se seleccionan los dos primeros trabajos y se determina la secuencia parcial óptima.
- Cuarto: Para el resto de los trabajos (de $j = 3$ a $j = n$), se inserta el *j-ésimo* trabajo en todas las posiciones posibles y se retiene aquella secuencia cuya función objetivo sea la de menor valor. En caso de que haya más de una secuencia con el mismo valor de la función objetivo, se seleccionará aquella que proporcione el mayor valor de $\left(\frac{\bar{f}-\sqrt{\frac{1}{n-1}\sum_j^n\left(f_j-\bar{f}\right)^2}}{C_{max}}\right)$
- Quinto: Repetir el cuarto paso hasta que todos los trabajos hayan sido insertados ($j = n$).

3.4 Algoritmo propuesto: NEH$_C$

El algoritmo que se propone en este trabajo consiste en una variante del algoritmo NEH original, diseñada para disminuir los *core idle-time* de las máquinas y así cumplir con los dos objetivos del problema que se plantea ($min\ w * C_{max} + (1-w) * \sum_{i=1}^{m} CIT_i$).

Para ello, se introduce una regla de prioridad basada en la PR del NEH, así como una nueva regla de desempate, fundamentada en el concepto de la utilización de las máquinas, que será explicado en la sección 4.2.

3.4.1 PR$_C$

La *priority rule*, designada como PR$_C$, consiste en ordenar los trabajos de mayor a menor en función de:

$$\left(\sum_{i=1}^{m} p_{ij} + \sum_{i=1}^{m} \left|p_{ij} - \bar{p}\right| + R_j\right)$$

donde R_j representa el rango de los *processing times* del trabajo *j*, es decir, la diferencia entre el valor máximo y el mínimo de los p_{ij}, que se calcula como: $R_j = max(p_{ij}) - min(p_{ij}), i \in (1, 2, ...m)$.

Como se ha explicado en la sección 3.3.1, la desviación absoluta, correspondiente al segundo término de la expresión de la PR$_C$, sirve para cuantificar la dispersión de los *processing times* de cada trabajo. Conocer información relativa a la dispersión de los valores de una población permite asignar una mayor prioridad a aquellas tareas que presenten una mayor variabilidad en sus tiempos de proceso, y así atenuar los efectos negativos que pueda tener sobre la solución final.
Para complementar la información proporcionada por la desviación absoluta se incluye también el rango, que permite acotar los intervalos entre los que se mueven los tiempos de proceso. Un rango menor implica valores de dispersión menores, y por consiguiente, tiempos de proceso más uniformes. De este modo, aquellos trabajos cuyo rango sea mayor deberán ser procesados antes que el resto.

3.4.2 TB$_C$

La regla de desempate que se propone, como se ha mencionado anteriormente, está basada en el concepto de utilización, que hace referencia al grado en que una máquina es aprovechada, o dicho de otra forma, qué porcentaje del tiempo se está procesando un trabajo con respecto al tiempo total en el que una máquina permanece en funcionamiento.

La utilización de una máquina i se puede definir de la siguiente manera:

$$U_i = \frac{\sum_{j=1}^{n} p_{ij}}{\sum_{j=1}^{n} p_{ij} + CIT_i}$$

Para un problema de m máquinas se utiliza la media de los factores de utilización:

$$U = \frac{1}{m}\sum_{i=1}^{m}\frac{\sum_{j=1}^{n} p_{ij}}{\sum_{j=1}^{n} p_{ij} + CIT_i}$$

Es fácil comprobar que, a medida que la utilización de una máquina aumenta, su *core idle-time* disminuye. Por lo tanto, la *tie-breaking rule* del algoritmo NEH$_C$ resolverá los desempates a favor de la secuencia que tenga una mayor utilización media.

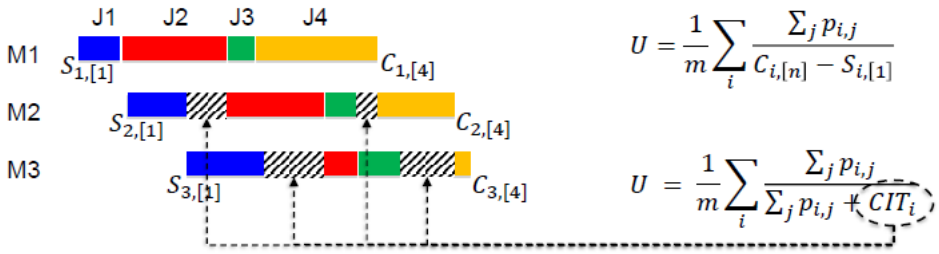


Figura 3.6 Utilización media de un conjunto de máquinas [11].

3.4.3 Algoritmo NEH$_C$

Basado en las reglas de prioridad y desempate descritas en las secciones 3.4.1 y 3.4.2 se propone el algoritmo NEH$_C$, cuya ejecución se lleva a cabo siguiendo los pasos enumerados a continuación:

- Primero: Se calcula el valor de $\left(\sum_{i=1}^{m} p_{ij} + \sum_{i=1}^{m} \left|p_{ij} - \bar{p}\right| + R_j\right)$ para cada uno de los trabajos.
- Segundo: Se ordenan los trabajos de mayor a menor en función de los valores calculados en el paso primero.
- Tercero: Se seleccionan los dos primeros trabajos y se determina la secuencia parcial óptima
- Cuarto: Para el resto de los trabajos (de $j = 3$ a $j = n$), se inserta el *j-ésimo* trabajo en todas las posiciones posibles y se retiene aquella secuencia cuya función objetivo sea la de menor valor. En caso de que haya más de una secuencia con el mismo valor de la función objetivo, se resolverá el empate a favor de la secuencia con el mayor valor de $U = \frac{1}{m}\sum_{i=1}^{m}\frac{\sum_{j=1}^{n} p_{ij}}{\sum_{j=1}^{n} p_{ij} + CIT_i}$
- Quinto: Repetir el cuarto paso hasta que todos los trabajos hayan sido insertados ($j = n$).

4 Desarrollo de los métodos

En este capítulo se expone el *software* utilizado para la implementación de los algoritmos descritos en el capítulo 3, la resolución de los problemas y el posterior análisis de los resultados obtenidos. También se presentan los problemas elegidos para efectuar las pruebas anteriormente mencionadas, así como los distintos parámetros utilizados para ello. Finalmente, se describen los indicadores con los que se realizará la comparación de los algoritmos.

4.1 Programación de los algoritmos

El programa elegido para la resolución de los problemas, *Matlab*, es un *software* lanzado por la compañía *MathWorks* en 1984. *Matlab* posee su propio lenguaje de programación, llamado "lenguaje M", en el cual se han implementado los algoritmos. Una de las principales ventajas con respecto a otros entornos de programación es la facilidad para la creación y manejo de funciones, lo cual ha resultado de especial utilidad debido al carácter iterativo de los algoritmos. La posibilidad de examinar la información almacenada en las variables directamente desde el *workspace* es otra de las virtudes de este *software*, ya que al disponer los valores en celdas permite que su exportación a Excel se pueda llevar a cabo de forma rápida y sencilla sin necesidad de cambiar su formato.

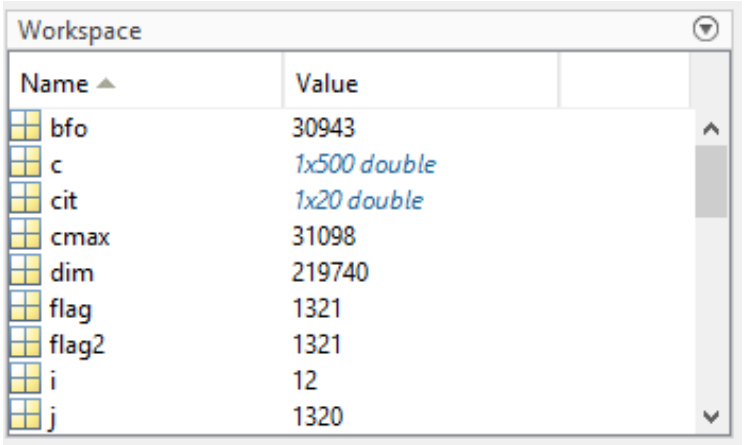
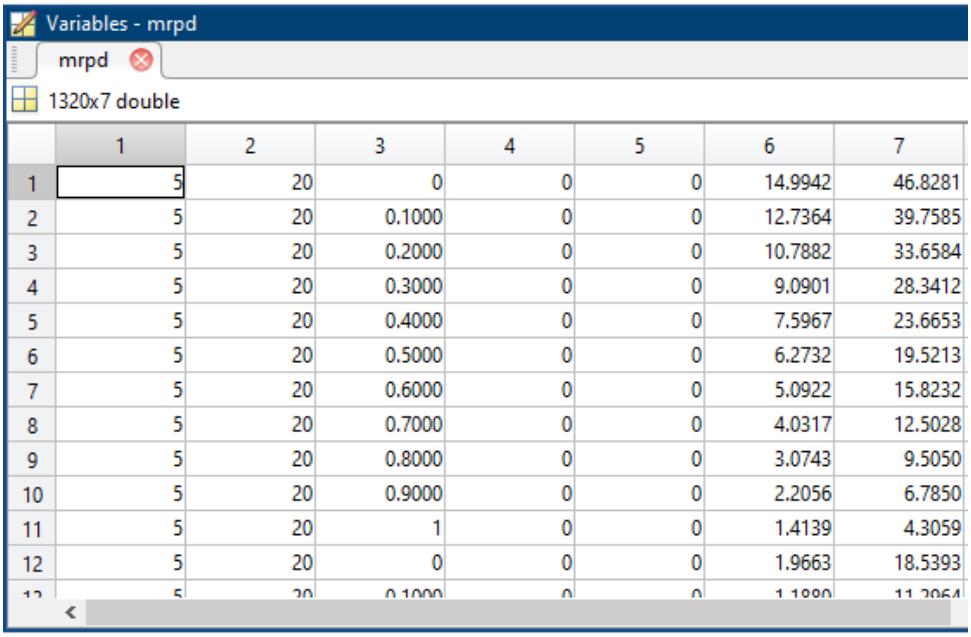


Figura 4.1 *Workspace* de *Matlab*.



Variables - mrpd

mrpd

1320x7 double

	1	2	3	4	5	6	7
1	5	20	0	0	0	14.9942	46.8281
2	5	20	0.1000	0	0	12.7364	39.7585
3	5	20	0.2000	0	0	10.7882	33.6584
4	5	20	0.3000	0	0	9.0901	28.3412
5	5	20	0.4000	0	0	7.5967	23.6653
6	5	20	0.5000	0	0	6.2732	19.5213
7	5	20	0.6000	0	0	5.0922	15.8232
8	5	20	0.7000	0	0	4.0317	12.5028
9	5	20	0.8000	0	0	3.0743	9.5050
10	5	20	0.9000	0	0	2.2056	6.7850
11	5	20	1	0	0	1.4139	4.3059
12	5	20	0	0	0	1.9663	18.5393

Figura 4.2 Formato de los datos almacenados en una matriz.

Por otra parte, el programa que se ha elegido para el análisis y comparación de los resultados es *Microsoft Excel*, debido a la amplia variedad que presenta en cuanto a tablas y gráficos con los que poder representar y comparar información.

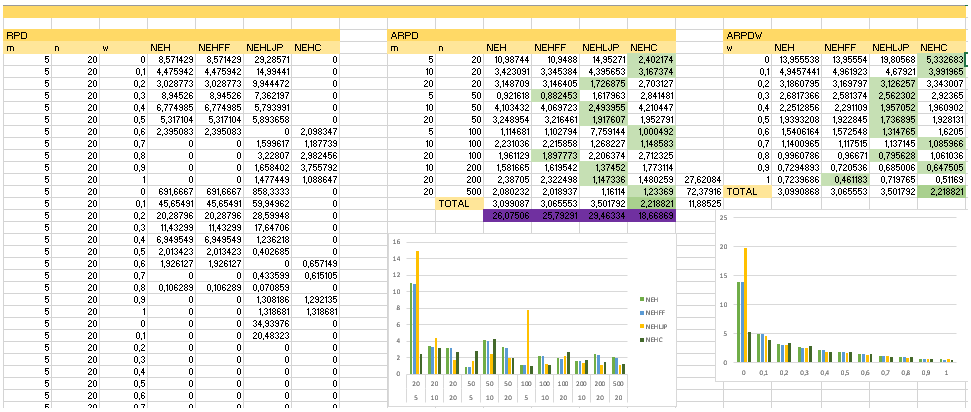


Figura 4.3 Análisis de soluciones en *Microsoft Excel*.

Finalmente, se deben considerar las siguientes condiciones experimentales:

- Se utilizará el mismo ordenador durante todo el experimento. En este caso se trata de un equipo personal con procesador INTEL CORE i7-6500U CPU, 2.50GHz y 12GB de memoria RAM.
- Se utilizará el mismo leguaje (lenguaje M), programa (*Matlab R2020b*, *Microsoft Excel*) y sistema operativo (*Microsoft Windows 10 Home*). Además, se hará uso de las mismas librerías y funciones.
- Se utilizarán las mismas instancias para las comparaciones. De esto se hablará con más detalle en la siguiente sección.

4.2 Generación de las instancias

Para comparar la eficacia de los algoritmos se han escogido las instancias de Taillard (Éric Taillard, 1993), en las que se proponen 120 instancias, divididas en 12 grupos de 10 problemas cada uno, cuyos tiempos de proceso han sido generados a partir de una distribución uniforme, tomando valores de entre 1 y 99. Dichos grupos han sido divididos en función del número de máquinas y trabajos que los componen, expuestos en la tabla 4.1. Dado que el valor de la función objetivo depende de la ponderación que se le asigne a cada objetivo, se estudiará cada problema variando el parámetro w entre 0 y 1 en intervalos de 0.1, lo que originará un total de 1320 problemas.

Tabla 4.1 Grupos de problemas según su tamaño.

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

```
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5   873654221        1278        1232
processing times :
 54 83 15 71 77 36 53 38 27 87 76 91 14 29 12 77 32 87 68 94
 79  3 11 99 56 70 99 60  5 56  3 61 73 75 47 14 21 86  5 77
 16 89 49 15 89 45 60 23 57 64  7  1 63 41 63 47 26 75 77 40
 66 58 31 68 78 91 13 59 49 85 85  9 39 41 56 40 54 77 51 31
 58 56 20 85 53 35 53 41 69 13 86 72  8 49 47 87 58 18 68 28
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5   379008056        1359        1290
processing times :
 26 38 27 88 95 55 54 63 23 45 86 43 43 40 37 54 35 59 43 50
 59 62 44 10 23 64 47 68 54  9 30 31 92  7 14 95 76 82 91 37
 78 90 64 49 47 20 61 93 36 47 70 54 87 13 40 34 55 13 11  5
 88 54 47 83 84  9 30 11 92 63 62 75 48 23 85 23  4 31 13 98
 69 30 61 35 53 98 94 33 77 31 54 71 78  9 79 51 76 56 80 72
number of jobs, number of machines, initial seed, upper bound and lower bound :
          20           5  1866992158        1081        1073
processing times :
 77 94  9 57 29 79 55 73 65 86 25 39 76 24 38  5 91 29 22 27
 39 31 46 18 93 58 85 58 97 10 79 93  2 87 17 18 10 50  8 26
 14 21 15 10 85 46 42 18 36  2 44 89  6  3  1 43 81 57 76 59
 11  2 36 30 89 10 88 22 31  9 43 91 26  3 75 99 63 83 70 84
 83 13 84 46 20 33 74 42 33 71 32 48 42 99  7 54  8 73 30 75
```

Figura 4.4 Instancias de Taillard: disposición de los problemas [Fuente: http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html].

4.3 Indicadores

A la hora de evaluar el desempeño de los algoritmos se considerarán tanto la calidad de las soluciones obtenidas como el tiempo de ejecución de cada uno de ellos. Dichos aspectos serán cuantificados mediante el ARPD y el tiempo de CPU:

- El *average relative percentage deviation*, o ARPD por sus siglas en inglés, mide la desviación relativa media de las soluciones obtenidas por un algoritmo en función de la mejor solución encontrada. Para ello, es necesario haber calculado previamente los RPD (*relative percentage deviation*) de cada una de las soluciones:

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

  donde OS se utiliza para denotar la solución que se quiere evaluar (*obtained solution*), y BS para denotar la mejor solución hallada (*best solution*), de modo que si una solución es la mejor de todo el conjunto su RPD valdrá 0.
  Una vez se han obtenido todos los RPD se pasa a calcular el ARPD como la media entre las k soluciones:

  $$ARPD = \frac{1}{k} \sum_{i=1}^{k} \mathrm{RPD}_i$$

- El tiempo de la CPU mide en segundos la cantidad de tiempo transcurrido durante la ejecución del algoritmo. Para poder llevar a cabo una evaluación más exhaustiva se han medido los tiempos de ejecución para cada uno de los problemas por separado.

5 Resultados y análisis

En este capítulo se exponen los resultados obtenidos por los 4 métodos (NEH, NEH_{FF}, NEH_{LJP}, NEH_C) tras la resolución de los problemas descritos en la sección 4.2 del capítulo anterior. Además, se evaluará la eficacia de las reglas de prioridad y desempate del método NEH_C (PR_C y TB_C), adaptándolas por separado a los algoritmos NEH, NEH_{FF} y NEH_{LJP}. Finalmente se analizarán los resultados obtenidos, con el objetivo de comprobar en qué aspectos destaca el algoritmo propuesto en el presente trabajo frente al resto.

En cada una de las secciones que conforman el capítulo se presentan los resultados obtenidos, correspondientes a los distintos experimentos. Dichos resultados han sido estructurados en 3 tablas diferentes:

- La primera tabla de cada sección muestra los ARPD de cada conjunto de problemas según su dimensión, es decir, la media de los ARPD de todos los problemas que comparten el mismo número de máquinas (m) y trabajos (n).
- La segunda tabla muestra la media de los ARPD según la ponderación del problema (w).
- Por último, en la tercera tabla se muestran los tiempos de CPU de cada conjunto de problemas según su dimensión.

En el anexo "ARPD en función de las características del problema" [6] aparecen las tablas correspondientes a los ARPD de cada conjunto de problemas según su dimensión (m, n) y ponderación (w).

5.1 PR

En primer lugar se han implementado únicamente las *priority rules* de cada método, con el fin de comparar su eficiencia a la hora de obtener una secuencia inicial. Los resultados obtenidos han sido los siguientes:

Tabla 5.1 ARPD de cada *priority rule* en función de la dimensión de los problemas.

m	n	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
5	20	3,40	3,40	9,30	15,89
10	20	5,34	5,34	8,90	12,22
20	20	2,51	2,51	3,90	5,27
5	50	5,10	5,10	8,04	13,51
10	50	3,33	3,33	1,40	4,12
20	50	2,55	2,55	4,63	3,87
5	100	0,62	0,62	3,98	4,75
10	100	2,46	2,46	6,51	7,18
20	100	0,62	0,62	7,45	6,05
10	200	4,96	4,96	2,83	3,19
20	200	0,29	0,29	4,72	5,17
20	500	3,02	3,02	2,46	2,13

Tabla 5.2 ARPD de cada *priority rule* en función de los pesos de los problemas.

w	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
0,0	5,232	5,232	11,334	15,826
0,1	4,423	4,423	9,019	12,230
0,2	3,849	3,849	7,657	10,150
0,3	3,387	3,387	6,609	8,591
0,4	2,991	2,991	5,720	7,302
0,5	2,633	2,633	4,917	6,169
0,6	2,301	2,301	4,158	5,127
0,7	1,996	1,996	3,426	4,147
0,8	1,698	1,698	2,680	3,180
0,9	1,465	1,465	1,956	2,255
1,0	1,439	1,439	1,352	1,468

Tabla 5.3 Tiempos de ejecución de cada *priority rule*.

m	n	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
5	20	0,0000	0,0016	0,0000	0,0001
10	20	0,0000	0,0000	0,0016	0,0000
20	20	0,0000	0,0000	0,0016	0,0000
5	50	0,0003	0,0016	0,0000	0,0000
10	50	0,0000	0,0000	0,0031	0,0000
20	50	0,0000	0,0000	0,0016	0,0000
5	100	0,0109	0,0016	0,0078	0,0016
10	100	0,0047	0,0016	0,0063	0,0016
20	100	0,0063	0,0000	0,0078	0,0016
10	200	0,0172	0,0125	0,0344	0,0109
20	200	0,0094	0,0047	0,0344	0,0078
20	500	0,0578	0,0203	0,1625	0,0297
Promedio		0,0089	0,0036	0,0217	0,0044

El análisis de los datos obtenidos al implementar la *priority rule* de los distintos métodos muestra que la TB_C es capaz de conseguir una mejor secuencia inicial en el 20.98 % de los casos, equivalente a 277 de los 1320 problemas propuestos, destacando por encima del resto únicamente en los problemas de mayor dimensión, compuestos por 20 máquinas y 500 trabajos, como se puede ver en la figura 5.1. El mejor resultado lo logra la *priority rule* del método NEH, que ha sido capaz de obtener la mejor secuencia inicial para el 54.17 % de los casos (715 problemas):

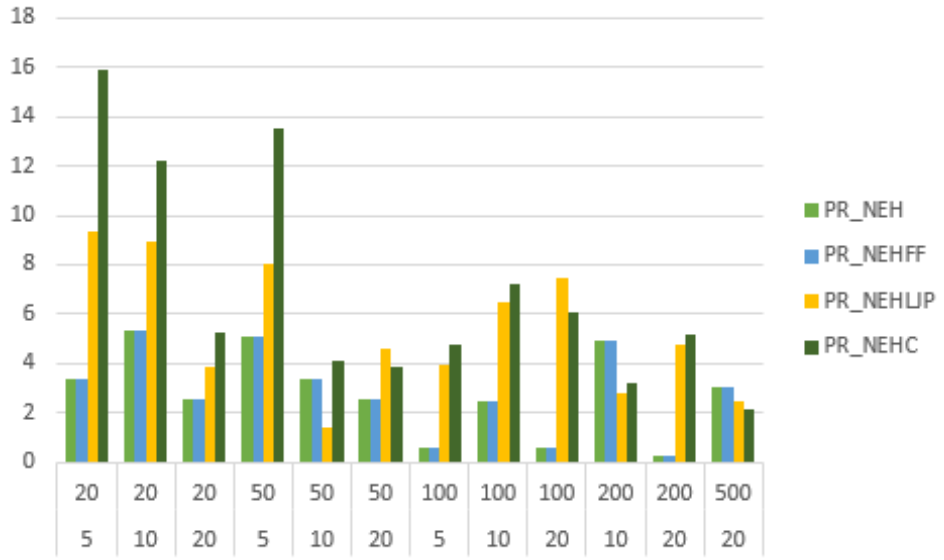


Figura 5.1 ARPD de cada *priority rule* en función de la dimensión de los problemas.

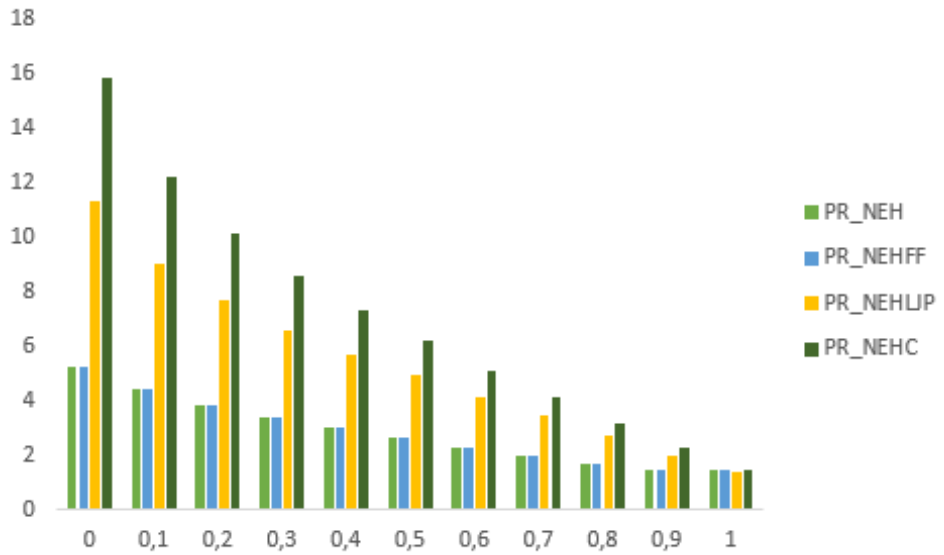


Figura 5.2 ARPD de cada *priority rule* en función de los pesos de los problemas.

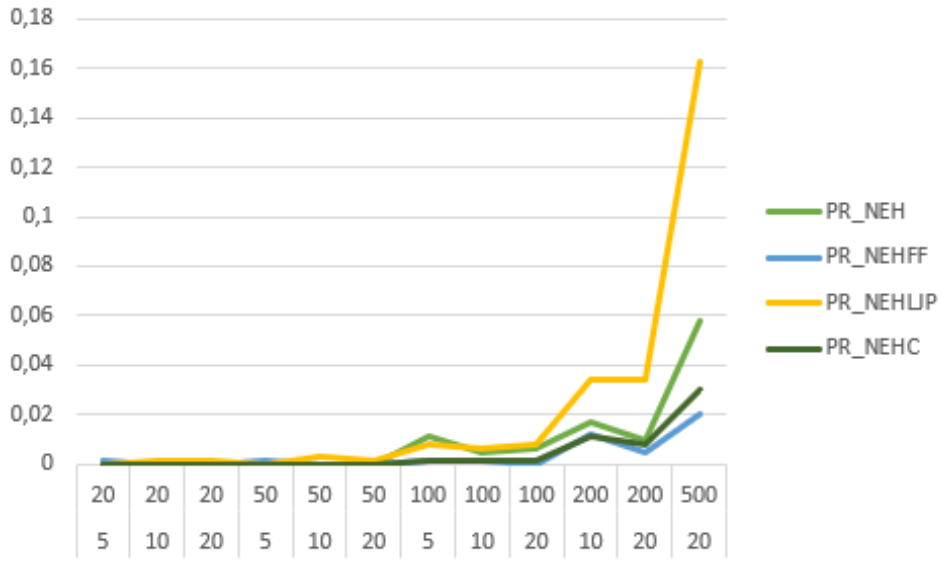


Figura 5.3 Tiempos de ejecución de cada *priority rule*.

5.2 PR+TB

A continuación, se muestran las soluciones obtenidas al implementar tanto la PR como la TB de cada uno de los métodos:

Tabla 5.4 ARPD de cada método en función de la dimensión de los problemas.

m	n	NEH	NEH_{FF}	NEH_{LJP}	NEH_{C}
5	20	10,99	10,95	14,95	2,40
10	20	3,42	3,35	4,40	3,17
20	20	3,15	3,15	1,73	2,70
5	50	0,92	0,88	1,62	2,84
10	50	4,10	4,07	2,49	4,21
20	50	3,25	3,22	1,92	1,95
5	100	1,11	1,10	7,76	1,00
10	100	2,23	2,22	1,27	1,15
20	100	1,96	1,90	2,21	2,71
10	200	1,58	1,62	1,37	1,77
20	200	2,39	2,32	1,15	1,48
20	500	2,08	2,02	1,16	1,23

Tabla 5.5 ARPD de cada método en función de los pesos de los problemas.

w	NEH	NEH_{FF}	NEH_{LJP}	NEH_C
0	13,96	13,96	19,81	5,33
0,1	4,95	4,96	4,68	3,99
0,2	3,19	3,17	3,13	3,34
0,3	2,68	2,58	2,56	2,92
0,4	2,25	2,29	1,96	1,96
0,5	1,94	1,92	1,74	1,93
0,6	1,54	1,57	1,31	1,62
0,7	1,14	1,12	1,14	1,09
0,8	1,00	0,97	0,80	1,06
0,9	0,73	0,72	0,69	0,65
1	0,72	0,46	0,72	0,51

Tabla 5.6 Tiempos de ejecución de cada método.

m	n	NEH	NEH_{FF}	NEH_{LJP}	NEH_C
5	20	0,0203	0,0104	0,0250	0,0118
10	20	0,0136	0,0044	0,0111	0,0053
20	20	0,0041	0,0053	0,0063	0,0058
5	50	0,0533	0,0591	0,0659	0,0624
10	50	0,0544	0,0645	0,0642	0,0646
20	50	0,0578	0,0709	0,0746	0,0764
5	100	0,4295	0,4575	0,4533	0,4702
10	100	0,4314	0,4814	0,4878	0,5091
20	100	0,4585	0,5456	0,5598	0,6043
10	200	3,7842	4,0947	4,1229	4,2581
20	200	4,0314	4,6024	4,6344	4,9423
20	500	76,1034	83,6560	84,0778	90,4483
Promedio		7,1202	7,8377	7,8819	8,4549

Al implementar las reglas de desempate en sus respectivos algoritmos, se puede observar cómo para el NEH_C se obtiene el mejor ARPD promedio de los 4 métodos comparados, con una diferencia del 27.62 % respecto al segundo mejor valor. Habiendo resuelto un total de 428 problemas (32.42 %) con la mejor solución de su conjunto, se destaca la eficacia del algoritmo NEH$-C$ en los problemas de dimensión 5x20, 10x20, 5x100, 10x100 y 20x500 (Figura 5.5), así como para las ponderaciones de valor 0, 0.1, 0.7 y 0.9 (Figura 5.6). Cabe mencionar que para el caso $w = 0$, es decir, para aquellos problemas donde el objetivo es disminuir únicamente el CIT, se obtienen los mejores resultados con una diferencia superior al 60 % respecto a los otros métodos, posicionándose como el mejor método tanto en términos de calidad en sus soluciones como en tiempo de ejecución.

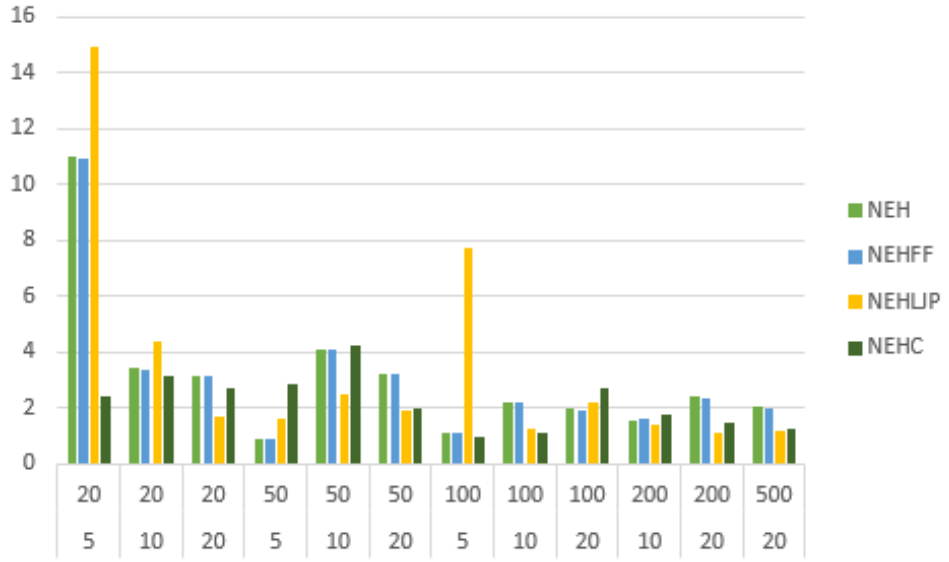


Figura 5.4 ARPD de cada método en función de la dimensión de los problemas.

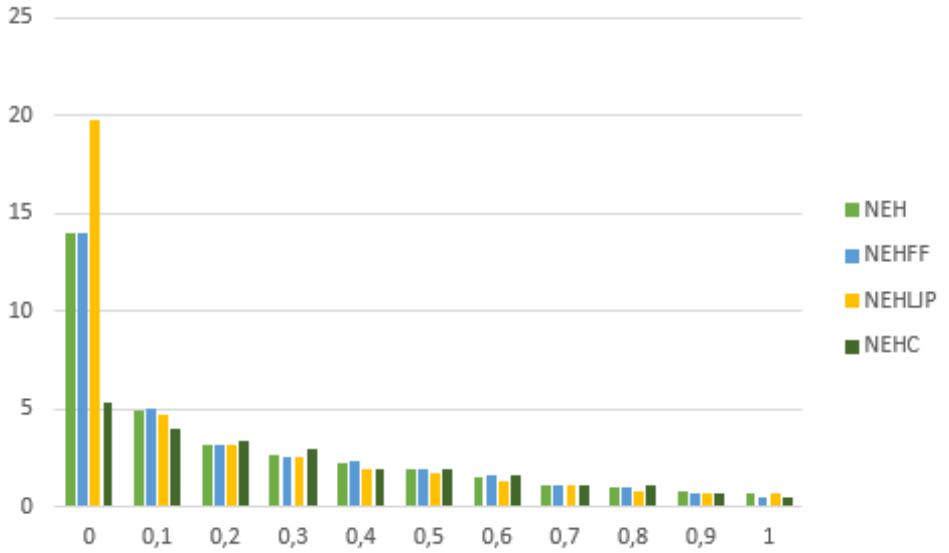


Figura 5.5 ARPD de cada método en función de los pesos de los problemas.

Con respecto a los tiempos de ejecución, el NEH se establece como el algoritmo de menor tiempo de ejecución medio con un total de 7.12 segundos, mientras que el tiempo medio de ejecución del algoritmo NEH_C es de 8.45 segundos. La diferencia de tiempo (1.33 segundos) se justifica debido a la complejidad de ambas heurísticas y al número de operaciones implícitas en cada una de ellas, pudiéndose apreciar que la diferencia de los tiempos de ejecución de ambos métodos tiende a aumentar más cuanto mayor es la dimensión de los problemas. No obstante, dicha diferencia es razonable y no influye sobremanera en la elección de un método u otro.

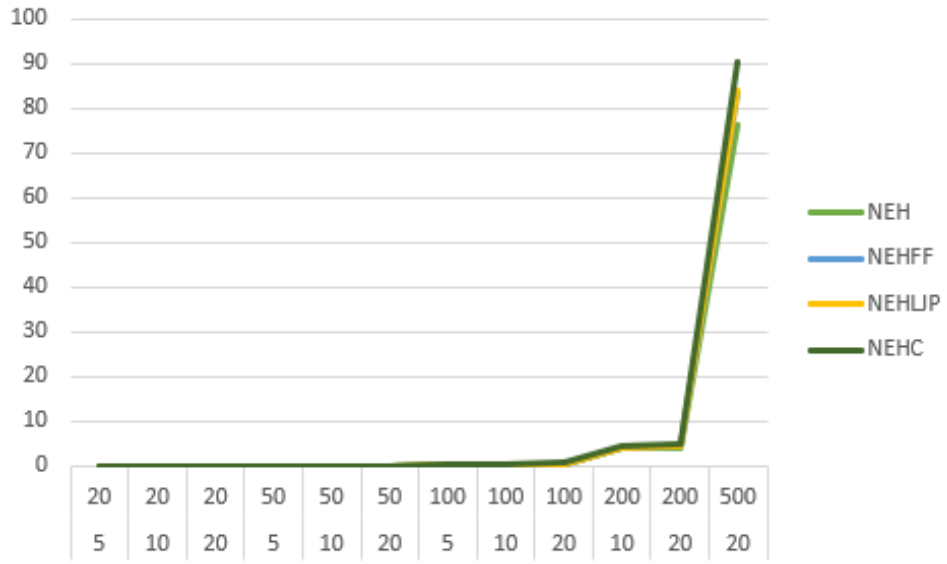


Figura 5.6 Tiempos de ejecución de cada método.

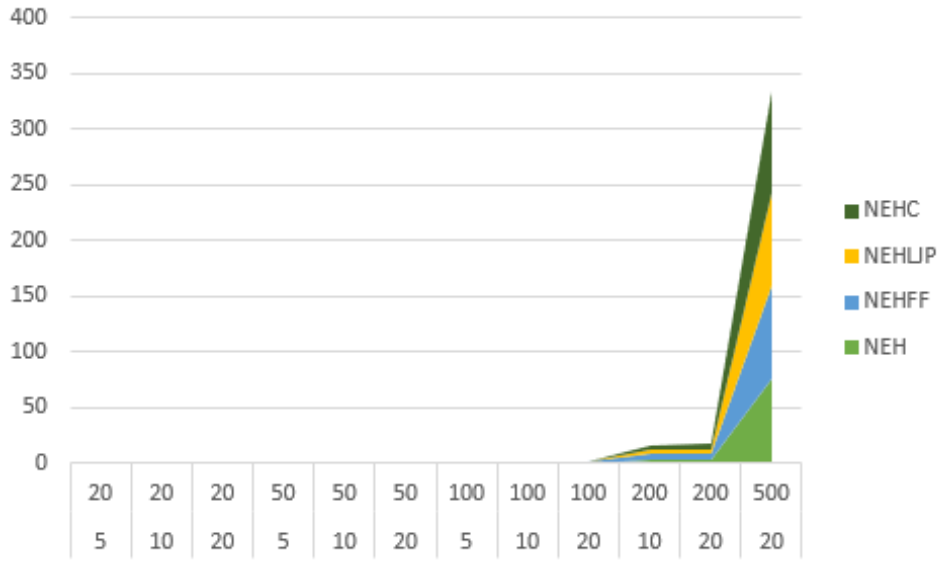


Figura 5.7 Tiempos de ejecución acumulados de cada método.

5.3 PR$_C$ + TB

Después de analizar los resultados obtenidos mediante los métodos NEH, NEH$_{FF}$, NEH$_{LJP}$ y NEH$_C$, se pasa a estudiar la eficacia de la PR$_C$ y la TB$_C$ por separado. En este primer ensayo se ha implementado la PR$_C$ en todos los métodos, con el fin de averiguar si ésta es capaz de obtener mejores resultados en combinación con la *tie breaking rule* de los demás métodos que las originales:

Tabla 5.7 ARPD de cada *tie breaking rule* implementada con la PR_C en función de la dimensión de los problemas.

m	n	$PR_C + TB_{NEH}$	$PR_C + TB_{FF}$	$PR_C + TB_{LJP}$	$PR_C + TB_C$
5	20	1,06	1,30	0,54	1,08
10	20	0,37	0,32	0,33	0,26
20	20	0,16	0,13	0,22	0,15
5	50	0,25	0,23	0,55	0,65
10	50	1,44	1,36	1,21	1,19
20	50	0,42	0,38	0,27	0,40
5	100	0,18	0,17	0,49	0,21
10	100	1,00	0,90	0,58	0,74
20	100	1,44	1,42	1,15	1,34
10	200	0,81	0,86	0,94	0,84
20	200	1,49	1,48	0,87	1,30
20	500	1,32	1,23	1,05	1,26

Tabla 5.8 ARPD de cada *tie breaking rule* implementada con la PR_C en función de los pesos de los problemas.

w	$PR_C + TB_{NEH}$	$PR_C + TB_{FF}$	$PR_C + TB_{LJP}$	$PR_C + TB_C$
0	2,26	2,51	1,97	2,15
0,1	1,53	1,49	1,05	1,38
0,2	1,19	1,14	0,78	1,14
0,3	0,84	0,84	0,58	0,91
0,4	0,76	0,76	0,50	0,77
0,5	0,71	0,62	0,60	0,55
0,6	0,50	0,53	0,59	0,48
0,7	0,35	0,37	0,37	0,38
0,8	0,31	0,33	0,31	0,33
0,9	0,19	0,19	0,26	0,18
1	0,46	0,18	0,50	0,34

Tabla 5.9 Tiempos de ejecución de cada *tie breaking rule* implementada con la PR_C.

m	n	$PR_C + TB_{NEH}$	$PR_C + TB_{FF}$	$PR_C + TB_{LJP}$	$PR_C + TB_C$
5	20	0,0115	0,0094	0,0087	0,0098
10	20	0,0041	0,0061	0,0060	0,0061
20	20	0,0060	0,0065	0,0068	0,0074
5	50	0,0601	0,0729	0,0717	0,0732
10	50	0,0570	0,0768	0,0838	0,0791
20	50	0,0719	0,0847	0,0913	0,0942
5	100	0,4574	0,5463	0,5658	0,5974
10	100	0,5517	0,6023	0,5939	0,6111
20	100	0,7720	0,6751	0,6964	0,7483
10	200	6,6557	5,0391	5,1135	5,2466
20	200	7,7572	5,6445	5,6540	6,0368
20	500	108,9591	102,4835	103,2376	109,2045
Promedio		10,4470	9,6039	9,6775	10,2262

Los resultados tras implementar la PR_C en todos los algoritmos junto con sus reglas de desempate originales muestran que la *tie breaking rule* que mejor desempeño logra junto con esta regla de prioridad es la TB_{LJP}, con un ARPD promedio inferior a la TB_{NEH}, TB_{FF} y TB_C en un 17.37 %, 16.25 % y 12.92 % respectivamente.

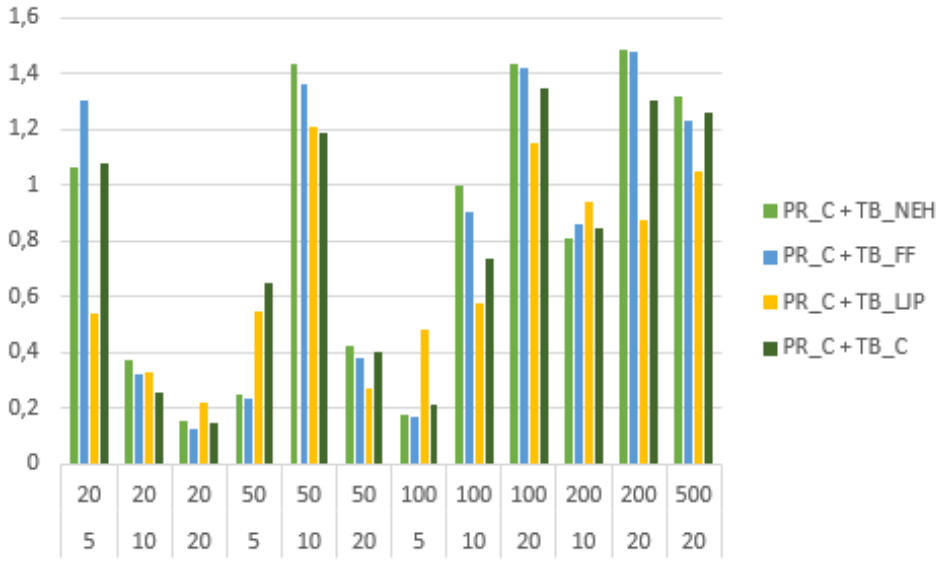


Figura 5.8 ARPD de cada *tie breaking rule* implementada con la PR_C en función de la dimensión de los problemas.

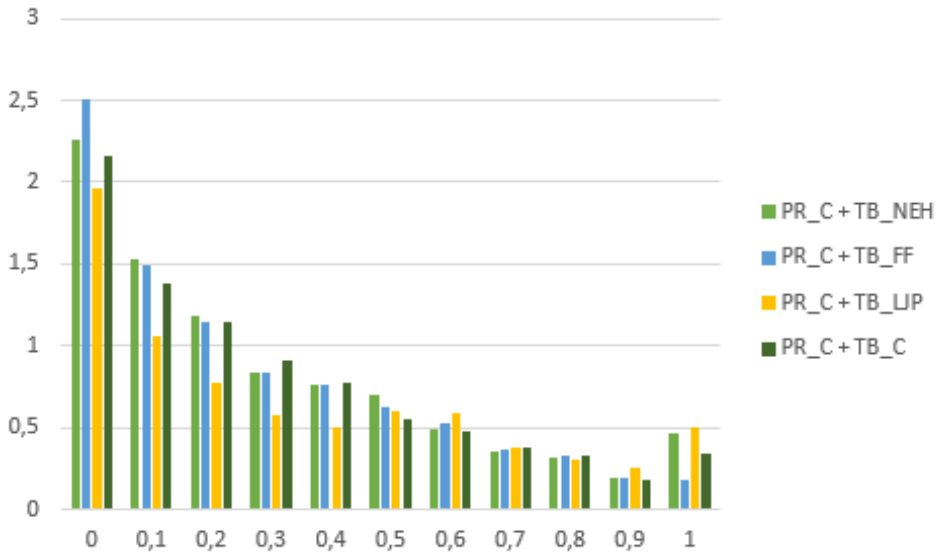


Figura 5.9 ARPD de cada *tie breaking rule* implementada con la PR_C en función de los pesos de los problemas.

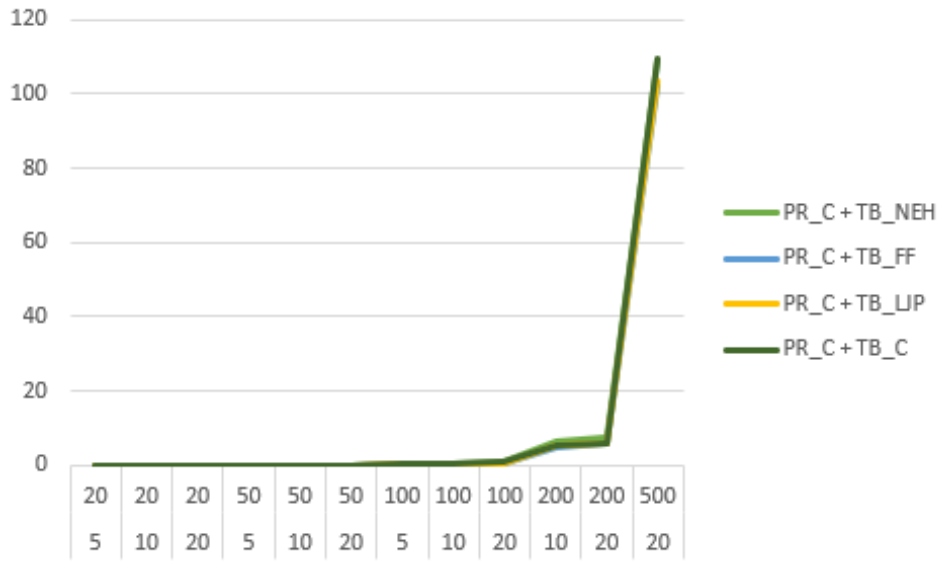


Figura 5.10 Tiempos de ejecución de cada método implementado con la PR_C.

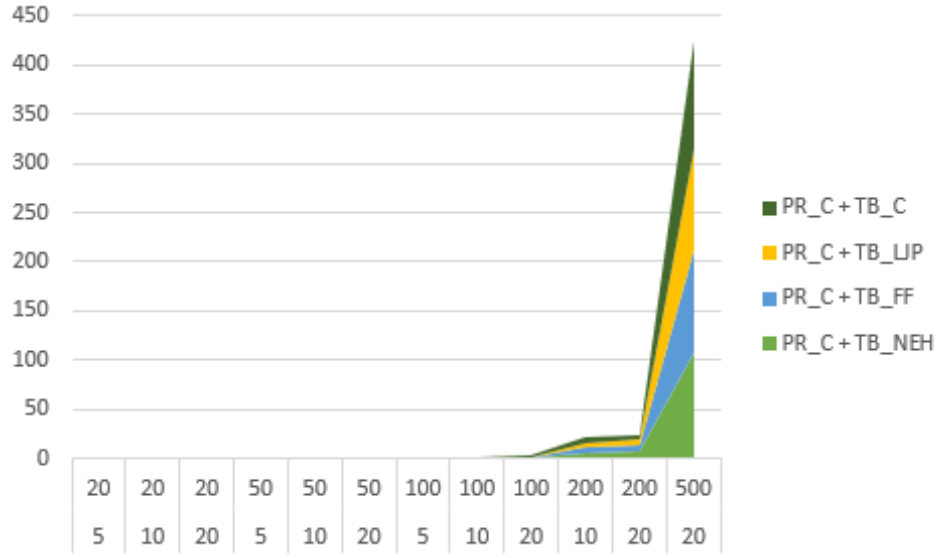


Figura 5.11 Tiempos de ejecución acumulados de cada método implementado con la PR_C.

A continuación, se ha implementado la TB_C en el algoritmo NEH_{LJP} y se ha comparado con los métodos originales (NEH, NEH_{FF} y NEH_C). Los resultados de las tablas 5.10 y 5.11 muestran que la nueva versión del algoritmo NEH_{LJP} es superior al resto, con un ARPD promedio inferior al NEH_C en un 4.99 %, destacando en los problemas de dimensión 5x20, 20x50, 10x100, 20x200 y 20x500 (Figura 5.12), así como para las ponderaciones de valor 0, 0.1, 0.2, 0.3, 0.4 y 0.7 (Figura 5.13).

Tabla 5.10 ARPD del método NEH_{LJP} implementado con la PR_C en función de la dimensión de los problemas.

m	n	NEH	NEH_{FF}	PR_c + TB_{LJP}	NEH_C
5	20	12,14	12,10	2,27	2,76
10	20	2,52	2,44	2,32	2,26
20	20	2,17	2,16	1,82	1,74
5	50	0,73	0,69	2,53	2,64
10	50	3,38	3,35	3,58	3,51
20	50	2,63	2,60	1,21	1,34
5	100	0,96	0,95	1,10	0,83
10	100	2,25	2,23	1,00	1,16
20	100	1,57	1,50	2,12	2,31
10	200	1,50	1,54	1,78	1,69
20	200	2,44	2,38	1,10	1,53
20	500	2,09	2,03	1,04	1,25

Tabla 5.11 ARPD del método NEH_{LJP} implementado con la PR_C en función de los pesos de los problemas.

w	NEH	NEH_{FF}	PR_c + TB_{LJP}	NEH_C
0	14,39	14,39	4,93	5,04
0,1	4,42	4,43	3,10	3,44
0,2	2,67	2,66	2,47	2,84
0,3	2,16	2,06	2,06	2,40
0,4	1,92	1,96	1,37	1,64
0,5	1,53	1,51	1,56	1,52
0,6	1,31	1,34	1,51	1,39
0,7	0,98	0,96	0,93	0,93
0,8	0,85	0,82	0,89	0,91
0,9	0,62	0,61	0,61	0,54
1	0,67	0,41	0,62	0,46

Tabla 5.12 Tiempos de ejecución del método NEH_{LJP} implementado con la PR_C.

m	n	NEH	NEH_{FF}	PR_c + TB_{LJP}	NEH_C
5	20	0,0260	0,0163	0,0176	0,0116
10	20	0,0175	0,0047	0,0065	0,0050
20	20	0,0043	0,0054	0,0057	0,0063
5	50	0,0582	0,0580	0,0631	0,0594
10	50	0,0591	0,0615	0,0635	0,0661
20	50	0,0602	0,0717	0,0724	0,0797
5	100	0,4426	0,4530	0,4565	0,4652
10	100	0,4980	0,4991	0,4817	0,5067
20	100	0,5259	0,5518	0,5605	0,6004
10	200	3,9429	4,1290	4,1874	4,3793
20	200	4,1973	4,6374	4,8859	5,0065
20	500	77,3044	84,6991	86,2719	90,4058
Promedio		7,2614	7,9323	8,0894	8,4660

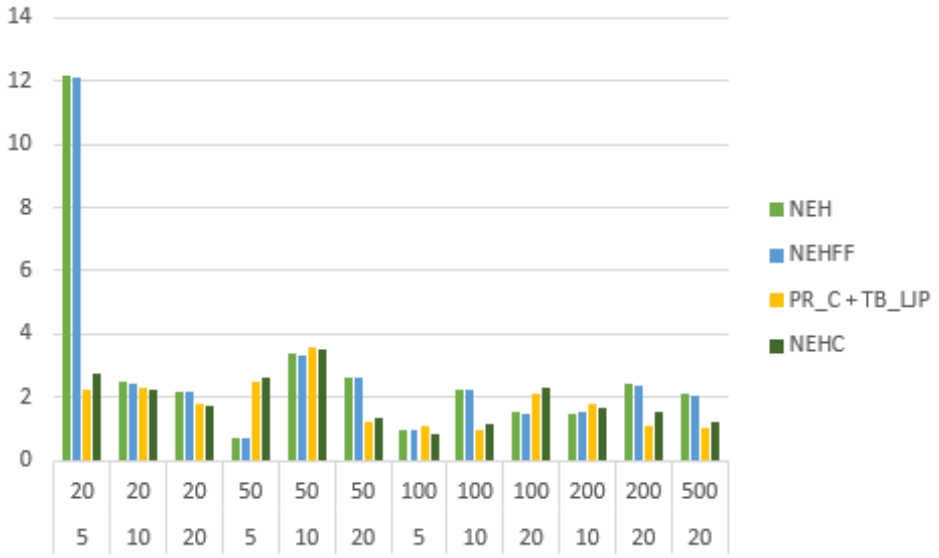


Figura 5.12 ARPD del método NEH_{LJP} implementado con la PR_C en función de la dimensión de los problemas.

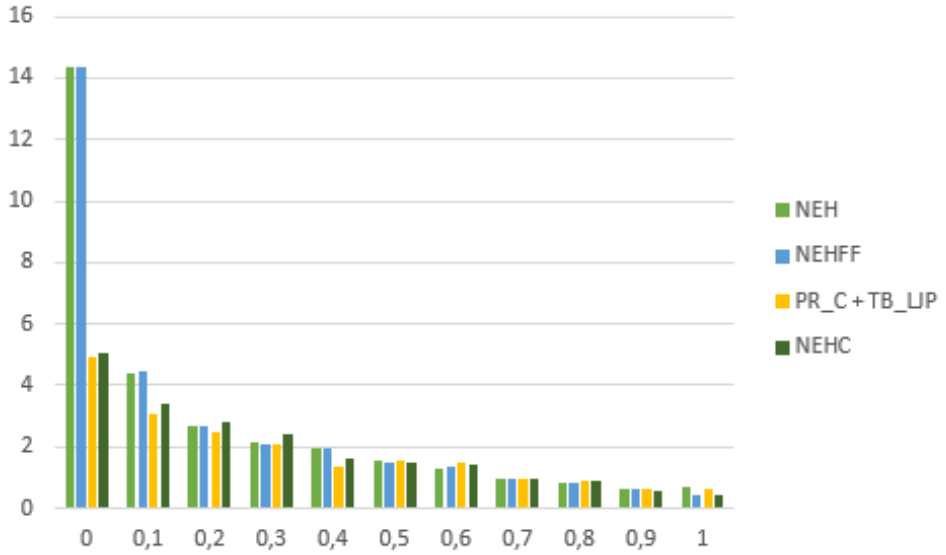


Figura 5.13 ARPD del método NEH_{LJP} implementado con la PR_C en función de los pesos de los problemas.

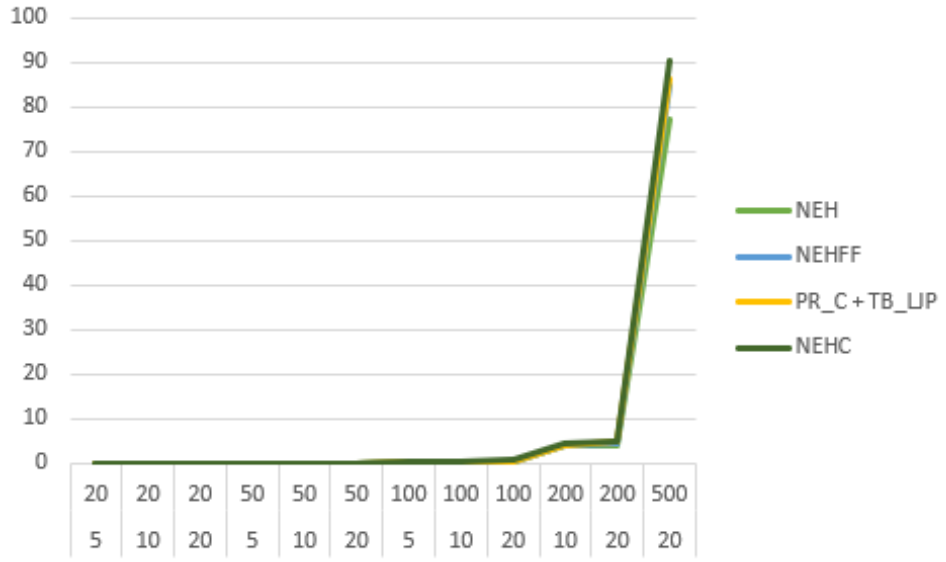


Figura 5.14 Tiempos de ejecución del método NEH_{LJP} implementado con la PR_C.

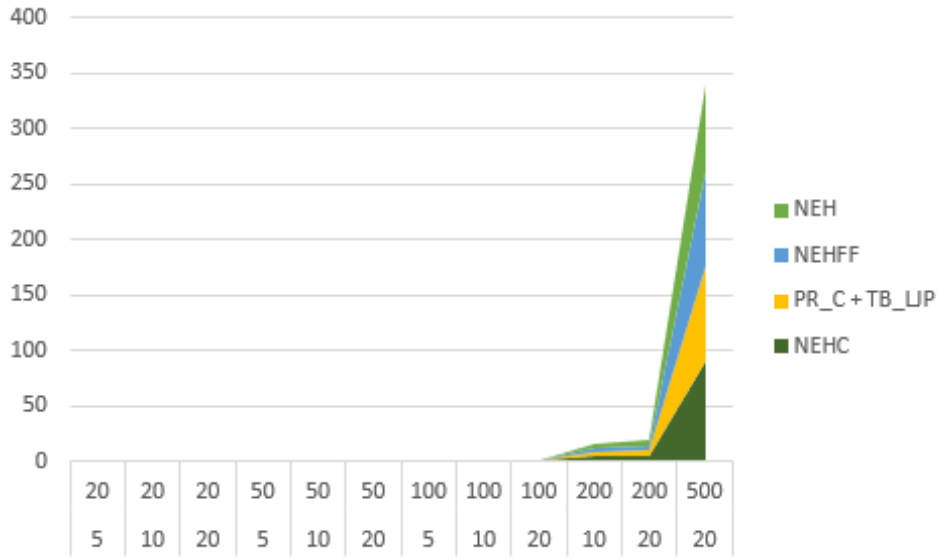


Figura 5.15 Tiempos de ejecución acumulados del método NEH_{LJP} implementado con la PR_C.

Uno de los efectos que se logra con la implementación de la PR_C es "suavizar" la función de los ARPD eliminando los "picos" provocados por los valores atípicos que aparecen en aquellos problemas en los que la ponderación es igual a 0, es decir, el objetivo es únicamente minimizar los *core idle-time*. Esto ocurre gracias a que se produce una mejor distribución en los trabajos, disminuyendo considerablemente los CIT y con ello el valor de la función objetivo. Además, no se producen cambios significativos en la calidad de los resultados, como se puede observar en la tabla 5.13:

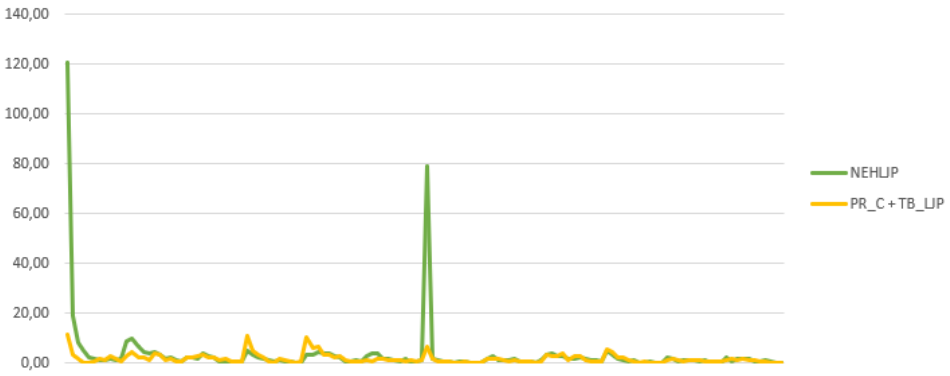


Figura 5.16 ARPD del método NEH_{LJP} original vs el NEH_{LJP} implementado con la PR_C.

Tabla 5.13 Diferencia entre la calidad de los resultados del NEH_{LJP} original y el NEH_{LJP} implementado con la PR_C.

	NEH_{LJP}	PR_C + TB_{LJP}	Variación
fo	5727,89	5730,53	-0,0005 %
$\mathbf{C}_{max}$	7045,52	7043,58	0,0275 %
$\sum CIT_i$	4770,15	4774,93	-0,1002 %

5.4 PR + TB_C

Por último, se ha implementado la TB_C en todos los algoritmos junto con sus *priority rules* originales, con el fin de averiguar si ésta es capaz de obtener mejores resultados en combinación con la *priority rule* de los demás métodos que las originales. Los resultados señalan de nuevo al NEH_C como el método que presenta una mayor calidad en sus soluciones, con un ARPD promedio un 30.21 % inferior al de la PR_{LJP} y un 25 % inferior al de la PR_{NEH}.

Tabla 5.14 ARPD de cada *priority rule* implementada con la TB_C en función de la dimensión de los problemas.

m	n	PR_{NEH} + TB_C	PR_{FF} + TB_C	PR_{LJP} + TB_C	PR_C + TB_C
5	20	8,48	8,48	15,82	2,42
10	20	3,35	3,35	4,20	3,12
20	20	3,18	3,18	1,64	2,73
5	50	0,97	0,97	2,02	2,79
10	50	4,09	4,09	2,30	4,34
20	50	3,16	3,16	1,88	1,91
5	100	2,45	2,45	2,97	0,97
10	100	2,00	2,00	1,19	1,32
20	100	1,65	1,65	2,48	2,57
10	200	1,72	1,72	1,32	1,82
20	200	2,29	2,29	1,20	1,32
20	500	2,03	2,03	0,98	1,23

Tabla 5.15 ARPD de cada *priority rule* implementada con la TB_C en función de los pesos de los problemas.

w	PR_{NEH} + TB_C	PR_{FF} + TB_C	PR_{LJP} + TB_C	PR_C + TB_C
0	13,08	13,08	16,96	5,16
0,1	4,73	4,73	4,56	4,04
0,2	2,96	2,96	3,25	3,36
0,3	2,58	2,58	2,37	2,85
0,4	2,33	2,33	1,77	2,10
0,5	1,84	1,84	1,66	1,88
0,6	1,55	1,55	1,24	1,58
0,7	1,14	1,14	1,01	1,10
0,8	0,96	0,96	0,89	1,06
0,9	0,76	0,76	0,67	0,67
1	0,50	0,50	0,46	0,51

Tabla 5.16 Tiempos de ejecución de cada *priority rule* implementada con la TB_C.

m	n	PR_{NEH} + TB_C	PR_{FF} + TB_C	PR_{LJP} + TB_C	PR_C + TB_C
5	20	0,0121	0,0075	0,0102	0,0087
10	20	0,0061	0,0048	0,0075	0,0060
20	20	0,0068	0,0061	0,0075	0,0074
5	50	0,0714	0,0615	0,0750	0,0720
10	50	0,0754	0,0663	0,0820	0,0786
20	50	0,0980	0,0793	0,1018	0,0976
5	100	0,5625	0,4705	0,5533	0,5624
10	100	0,7264	0,5145	0,6124	0,6058
20	100	0,8835	0,6224	0,7254	0,7170
10	200	7,2422	4,2696	5,2063	5,1902
20	200	8,7665	4,9436	5,9804	6,0395
20	500	115,2004	100,4001	109,0891	112,3668
Promedio		11,1376	9,2872	10,2042	10,4793

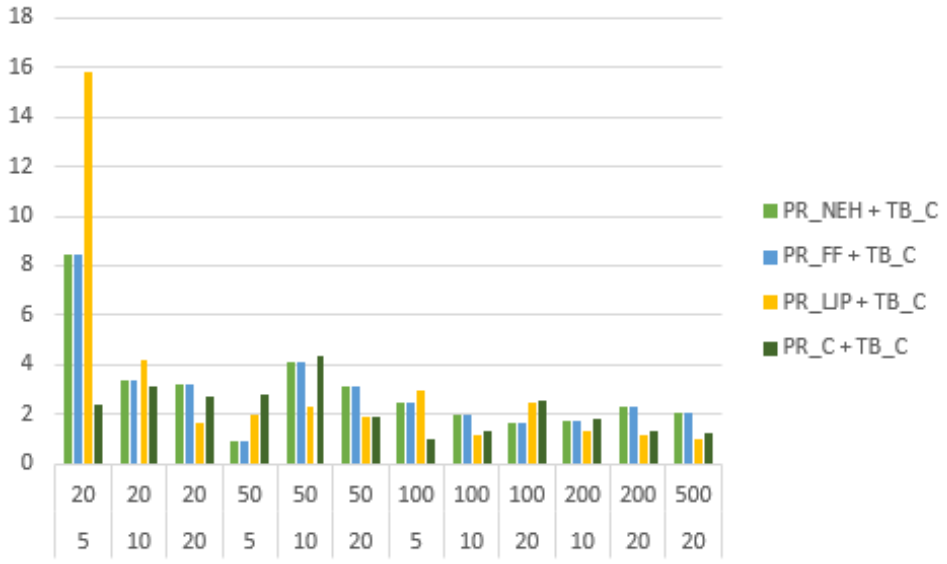


Figura 5.17 ARPD de cada *priority rule* implementada con la TB_C en función de la dimensión de los problemas.

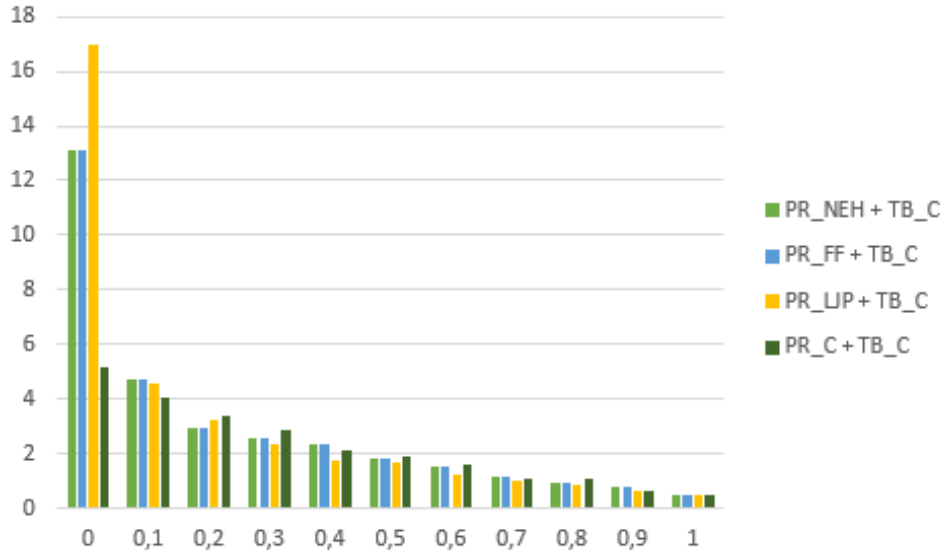


Figura 5.18 ARPD de cada *priority rule* implementada con la TB_C en función de los pesos de los problemas.

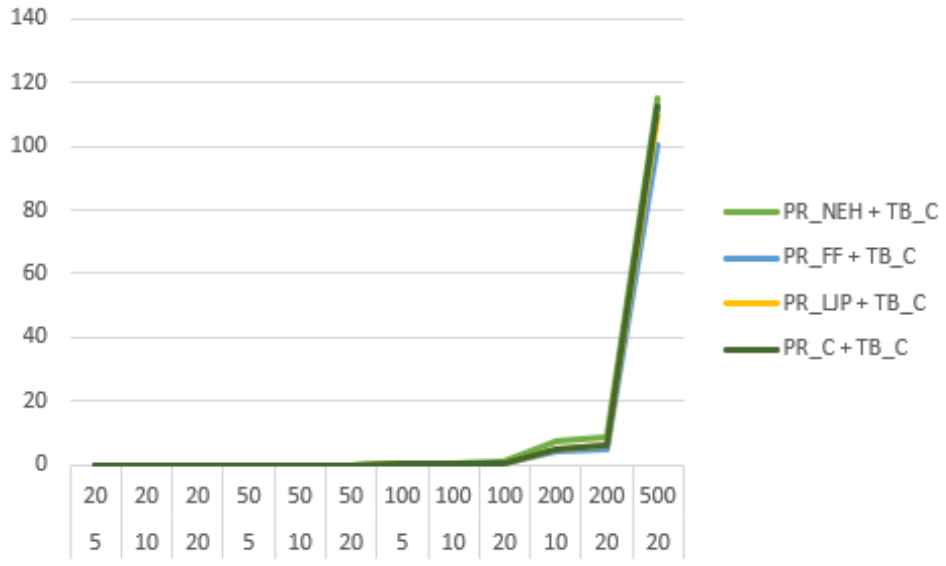


Figura 5.19 Tiempos de ejecución de cada *priority rule* implementada con la TB_C.

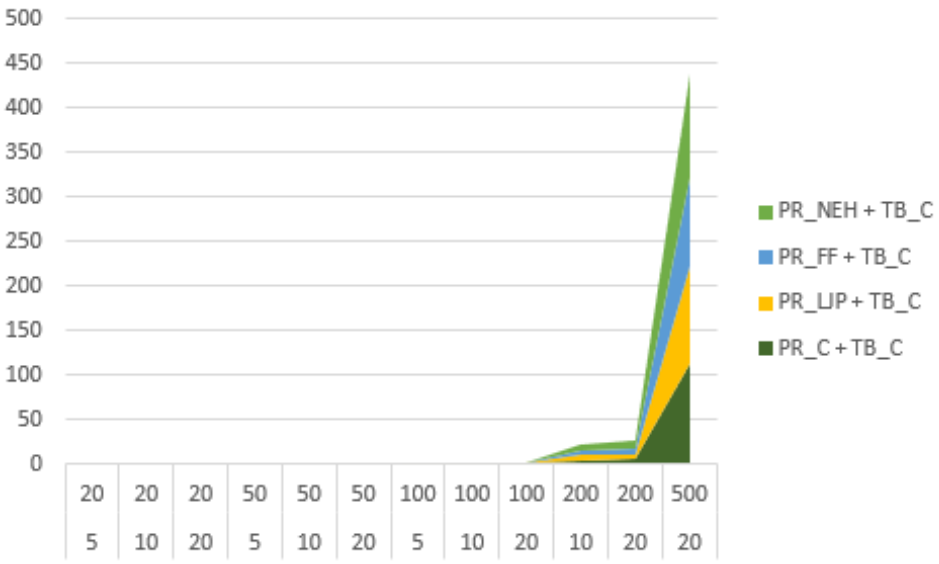


Figura 5.20 Tiempos de ejecución acumulados de cada *priority rule* implementada con la TB_C.

6 Conclusiones

En este capítulo se exponen las principales conclusiones alcanzadas en el presente Trabajo de Fin de Grado, en el que se aborda el problema de programación de operaciones en un taller de tipo *flowshop* con permutación mediante la proposición de un nuevo algoritmo basado en la heurística del NEH, llamado NEH_C. Para ello, se han llevado a cabo los siguientes pasos, resumidos a continuación:

- En primer lugar, a lo largo de los capítulos 1 y 2, se ha presentado el problema junto con los objetivos y las particularidades del modelo, así como la justificación del mismo, introduciendo previamente los conceptos básicos relativos a la programación de operaciones.
- A continuación, se han introducido los métodos con los cuales se ha llevado a cabo la comparación de los resultados, así como el nuevo algoritmo propuesto, detallando su funcionamiento y las reglas de prioridad y desempate que lo componen (capítulo 3). En el capítulo 4 se han presentado los problemas utilizados para las pruebas, así como los indicadores en función de los cuales se han evaluado los resultados obtenidos.
- Finalmente, en el capítulo 5 se muestran y analizan los resultados obtenidos.

Tras el análisis de los resultados, se pueden realizar las siguientes afirmaciones:

- El método NEH_C permite obtener soluciones de mayor calidad para el problema propuesto que los métodos NEH, NEH_{FF} y NEH_{LJP}, destacando de forma significativa en el caso $w = 0$, correspondiente al objetivo de *total core idle-time* ($min\ \sum_{i=0}^{m} CIT_i$). No obstante, sus tiempos de ejecución tienden a aumentar más en proporción al resto cuanto mayor es la dimensión del problema, siendo el NEH el mejor algoritmo en términos de tiempo.
- La nueva regla de prioridad propuesta en este trabajo, denominada PR_C, es capaz de mejorar el método NEH_{LJP} al ser implementada junto con su *tie-breaking rule*, TB_{LJP}.

Anexo: Códigos de Matlab

PR

PR_{NEH}

```
function [seqinic]=seqinicialneh(seq,pij,m,n)

\%Creo dos vectores, uno que almacene los índices de los trabajos

\%y otro con sus valores a ordenar asociados:

seqaux=[1:n];

seqini=[1:n];

\%Relleno el vector con los valores a ordenar:

for j=1:n

seqaux(j)=sumapj(j,seq,pij,m);

end

\%Y ahora lo ordeno:

seqinic=ordenamam(seqini,seqaux,n);

end
```

PR$_{LJP}$

```
function [seqinic]=seqinicialnehljp(seq,pij,m,n)

\%Creo dos vectores, uno que almacene los índices de los trabajos

\%y otro con sus valores a ordenar asociados:

seqaux=[1:n];

seqini=[1:n];

\%Relleno el vector con los valores a ordenar:

for j=1:n

seqaux(j)=(sumapj(j,seq,pij,m)/m)+(desviacionabsolutapj(j,seq,pij,m)/m)+
(skepj(j,seq,pij,m)^(1/3))+(1/(kurpj(j,seq,pij,m)^(1/4)));

end

\%Y ahora lo ordeno:

seqinic=ordenamam(seqini,seqaux,n);

end
```

PR$_{C}$

```
function [seqinic]=seqinicialnehc(seq,pij,m,n)

\%Creo dos vectores, uno que almacene los índices de los trabajos

\%y otro con sus valores a ordenar asociados:

seqaux=[1:n];

seqini=[1:n];

\%Relleno el vector con los valores a ordenar:

for j=1:n

seqaux(j)=sumapj(j,seq,pij,m)+desviacionabsolutapj(j,seq,pij,m)+rangopj(j,seq,pij,m);

end

\%Y ahora lo ordeno:

seqinic=ordenamam(seqini,seqaux,n);
```

```
end
```

TB

TB_{NEH}

```
seqf=[seqi(1),seqi(2)];

\%Primero compruebo que la subsecuencia sea la mejor de las dos posibles:

seqfalt=[seqi(2),seqi(1)];

\%Evalúo la función objetivo de ambas subsecuencias:

[c,cit]=implementa(seqf,pij,m,2);

cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo1=funcionobjetivo(w,cmax,sumacit);

[c,cit]=implementa(seqfalt,pij,m,2);

cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo2=funcionobjetivo(w,cmax,sumacit);

\%Y comparo:

if fo2<fo1

seqf=seqfalt;

end

\%Una vez tengo la subsecuencia óptima inserto el resto de trabajos:

for i=3:n

\%Hago la inserción del siguiente trabajo

seqf(i)=seqi(i);

\%Ahora saco todas las posibles combinaciones:

combi=combinaciones(seqf,i);
```

```
\%Busco el menor valor de la FO:

\%Empiezo por darle el menor a la primera combinacion:

[c,cit]=implementa(combi(1,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

minfo=funcionobjetivo(w,cmax,sumacit);

\%Ahora busco una FO menor:

for j=2:i

[c,cit]=implementa(combi(j,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

foaux=funcionobjetivo(w,cmax,sumacit);

if foaux<minfo

seqf=combi(j,1:i);

minfo=foaux;

end

end

end
```

TB$_{FF}$

```
seqf=[seqi(1),seqi(2)];

\%Primero compruebo que la subsecuencia sea la mejor de las dos posibles:

seqfalt=[seqi(2),seqi(1)];

\%Evalúo la función objetivo de ambas subsecuencias:

\%Guardo el valor del IT por si fuera necesario resolver un desempate

[c,cit]=implementa(seqf,pij,m,2);

cmax=makespan(c,2);
```

```
sumacit=sumcit(cit,m);

fo1=funcionobjetivo(w,cmax,sumacit);

tit1=tbrnehff(seqf,pij,m,sumacit);

[c,cit]=implementa(seqfalt,pij,m,2);

cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo2=funcionobjetivo(w,cmax,sumacit);

tit2=tbrnehff(seqfalt,pij,m,sumacit);

\%Y comparo:

if fo2<fo1

seqf=seqfalt;

end

if fo2==fo1 && tit2<tit1

seqf=seqfalt;

end

\%Una vez tengo la subsecuencia óptima inserto el resto de trabajos:

for i=3:n

\%Hago la inserción del siguiente trabajo

seqf(i)=seqi(i);

\%Ahora saco todas las posibles combinaciones

combi=combinaciones(seqf,i);

\%Busco el menor valor de la FO:

\%Empiezo por darle el menor a la primera combinacion:

[c,cit]=implementa(combi(1,1:i),pij,m,i);

cmax=makespan(c,i);
```

```
sumacit=sumcit(cit,m);

minfo=funcionobjetivo(w,cmax,sumacit);

\%Ahora busco una FO menor:

for j=2:i

[c,cit]=implementa(combi(j,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

foaux=funcionobjetivo(w,cmax,sumacit);

if foaux<minfo

minfo=foaux;

end

end

\%ahora que tengo la menor funcion objetivo elijo de entre todas la

\%que mejor satisfaga la condición de la TB:

menorit=inf;

for j=1:i

[c,cit]=implementa(combi(j,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

foaux=funcionobjetivo(w,cmax,sumacit);

titaux=tbrnehff(combi(j,1:i),pij,m,sumacit);

if foaux==minfo && titaux<menorit

seqf=combi(j,1:i);

menorit=titaux;

end
```

```
end

end
```

```
function [tit]=tbrnehff(seq,pij,m,sumacit)

\%Primero calculo los front idle-time:

fit=zeros(1,m);

fit(1)=0;

for j=2:m

fit(j)=fit(j-1)+pij(j-1,seq(1));

end

\%Ahora la suma:

sumfit=0;

for j=1:m

sumfit=sumfit+fit(j);

end

\%Y la suma del fit con el cit:

tit=sumfit+sumacit;

end
```

TB$_{LJP}$

```
seqf=[seqi(1),seqi(2)];

\%Primero compruebo que la subsecuencia sea la mejor de las dos posibles:

seqfalt=[seqi(2),seqi(1)];

\%Evalúo la función objetivo de ambas subsecuencias:

\%Guardo el valor de "p" por si fuera necesario resolver un desempate:

[c,cit]=implementa(seqf,pij,m,2);
```

```
cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo1=funcionobjetivo(w,cmax,sumacit);

p1=tbrnehljp(seqf,pij,2,c,cmax);

[c,cit]=implementa(seqfalt,pij,m,2);

cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo2=funcionobjetivo(w,cmax,sumacit);

p2=tbrnehljp(seqfalt,pij,2,c,cmax);

\%Y comparo:

if fo2<fo1

seqf=seqfalt;

end

if fo2==fo1 && p2>p1

seqf=seqfalt;

end

\%Una vez tengo la subsecuencia óptima inserto el resto de trabajos:

for i=3:n

\%Hago la inserción del siguiente trabajo

seqf(i)=seqi(i);

\%Ahora saco todas las posibles combinaciones

combi=combinaciones(seqf,i);

\%Busco el menor valor de la FO:

\%Empiezo por darle el menor a la primera combinacion:

[c,cit]=implementa(combi(1,1:i),pij,m,i);
```

```
cmax=makespan(c,i);

sumacit=sumcit(cit,m);

minfo=funcionobjetivo(w,cmax,sumacit);

\%Ahora busco una FO menor:

for j=2:i

[c,cit]=implementa(combi(j,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

foaux=funcionobjetivo(w,cmax,sumacit);

if foaux<minfo

minfo=foaux;

end

end

\%ahora que tengo la menor funcion objetivo elijo de entre todas la

\%que mejor satisfaga la condición de la TB:

mayorp=0;

for j=1:i

[c,cit]=implementa(combi(j,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

foaux=funcionobjetivo(w,cmax,sumacit);

paux=tbrnehljp(combi(j,1:i),pij,i,c,cmax);

if foaux==minfo && paux>mayorp

seqf=combi(j,1:i);

mayorp=paux;
```

```
end

end

end
```

```
function [p]=tbrnehljp(seq,pij,i,c,cmax)

\%Primero calculo los completion times en la primera máquina:

cm1=zeros(1,i);

cm1(1)=pij(1,seq(1));

for j=2:i

cm1(j)=cm1(j-1)+pij(1,seq(j));

end

\%A continuación calculo los flowtimes:

f=zeros(1,i);

f(1)=c(1);

for j=2:i

f(j)=c(j)-cm1(j-1);

end

\%Calculo la media de los flowtimes:

mediaf=0;

for j=1:i

mediaf=mediaf+f(j);

end

mediaf=mediaf/i;

\%Por último calculo el término del sumatorio:

terminosum=0;

for j=1:i
```

```
terminosum=terminosum+(f(j)-mediaf)^2;

end

\%Lo multiplico por 1/n-1, así luego es más cómodo al hacer la raíz:

terminosum=terminosum*(1/(i-1));

p=(mediaf-sqrt(terminosum))/cmax;
```

TB$_C$

```
seqf=[seqi(1),seqi(2)];

\%Primero compruebo que la subsecuencia sea la mejor de las dos posibles:

seqfalt=[seqi(2),seqi(1)];

\%Evalúo la función objetivo de ambas subsecuencias:

\%Guardo el valor de "u" por si fuera necesario resolver
un desempate

[c,cit]=implementa(seqf,pij,m,2);

cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo1=funcionobjetivo(w,cmax,sumacit);

u1=tbrnehc(seqf,pij,m,2,cit);

[c,cit]=implementa(seqfalt,pij,m,2);

cmax=makespan(c,2);

sumacit=sumcit(cit,m);

fo2=funcionobjetivo(w,cmax,sumacit);

u2=tbrnehc(seqfalt,pij,m,2,cit);

\%Y comparo:

if fo2<fo1

seqf=seqfalt;

end
```

```
if fo2==fo1 && u2>u1

seqf=seqfalt;

end

\%Una vez tengo la subsecuencia óptima inserto el resto de trabajos:

for i=3:n

\%Hago la inserción del siguiente trabajo

seqf(i)=seqi(i);

\%Ahora saco todas las posibles combinaciones

combi=combinaciones(seqf,i);

\%Busco el menor valor de la FO:

\%Empiezo por darle el menor a la primera combinacion:

[c,cit]=implementa(combi(1,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

minfo=funcionobjetivo(w,cmax,sumacit);

\%Ahora busco una FO menor:

for j=2:i

[c,cit]=implementa(combi(j,1:i),pij,m,i);

cmax=makespan(c,i);

sumacit=sumcit(cit,m);

foaux=funcionobjetivo(w,cmax,sumacit);

if foaux<minfo

minfo=foaux;

end

end
```

```
\%ahora que tengo la menor funcion objetivo elijo de entre todas la
\%que mejor satisfaga la condición de la TB:
mayoru=0;
for j=1:i
[c,cit]=implementa(combi(j,1:i),pij,m,i);
cmax=makespan(c,i);
sumacit=sumcit(cit,m);
foaux=funcionobjetivo(w,cmax,sumacit);
uaux=tbrnehc(combi(j,1:i),pij,m,i,cit);
if foaux==minfo && uaux>mayoru
seqf=combi(j,1:i);
mayoru=uaux;
end
end
end
```

```
function [u]=tbrnehc(seq,pij,m,n,cit)
u=0;
for i=1:m
u=u+(sumapi(i,seq,pij,n)/(sumapi(i,seq,pij,n)+cit(i)));
end
u=u/m;
end
```

ARPD

RPD

```
\%Inicio el flag para recorrer las posiciones de las instancias:

flag=0;

\%Creo la matriz de RPD:

mrpd=zeros(1320,7);

\%Inicio un segundo flag para rellenar la matriz de RPD:

flag2=1;

\%Recorro cada uno de los 120 problemas:

for s=1:120

n=ta(1+flag);

m=ta(2+flag);

\%Relleno la tabla con las combinaciones de w por cada problema:

for k=0:10

mrpd(flag2,1)=m;

mrpd(flag2,2)=n;

w=0.1*k;

mrpd(flag2,3)=w;

flag2=flag2+1;

end

flag=2+flag+(m*n);

end

\%Ahora calculo el RPD de cada problema:

for i=1:1320

bfo=bestfo(msolneh(i,4),msolnehff(i,4),msolnehljp(i,4),msolnehc(i,4));
```

```
mrpd(i,4)=((msolneh(i,4)-bfo)/bfo)*100;

mrpd(i,5)=((msolnehff(i,4)-bfo)/bfo)*100;

mrpd(i,6)=((msolnehljp(i,4)-bfo)/bfo)*100;

mrpd(i,7)=((msolnehc(i,4)-bfo)/bfo)*100;

end
```

ARPD

```
\%Creo una matriz para los ARPD de cada problema (marpd), otra

\%para el total (marpdt) y una última para los ARPD de cada w (marpdw):

marpdp=zeros(12,6);

marpdt=zeros(1,4);

marpdw=zeros(11,5);

\%Calculo los ARPD de cada problema:

flag=1;

for i=1:12

marpdp(i,1)=mrpd(flag,1);

marpdp(i,2)=mrpd(flag,2);

for j=flag:flag+109

marpdp(i,3)=marpdp(i,3)+mrpd(j,4);

marpdp(i,4)=marpdp(i,4)+mrpd(j,5);

marpdp(i,5)=marpdp(i,5)+mrpd(j,6);

marpdp(i,6)=marpdp(i,6)+mrpd(j,7);

end

marpdp(i,3)=marpdp(i,3)/110;

marpdp(i,4)=marpdp(i,4)/110;

marpdp(i,5)=marpdp(i,5)/110;

marpdp(i,6)=marpdp(i,6)/110;
```

```
flag=j+1;

end

\%Ahora calculo la media de todos los ARPD:

for i=1:12

marpdt(1,1)=marpdt(1,1)+marpdp(i,3);

marpdt(1,2)=marpdt(1,2)+marpdp(i,4);

marpdt(1,3)=marpdt(1,3)+marpdp(i,5);

marpdt(1,4)=marpdt(1,4)+marpdp(i,6);

end

marpdt(1,1)=marpdt(1,1)/12;

marpdt(1,2)=marpdt(1,2)/12;

marpdt(1,3)=marpdt(1,3)/12;

marpdt(1,4)=marpdt(1,4)/12;

\%Por último calculo los ARPD por ponderaciones:

for i=1:11

marpdw(i,1)=0.1*(i-1);

for j=i:11:1320

marpdw(i,2)=marpdw(i,2)+mrpd(j,4);

marpdw(i,3)=marpdw(i,3)+mrpd(j,5);

marpdw(i,4)=marpdw(i,4)+mrpd(j,6);

marpdw(i,5)=marpdw(i,5)+mrpd(j,7);

end

marpdw(i,2)=marpdw(i,2)/120;

marpdw(i,3)=marpdw(i,3)/120;

marpdw(i,4)=marpdw(i,4)/120;
```

```
marpdw(i,5)=marpdw(i,5)/120;

end
```

CPU

```
\%Creo una matriz para los CPU de cada problema (mcpu) y otra para el total (mcput):

mcpu=zeros(12,6);

mcput=zeros(1,4);

\%Calculo los CPU de cada problema:

flag=1;

for i=1:12

mcpu(i,1)=mrpd(flag,1);

mcpu(i,2)=mrpd(flag,2);

for j=flag:flag+109

mcpu(i,3)=mcpu(i,3)+msolneh(j,7);

mcpu(i,4)=mcpu(i,4)+msolnehff(j,7);

mcpu(i,5)=mcpu(i,5)+msolnehljp(j,7);

mcpu(i,6)=mcpu(i,6)+msolnehc(j,7);

end

mcpu(i,3)=mcpu(i,3)/110;

mcpu(i,4)=mcpu(i,4)/110;

mcpu(i,5)=mcpu(i,5)/110;

mcpu(i,6)=mcpu(i,6)/110;

flag=j+1;

end

\%Ahora calculo la media de todos los CPU:

for i=1:12
```

```
mcput(1,1)=mcput(1,1)+mcpu(i,3);

mcput(1,2)=mcput(1,2)+mcpu(i,4);

mcput(1,3)=mcput(1,3)+mcpu(i,5);

mcput(1,4)=mcput(1,4)+mcpu(i,6);

end

mcput(1,1)=mcput(1,1)/12;

mcput(1,2)=mcput(1,2)/12;

mcput(1,3)=mcput(1,3)/12;

mcput(1,4)=mcput(1,4)/12;
```

Funciones auxiliares

Valor de la función objetivo

```
function [fo]=funcionobjetivo(w,cmax,sumacit)

fo=(w*cmax)+((1-w)*sumacit);

end
```

Identificación de la mejor solución

```
function [bfo]=bestfo(neh,nehff,nehljp,nehc)

soluciones=[neh,nehff,nehljp,nehc];

bfo=min(soluciones);

end
```

Intercambio de posiciones en vectores

```
function [v]=cambio(v,i,j)

vaux=[1,2];

vaux(1)=v(i);

vaux(2)=v(j);

v(i)=vaux(2);

v(j)=vaux(1);

end
```

Combinaciones para una secuencia de *n* elementos

```
function [combi]=combinaciones(seqf,i)

combi=zeros(i,i);

\%Copio seqf:

for j=1:i

for k=1:i

combi(j,k)=seqf(k);

end
```

```
end

\%Ahora creo las i combinaciones:

flag=1;

for j=2:i

for k=flag:i-1

combi(j,1:i)=cambio(combi(j,1:i),k,i);

end

flag=flag+1;

end

end
```

Desviación absoluta

```
function[desvabs]=desviacionabsolutapj(j,seq,pij,m)

desvabs=0;

for i=1:m

desvabs=desvabs+abs(pij(i,seq(j))-mediapj(j,seq,pij,m));

end
```

***Completion times* y CIT de una secuencia**

```
function [c,cit]=implementa(seq,pij,m,n)

\%Creo un vector para el makespan y otro para el CIT:

c=[1:n];

cit=[1:m];

for j=1:n

c(j)=0;

end

for i=1:m

cit(i)=0;
```

```
end

\%Primero preparo el trabajo 1 fuera del bucle porque no tiene

\%antecesores y después meto el resto de trabajos en el bucle:

for i=1:m

c(1)=c(1)+pij(i,seq(1));

for j=2:n

if c(j-1)<c(j)

\%Si el trabajo anterior acabó antes que el nuevo, este debe

\%esperar a acabar y habrá que añadir el CIT:

cit(i)=cit(i)+(c(j)-c(j-1));

c(j)=c(j)+pij(i,seq(j));

end

if c(j-1)==c(j)

\%Si coinciden simplemente se suma el tiempo de proceso:

c(j)=c(j-1)+pij(i,seq(j));

end

if c(j-1)>c(j)

\%Si el c del trabajo anterior supera al nuevo, entonces el

\%nuevo trabajo va inmediatamente después de este:

c(j)=c(j-1)+pij(i,seq(j));

end

end

end

end
```

Makespan

```
function [cmax]=makespan(c,n)

cmax=0;

for j=1:n

if c(j)>cmax

cmax=c(j);

end

end

end
```

Suma de los CIT de todas las máquinas

```
function [sumacit]=sumcit(cit,m)

sumacit=0;

for i=1:m

sumacit=sumacit+cit(i);

end

end
```

Kurtosis

```
function [kur]=kurpj(j,seq,pij,m)

numerador=0;

for i=1:m

numerador=numerador+(pij(i,seq(j))-(mediapj(j,seq,pij,m)))^4;

end

numerador=numerador/m;

denominador=0;

for i=1:m

denominador=denominador+(pij(i,seq(j))-(mediapj(j,seq,pij,m)))^2;
```

```
end

denominador=denominador/m;

denominador=denominador^;

kur=numerador/denominador;

end
```

Skewness

```
function[ske]=skepj(j,seq,pij,m)

numerador=0;

for i=1:m

numerador=numerador+(pij(i,seq(j))-(mediapj(j,seq,pij,m)))^3;

end

numerador=numerador/m;

denominador=0;

for i=1:m

denominador=denominador+(pij(i,seq(j))-(mediapj(j,seq,pij,m)))^2;

end

denominador=denominador/m;

denominador=sqrt(denominador);

denominador=denominador^3;

ske=numerador/denominador;

end
```

Media de los p_{ij} de un trabajo

```
function [mediapj]=mediapj(j,seq,pij,m)

aux=sumapj(j,seq,pij,m);

mediapj=aux/m;

end
```

Suma de los p_{ij} de un trabajo

```
function [sumpj]=sumapj(j,seq,pij,m)

sumpj=0;

for i=1:m

sumpj=sumpj+pij(i,seq(j));

end

end
```

Ordenar una secuencia de mayor a menor en función de sus p_{pij}

```
function [seqorden]=ordenamam(seq,seqaux,n)

\%copio el vector de la secuencia y de los pij

seqorden=[1:n];

seqaux2=[1:n];

for i=1:n

seqorden(i)=seq(i);

end

for i=1:n

seqaux2(i)=seqaux(i);

end

\%ahora repaso el vector de pij para ordenarlos

for i=1:n

for j=1:n

if seqaux2(i)>seqaux2(j)

seqaux2=cambio(seqaux2,i,j);

seqorden=cambio(seqorden,i,j);

end

end
```

```
end

end
```

Rango de los p_{ij} de un trabajo

```
function [rango]=rangopj(j,seq,pij,m)

\%Pongo como valor de referencia el primer componente para poder comparar:

max=pij(1,seq(j));

min=pij(1,seq(j));

for i=1:m

if pij(i,seq(j))>max

max=pij(i,seq(j));

end

if pij(i,seq(j))<min

min=pij(i,seq(j));

end

rango=max-min;

end
```

Ejemplo de implementación del algoritmo NEH$_C$

```
\%Cargo las instancias:

ta=importdata('taillardbenchmark.txt');

\%Inicio el flag para recorrer las posiciones de las instancias:

flag=0;

\%Creo la matriz de las soluciones:

msolnehc=zeros(1320,7);

\%Inicio un segundo flag para rellenar la matriz de soluciones:

flag2=1;

\%Recorro cada uno de los 120 problemas:

for s=1:120

\%Inicio el contador del tiempo de CPU:

tinicio1=cputime;

\%Incluyo los datos del problema (Es necesario adaptar el formato

\%de los tiempos de proceso)

n=ta(1+flag);

m=ta(2+flag);

dim=2+flag+(m*n);

pij=ta(3+flag:dim);

pij=reshape(pij,n,m);

pij=transpose(pij);

seq=[1:n];

\%Se ejecuta la priority rule:

seqi=seqinicialnehc(seq,pij,m,n);

\%tinicio1 es el tiempo que se tarda en resolver la secuencia inicial:

tinicio1=cputime-tinicio1;
```

```
\%Ahora relleno la tabla con las combinaciones de w por cada
\%problema:

for k=0:10

\%tinicio2 es el tiempo que se tarda en calcular la fo:

tinicio2=cputime;

msolnehc(flag2,1)=m;

msolnehc(flag2,2)=n;

w=0.1*k;

msolnehc(flag2,3)=w;

\%Se aplica la TB:
```

tiebreakingnehc;

```
\%Se implementa la secuencia final:

[c,cit]=implementa(seqf,pij,m,n);

\%Se calcula el makespan y el CIT:

cmax=makespan(c,n);

sumacit=sumcit(cit,m);

\%Se almacenan los resultados en la tabla:

msolnehc(flag2,4)=funcionobjetivo(w,cmax,sumacit);

msolnehc(flag2,5)=cmax;

msolnehc(flag2,6)=sumacit;

\%El tiempo de ejecución es la suma entre ambos tiempos:

msolnehc(flag2,7)=cputime-tinicio2+tinicio1;

flag2=flag2+1;

end

flag=dim;

end
```

Anexo: ARPD en función de las características del problema

ARPD de cada *priority rule* en función de las características de los problemas

Tabla 1 ARPD de cada *priority rule* en función de las características de los problemas.

m	n	w	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
5	20	0,0	6,66	6,66	32,48	57,11
5	20	0,1	5,24	5,24	18,39	33,27
5	20	0,2	4,31	4,31	12,83	23,28
5	20	0,3	3,63	3,63	9,60	17,23
5	20	0,4	3,13	3,13	7,40	13,01
5	20	0,5	2,73	2,73	5,78	9,83
5	20	0,6	2,46	2,46	4,56	7,37
5	20	0,7	2,35	2,35	3,69	5,47
5	20	0,8	2,31	2,31	3,04	3,94
5	20	0,9	2,28	2,28	2,49	2,65
5	20	1,0	2,36	2,36	2,13	1,64
10	20	0,0	9,06	9,06	17,32	23,25
10	20	0,1	8,21	8,21	15,39	20,79
10	20	0,2	7,38	7,38	13,54	18,40
10	20	0,3	6,57	6,57	11,76	16,10
10	20	0,4	5,82	5,82	10,09	13,93
10	20	0,5	5,09	5,09	8,48	11,80
10	20	0,6	4,37	4,37	6,91	9,72
10	20	0,7	3,67	3,67	5,39	7,68
10	20	0,8	2,97	2,97	3,91	5,66
10	20	0,9	2,81	2,81	2,96	4,19
10	20	1,0	2,86	2,86	2,22	2,90

m	n	w	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
20	20	0,0	3,37	3,37	5,64	7,44
20	20	0,1	3,26	3,26	5,44	7,19
20	20	0,2	3,13	3,13	5,21	6,90
20	20	0,3	2,98	2,98	4,94	6,57
20	20	0,4	2,81	2,81	4,62	6,18
20	20	0,5	2,63	2,63	4,27	5,74
20	20	0,6	2,43	2,43	3,85	5,21
20	20	0,7	2,18	2,18	3,32	4,56
20	20	0,8	1,88	1,88	2,67	3,75
20	20	0,9	1,51	1,51	1,81	2,69
20	20	1,0	1,54	1,54	1,17	1,77
5	50	0,0	15,48	15,48	21,02	35,73
5	50	0,1	10,37	10,37	15,82	26,83
5	50	0,2	7,49	7,49	12,28	20,80
5	50	0,3	5,63	5,63	9,69	16,38
5	50	0,4	4,33	4,33	7,68	12,98
5	50	0,5	3,36	3,36	6,09	10,27
5	50	0,6	2,62	2,62	4,78	8,06
5	50	0,7	2,10	2,10	3,77	6,28
5	50	0,8	1,71	1,71	2,94	4,80
5	50	0,9	1,54	1,54	2,38	3,68
5	50	1,0	1,53	1,53	2,00	2,82
10	50	0,0	4,51	4,51	2,43	7,34
10	50	0,1	4,24	4,24	2,20	6,67
10	50	0,2	3,98	3,98	1,97	6,00
10	50	0,3	3,72	3,72	1,75	5,33
10	50	0,4	3,48	3,48	1,54	4,69
10	50	0,5	3,26	3,26	1,34	4,05
10	50	0,6	3,03	3,03	1,14	3,42
10	50	0,7	2,82	2,82	0,94	2,80
10	50	0,8	2,64	2,64	0,77	2,20
10	50	0,9	2,53	2,53	0,68	1,67
10	50	1,0	2,53	2,53	0,68	1,25
20	50	0,0	3,34	3,34	6,61	5,84
20	50	0,1	3,25	3,25	6,40	5,62
20	50	0,2	3,15	3,15	6,14	5,37
20	50	0,3	3,04	3,04	5,85	5,08
20	50	0,4	2,90	2,90	5,51	4,75
20	50	0,5	2,73	2,73	5,11	4,34
20	50	0,6	2,53	2,53	4,61	3,86
20	50	0,7	2,28	2,28	4,00	3,25
20	50	0,8	1,96	1,96	3,24	2,49
20	50	0,9	1,58	1,58	2,27	1,54
20	50	1,0	1,33	1,33	1,24	0,52

m	n	w	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
5	100	0,0	0,97	0,97	10,47	13,06
5	100	0,1	0,79	0,79	7,44	9,30
5	100	0,2	0,68	0,68	5,66	7,00
5	100	0,3	0,62	0,62	4,47	5,44
5	100	0,4	0,57	0,57	3,61	4,31
5	100	0,5	0,53	0,53	2,96	3,45
5	100	0,6	0,51	0,51	2,46	2,78
5	100	0,7	0,52	0,52	2,08	2,27
5	100	0,8	0,53	0,53	1,77	1,85
5	100	0,9	0,56	0,56	1,54	1,51
5	100	1,0	0,64	0,64	1,40	1,29
10	100	0,0	4,32	4,32	12,75	14,11
10	100	0,1	3,88	3,88	11,33	12,53
10	100	0,2	3,47	3,47	9,98	11,03
10	100	0,3	3,07	3,07	8,68	9,60
10	100	0,4	2,70	2,70	7,44	8,22
10	100	0,5	2,34	2,34	6,24	6,89
10	100	0,6	2,00	2,00	5,09	5,61
10	100	0,7	1,68	1,68	3,98	4,38
10	100	0,8	1,36	1,36	2,91	3,19
10	100	0,9	1,08	1,08	1,89	2,06
10	100	1,0	1,23	1,23	1,32	1,37
20	100	0,0	0,76	0,76	11,05	8,85
20	100	0,1	0,73	0,73	10,63	8,52
20	100	0,2	0,70	0,70	10,16	8,15
20	100	0,3	0,67	0,67	9,61	7,72
20	100	0,4	0,63	0,63	8,97	7,22
20	100	0,5	0,60	0,60	8,23	6,66
20	100	0,6	0,57	0,57	7,36	5,98
20	100	0,7	0,53	0,53	6,29	5,15
20	100	0,8	0,48	0,48	4,97	4,12
20	100	0,9	0,42	0,42	3,27	2,81
20	100	1,0	0,75	0,75	1,42	1,46
10	200	0,0	9,09	9,09	5,22	6,01
10	200	0,1	8,15	8,15	4,68	5,37
10	200	0,2	7,25	7,25	4,16	4,75
10	200	0,3	6,39	6,39	3,65	4,16
10	200	0,4	5,57	5,57	3,17	3,59
10	200	0,5	4,78	4,78	2,71	3,05
10	200	0,6	4,01	4,01	2,26	2,52
10	200	0,7	3,28	3,28	1,83	2,02
10	200	0,8	2,57	2,57	1,43	1,54
10	200	0,9	1,98	1,98	1,11	1,16
10	200	1,0	1,53	1,53	0,95	0,92

m	n	w	PR_{NEH}	PR_{FF}	PR_{LJP}	PR_{C}
20	200	0,0	0,43	0,43	7,20	7,82
20	200	0,1	0,41	0,41	6,89	7,49
20	200	0,2	0,39	0,39	6,54	7,12
20	200	0,3	0,36	0,36	6,15	6,70
20	200	0,4	0,33	0,33	5,70	6,22
20	200	0,5	0,29	0,29	5,18	5,66
20	200	0,6	0,25	0,25	4,57	5,01
20	200	0,7	0,20	0,20	3,86	4,25
20	200	0,8	0,14	0,14	3,00	3,33
20	200	0,9	0,10	0,10	1,98	2,23
20	200	1,0	0,29	0,29	0,95	1,10
20	500	0,0	4,80	4,80	3,82	3,36
20	500	0,1	4,55	4,55	3,63	3,18
20	500	0,2	4,27	4,27	3,41	2,99
20	500	0,3	3,97	3,97	3,17	2,77
20	500	0,4	3,63	3,63	2,91	2,54
20	500	0,5	3,25	3,25	2,62	2,27
20	500	0,6	2,82	2,82	2,29	1,98
20	500	0,7	2,34	2,34	1,93	1,65
20	500	0,8	1,80	1,80	1,52	1,28
20	500	0,9	1,20	1,20	1,08	0,87
20	500	1,0	0,67	0,67	0,75	0,58
Promedio			2,85	2,85	5,34	6,94

ARPD de cada método en función de las características de los problemas

Tabla 2 ARPD de cada método en función de las características de los problemas.

m	n	w	NEH	NEH_{FF}	NEH_{LJP}	NEH_{C}
5	20	0,0	6,66	6,66	32,48	57,11
5	20	0,1	5,24	5,24	18,39	33,27
5	20	0,2	4,31	4,31	12,83	23,28
5	20	0,3	3,63	3,63	9,60	17,23
5	20	0,4	3,13	3,13	7,40	13,01
5	20	0,5	2,73	2,73	5,78	9,83
5	20	0,6	2,46	2,46	4,56	7,37
5	20	0,7	2,35	2,35	3,69	5,47
5	20	0,8	2,31	2,31	3,04	3,94
5	20	0,9	2,28	2,28	2,49	2,65
5	20	1,0	2,36	2,36	2,13	1,64
10	20	0,0	9,06	9,06	17,32	23,25
10	20	0,1	8,21	8,21	15,39	20,79
10	20	0,2	7,38	7,38	13,54	18,40
10	20	0,3	6,57	6,57	11,76	16,10
10	20	0,4	5,82	5,82	10,09	13,93
10	20	0,5	5,09	5,09	8,48	11,80
10	20	0,6	4,37	4,37	6,91	9,72
10	20	0,7	3,67	3,67	5,39	7,68
10	20	0,8	2,97	2,97	3,91	5,66
10	20	0,9	2,81	2,81	2,96	4,19
10	20	1,0	2,86	2,86	2,22	2,90
20	20	0,0	3,37	3,37	5,64	7,44
20	20	0,1	3,26	3,26	5,44	7,19
20	20	0,2	3,13	3,13	5,21	6,90
20	20	0,3	2,98	2,98	4,94	6,57
20	20	0,4	2,81	2,81	4,62	6,18
20	20	0,5	2,63	2,63	4,27	5,74
20	20	0,6	2,43	2,43	3,85	5,21
20	20	0,7	2,18	2,18	3,32	4,56
20	20	0,8	1,88	1,88	2,67	3,75
20	20	0,9	1,51	1,51	1,81	2,69
20	20	1,0	1,54	1,54	1,17	1,77
5	50	0,0	15,48	15,48	21,02	35,73
5	50	0,1	10,37	10,37	15,82	26,83
5	50	0,2	7,49	7,49	12,28	20,80
5	50	0,3	5,63	5,63	9,69	16,38
5	50	0,4	4,33	4,33	7,68	12,98
5	50	0,5	3,36	3,36	6,09	10,27
5	50	0,6	2,62	2,62	4,78	8,06
5	50	0,7	2,10	2,10	3,77	6,28
5	50	0,8	1,71	1,71	2,94	4,80

m	n	w	NEH	NEH_{FF}	NEH_{LJP}	NEH_C
5	50	0,9	1,54	1,54	2,38	3,68
5	50	1,0	1,53	1,53	2,00	2,82
10	50	0,0	4,51	4,51	2,43	7,34
10	50	0,1	4,24	4,24	2,20	6,67
10	50	0,2	3,98	3,98	1,97	6,00
10	50	0,3	3,72	3,72	1,75	5,33
10	50	0,4	3,48	3,48	1,54	4,69
10	50	0,5	3,26	3,26	1,34	4,05
10	50	0,6	3,03	3,03	1,14	3,42
10	50	0,7	2,82	2,82	0,94	2,80
10	50	0,8	2,64	2,64	0,77	2,20
10	50	0,9	2,53	2,53	0,68	1,67
10	50	1,0	2,53	2,53	0,68	1,25
20	50	0,0	3,34	3,34	6,61	5,84
20	50	0,1	3,25	3,25	6,40	5,62
20	50	0,2	3,15	3,15	6,14	5,37
20	50	0,3	3,04	3,04	5,85	5,08
20	50	0,4	2,90	2,90	5,51	4,75
20	50	0,5	2,73	2,73	5,11	4,34
20	50	0,6	2,53	2,53	4,61	3,86
20	50	0,7	2,28	2,28	4,00	3,25
20	50	0,8	1,96	1,96	3,24	2,49
20	50	0,9	1,58	1,58	2,27	1,54
20	50	1,0	1,33	1,33	1,24	0,52
5	100	0,0	0,97	0,97	10,47	13,06
5	100	0,1	0,79	0,79	7,44	9,30
5	100	0,2	0,68	0,68	5,66	7,00
5	100	0,3	0,62	0,62	4,47	5,44
5	100	0,4	0,57	0,57	3,61	4,31
5	100	0,5	0,53	0,53	2,96	3,45
5	100	0,6	0,51	0,51	2,46	2,78
5	100	0,7	0,52	0,52	2,08	2,27
5	100	0,8	0,53	0,53	1,77	1,85
5	100	0,9	0,56	0,56	1,54	1,51
5	100	1,0	0,64	0,64	1,40	1,29
10	100	0,0	4,32	4,32	12,75	14,11
10	100	0,1	3,88	3,88	11,33	12,53
10	100	0,2	3,47	3,47	9,98	11,03
10	100	0,3	3,07	3,07	8,68	9,60
10	100	0,4	2,70	2,70	7,44	8,22
10	100	0,5	2,34	2,34	6,24	6,89
10	100	0,6	2,00	2,00	5,09	5,61
10	100	0,7	1,68	1,68	3,98	4,38
10	100	0,8	1,36	1,36	2,91	3,19

m	n	w	NEH	NEH_{FF}	NEH_{LJP}	NEH_{C}
10	100	0,9	1,08	1,08	1,89	2,06
10	100	1,0	1,23	1,23	1,32	1,37
20	100	0,0	0,76	0,76	11,05	8,85
20	100	0,1	0,73	0,73	10,63	8,52
20	100	0,2	0,70	0,70	10,16	8,15
20	100	0,3	0,67	0,67	9,61	7,72
20	100	0,4	0,63	0,63	8,97	7,22
20	100	0,5	0,60	0,60	8,23	6,66
20	100	0,6	0,57	0,57	7,36	5,98
20	100	0,7	0,53	0,53	6,29	5,15
20	100	0,8	0,48	0,48	4,97	4,12
20	100	0,9	0,42	0,42	3,27	2,81
20	100	1,0	0,75	0,75	1,42	1,46
10	200	0,0	9,09	9,09	5,22	6,01
10	200	0,1	8,15	8,15	4,68	5,37
10	200	0,2	7,25	7,25	4,16	4,75
10	200	0,3	6,39	6,39	3,65	4,16
10	200	0,4	5,57	5,57	3,17	3,59
10	200	0,5	4,78	4,78	2,71	3,05
10	200	0,6	4,01	4,01	2,26	2,52
10	200	0,7	3,28	3,28	1,83	2,02
10	200	0,8	2,57	2,57	1,43	1,54
10	200	0,9	1,98	1,98	1,11	1,16
10	200	1,0	1,53	1,53	0,95	0,92
20	200	0,0	0,43	0,43	7,20	7,82
20	200	0,1	0,41	0,41	6,89	7,49
20	200	0,2	0,39	0,39	6,54	7,12
20	200	0,3	0,36	0,36	6,15	6,70
20	200	0,4	0,33	0,33	5,70	6,22
20	200	0,5	0,29	0,29	5,18	5,66
20	200	0,6	0,25	0,25	4,57	5,01
20	200	0,7	0,20	0,20	3,86	4,25
20	200	0,8	0,14	0,14	3,00	3,33
20	200	0,9	0,10	0,10	1,98	2,23
20	200	1,0	0,29	0,29	0,95	1,10
20	500	0,0	4,80	4,80	3,82	3,36
20	500	0,1	4,55	4,55	3,63	3,18
20	500	0,2	4,27	4,27	3,41	2,99
20	500	0,3	3,97	3,97	3,17	2,77
20	500	0,4	3,63	3,63	2,91	2,54
20	500	0,5	3,25	3,25	2,62	2,27
20	500	0,6	2,82	2,82	2,29	1,98
20	500	0,7	2,34	2,34	1,93	1,65
20	500	0,8	1,80	1,80	1,52	1,28
20	500	0,9	1,20	1,20	1,08	0,87
20	500	1,0	0,67	0,67	0,75	0,58
Promedio			2,856	2,856	5,348	6,949

ARPD de cada *tie breaking rule* implementada con la PR_C en función de las características de los problemas

Tabla 3 ARPD de cada *tie breaking rule* implementada con la PR_C en función de las características de los problemas.

m	n	w	$PR_C + TB_{NEH}$	$PR_C + TB_{FF}$	$PR_C + TB_{LJP}$	$PR_C + TB_C$
5	20	0,0	4,88	7,83	4,52	4,88
5	20	0,1	2,11	2,11	0,00	2,11
5	20	0,2	2,38	2,38	0,00	2,38
5	20	0,3	0,56	0,56	0,14	0,56
5	20	0,4	0,28	0,28	0,09	0,28
5	20	0,5	0,21	0,21	0,10	0,21
5	20	0,6	0,21	0,21	0,00	0,21
5	20	0,7	0,10	0,10	0,23	0,10
5	20	0,8	0,07	0,02	0,14	0,02
5	20	0,9	0,03	0,03	0,49	0,03
5	20	1,0	0,86	0,63	0,26	1,07
10	20	0,0	0,58	0,58	0,00	0,58
10	20	0,1	0,53	0,47	0,06	0,47
10	20	0,2	0,35	0,35	0,00	0,35
10	20	0,3	0,29	0,29	0,00	0,29
10	20	0,4	0,23	0,23	0,00	0,23
10	20	0,5	1,54	0,00	1,29	0,00
10	20	0,6	0,07	0,91	1,02	0,00
10	20	0,7	0,20	0,20	0,17	0,01
10	20	0,8	0,05	0,05	0,20	0,05
10	20	0,9	0,02	0,02	0,11	0,04
10	20	1,0	0,21	0,41	0,73	0,81
20	20	0,0	0,87	0,87	0,20	0,27
20	20	0,1	0,00	0,00	0,20	0,00
20	20	0,2	0,13	0,13	0,19	0,13
20	20	0,3	0,12	0,12	0,15	0,12
20	20	0,4	0,00	0,29	0,44	0,29
20	20	0,5	0,00	0,00	0,14	0,00
20	20	0,6	0,17	0,00	0,17	0,25
20	20	0,7	0,00	0,00	0,30	0,19
20	20	0,8	0,00	0,00	0,00	0,00
20	20	0,9	0,04	0,00	0,04	0,00
20	20	1,0	0,40	0,00	0,61	0,35
5	50	0,0	0,63	0,63	1,55	1,89
5	50	0,1	0,18	0,18	1,18	1,36
5	50	0,2	0,08	0,08	0,98	1,00
5	50	0,3	0,14	0,14	0,75	0,94
5	50	0,4	0,00	0,00	0,41	0,25
5	50	0,5	0,41	0,52	0,01	0,51
5	50	0,6	0,51	0,37	0,36	0,37

m	n	w	$PR_C + TB_{NEH}$	$PR_C + TB_{FF}$	$PR_C + TB_{LJP}$	$PR_C + TB_C$
5	50	0,7	0,24	0,16	0,13	0,24
5	50	0,8	0,11	0,14	0,26	0,15
5	50	0,9	0,25	0,17	0,13	0,25
5	50	1,0	0,16	0,17	0,27	0,20
10	50	0,0	3,61	3,61	3,45	3,28
10	50	0,1	3,71	3,71	1,16	3,36
10	50	0,2	1,14	1,14	2,14	0,87
10	50	0,3	1,82	1,82	1,26	1,34
10	50	0,4	1,63	1,28	1,56	0,95
10	50	0,5	0,59	1,06	0,70	0,78
10	50	0,6	0,55	0,55	1,25	0,70
10	50	0,7	0,87	0,87	0,11	0,87
10	50	0,8	0,34	0,39	0,60	0,39
10	50	0,9	0,42	0,48	0,24	0,34
10	50	1,0	1,12	0,05	0,84	0,18
20	50	0,0	0,57	0,57	0,11	0,57
20	50	0,1	0,48	0,48	0,10	0,48
20	50	0,2	0,45	0,45	0,03	0,45
20	50	0,3	0,18	0,18	0,16	0,18
20	50	0,4	0,43	0,43	0,37	0,43
20	50	0,5	0,20	0,20	0,31	0,00
20	50	0,6	0,17	0,17	0,36	0,17
20	50	0,7	0,54	0,75	0,39	0,98
20	50	0,8	0,60	0,60	0,02	0,53
20	50	0,9	0,17	0,17	0,24	0,17
20	50	1,0	0,84	0,22	0,87	0,45
5	100	0,0	0,60	0,60	2,95	1,14
5	100	0,1	0,11	0,11	0,60	0,11
5	100	0,2	0,03	0,03	0,37	0,03
5	100	0,3	0,19	0,19	0,04	0,19
5	100	0,4	0,07	0,07	0,21	0,06
5	100	0,5	0,10	0,10	0,04	0,13
5	100	0,6	0,20	0,17	0,04	0,17
5	100	0,7	0,15	0,15	0,41	0,15
5	100	0,8	0,22	0,31	0,27	0,22
5	100	0,9	0,01	0,01	0,13	0,01
5	100	1,0	0,28	0,08	0,26	0,09
10	100	0,0	3,62	3,62	1,00	2,09
10	100	0,1	2,51	2,51	1,42	1,81
10	100	0,2	1,08	0,81	0,82	0,74
10	100	0,3	0,57	0,57	0,60	0,92
10	100	0,4	0,77	0,66	0,31	0,66
10	100	0,5	0,46	0,03	0,95	0,03
10	100	0,6	0,48	0,48	0,30	0,52

m	n	w	$PR_C + TB_{NEH}$	$PR_C + TB_{FF}$	$PR_C + TB_{LJP}$	$PR_C + TB_C$
10	100	0,7	0,31	0,31	0,19	0,42
10	100	0,8	0,59	0,52	0,22	0,43
10	100	0,9	0,21	0,21	0,15	0,21
10	100	1,0	0,39	0,23	0,38	0,27
20	100	0,0	3,63	3,63	2,74	2,49
20	100	0,1	1,04	1,04	2,50	1,10
20	100	0,2	2,41	2,08	1,94	2,15
20	100	0,3	2,19	2,19	0,47	2,43
20	100	0,4	2,02	2,23	0,16	2,12
20	100	0,5	0,89	1,17	1,14	1,08
20	100	0,6	1,24	0,98	1,06	0,98
20	100	0,7	0,75	0,75	0,72	0,71
20	100	0,8	0,74	1,07	0,86	1,07
20	100	0,9	0,30	0,28	0,39	0,36
20	100	1,0	0,59	0,17	0,70	0,29
10	200	0,0	3,61	3,61	4,42	4,34
10	200	0,1	2,04	2,04	1,49	1,80
10	200	0,2	1,13	1,15	0,48	1,00
10	200	0,3	0,27	0,66	1,34	0,08
10	200	0,4	0,56	0,56	0,61	0,49
10	200	0,5	0,46	0,70	0,36	0,78
10	200	0,6	0,26	0,32	0,24	0,27
10	200	0,7	0,16	0,16	0,50	0,12
10	200	0,8	0,15	0,06	0,25	0,18
10	200	0,9	0,08	0,07	0,30	0,10
10	200	1,0	0,17	0,13	0,33	0,12
20	200	0,0	2,89	2,89	1,38	1,94
20	200	0,1	2,55	2,55	1,82	1,66
20	200	0,2	2,21	2,21	1,07	2,12
20	200	0,3	1,98	1,74	0,47	2,33
20	200	0,4	1,89	1,97	0,53	2,30
20	200	0,5	2,11	2,27	0,76	1,71
20	200	0,6	1,01	1,08	1,28	0,89
20	200	0,7	0,38	0,38	0,73	0,29
20	200	0,8	0,49	0,53	0,47	0,50
20	200	0,9	0,48	0,61	0,59	0,43
20	200	1,0	0,36	0,04	0,52	0,16
20	500	0,0	1,66	1,66	1,27	2,39
20	500	0,1	3,05	2,68	2,11	2,29
20	500	0,2	2,86	2,91	1,34	2,49
20	500	0,3	1,75	1,58	1,60	1,57
20	500	0,4	1,19	1,14	1,28	1,16
20	500	0,5	1,49	1,21	1,34	1,42
20	500	0,6	1,05	1,13	1,03	1,23
20	500	0,7	0,56	0,64	0,61	0,44
20	500	0,8	0,44	0,31	0,39	0,47
20	500	0,9	0,25	0,28	0,29	0,26
20	500	1,0	0,18	0,03	0,28	0,12
Promedio			0,83	0,82	0,68	0,78

ARPD del método NEH_{LJP} implementado con la PR_C en función de las características de los problemas

Tabla 4 ARPD del método NEH_{LJP} implementado con la PR_C en función de las características de los problemas.

m	n	w	NEH	NEH_{FF}	PR_c + TB_{LJP}	NEH_C
5	20	0,0	108,33	108,33	11,30	10,84
5	20	0,1	11,16	11,16	3,24	5,61
5	20	0,2	2,92	2,92	1,54	4,05
5	20	0,3	4,00	4,00	0,29	0,71
5	20	0,4	2,25	2,25	0,40	0,59
5	20	0,5	1,37	1,37	0,60	0,70
5	20	0,6	0,79	0,79	1,55	1,77
5	20	0,7	0,50	0,50	1,17	1,04
5	20	0,8	0,28	0,27	2,65	2,52
5	20	0,9	0,93	0,93	1,63	1,17
5	20	1,0	1,04	0,61	0,59	1,40
10	20	0,0	4,82	4,82	2,90	3,50
10	20	0,1	5,70	5,70	4,44	4,84
10	20	0,2	4,06	4,06	2,41	2,76
10	20	0,3	3,37	3,31	2,19	2,48
10	20	0,4	1,78	1,91	1,52	1,75
10	20	0,5	1,95	1,95	3,87	2,58
10	20	0,6	2,11	2,11	3,47	2,45
10	20	0,7	0,94	0,94	1,18	1,02
10	20	0,8	0,71	0,71	1,91	1,77
10	20	0,9	0,89	0,89	0,91	0,84
10	20	1,0	1,40	0,48	0,75	0,83
20	20	0,0	3,17	3,17	2,56	2,63
20	20	0,1	2,89	2,89	2,20	2,00
20	20	0,2	1,38	1,38	2,78	2,71
20	20	0,3	3,11	3,11	3,41	3,37
20	20	0,4	2,99	2,99	2,11	1,95
20	20	0,5	2,56	2,56	2,44	2,30
20	20	0,6	2,91	2,91	1,10	1,17
20	20	0,7	1,59	1,59	1,53	1,42
20	20	0,8	1,23	1,23	0,63	0,63
20	20	0,9	1,23	1,23	0,49	0,45
20	20	1,0	0,78	0,76	0,76	0,50
5	50	0,0	2,43	2,43	11,11	11,56
5	50	0,1	1,17	1,17	5,29	5,49
5	50	0,2	0,64	0,64	3,52	3,53
5	50	0,3	0,89	0,45	2,53	2,72
5	50	0,4	0,45	0,45	0,69	0,53
5	50	0,5	0,65	0,65	0,35	0,86
5	50	0,6	0,34	0,34	1,87	1,87
5	50	0,7	0,10	0,10	1,06	1,16
5	50	0,8	0,77	0,77	0,61	0,49
5	50	0,9	0,35	0,35	0,28	0,40
5	50	1,0	0,25	0,25	0,52	0,45

m	n	w	NEH	NEH_{FF}	$PR_c + TB_{LJP}$	NEH_C
10	50	0,0	12,06	12,06	10,22	9,66
10	50	0,1	6,05	6,05	6,26	8,33
10	50	0,2	4,86	4,86	6,67	5,32
10	50	0,3	2,72	2,72	3,62	3,77
10	50	0,4	3,62	3,62	3,68	3,06
10	50	0,5	1,82	1,82	2,25	2,32
10	50	0,6	2,09	2,09	2,83	2,27
10	50	0,7	1,64	1,38	1,12	1,88
10	50	0,8	0,65	0,69	0,96	0,75
10	50	0,9	0,81	0,81	0,84	0,96
10	50	1,0	0,92	0,77	0,96	0,30
20	50	0,0	7,81	7,81	1,04	1,49
20	50	0,1	5,21	5,21	0,83	1,20
20	50	0,2	2,73	2,73	2,06	2,49
20	50	0,3	1,79	1,79	2,02	2,04
20	50	0,4	2,65	2,65	1,48	1,53
20	50	0,5	1,53	1,53	1,39	1,07
20	50	0,6	1,11	1,11	1,35	1,17
20	50	0,7	2,37	2,37	0,52	1,11
20	50	0,8	2,11	2,11	1,03	1,54
20	50	0,9	0,62	0,69	0,58	0,50
20	50	1,0	0,99	0,56	1,08	0,66
5	100	0,0	5,55	5,55	6,75	4,97
5	100	0,1	1,28	1,28	1,78	1,29
5	100	0,2	1,00	1,00	0,59	0,25
5	100	0,3	0,51	0,51	0,71	0,87
5	100	0,4	0,35	0,35	0,56	0,42
5	100	0,5	0,33	0,33	0,21	0,30
5	100	0,6	0,84	0,81	0,08	0,20
5	100	0,7	0,38	0,38	0,54	0,29
5	100	0,8	0,04	0,04	0,35	0,29
5	100	0,9	0,08	0,06	0,22	0,09
5	100	1,0	0,19	0,10	0,32	0,14
10	100	0,0	8,23	8,23	1,91	3,00
10	100	0,1	5,36	5,36	1,88	2,28
10	100	0,2	3,54	3,54	1,68	1,61
10	100	0,3	1,98	1,98	1,06	1,38
10	100	0,4	1,67	1,67	0,76	1,11
10	100	0,5	0,99	1,12	1,31	0,39
10	100	0,6	0,71	0,78	0,53	0,74
10	100	0,7	0,68	0,68	0,51	0,74
10	100	0,8	0,55	0,57	0,51	0,72
10	100	0,9	0,36	0,36	0,31	0,37
10	100	1,0	0,66	0,26	0,54	0,44
20	100	0,0	2,80	2,80	3,19	2,94
20	100	0,1	2,16	2,16	3,07	1,66
20	100	0,2	1,89	1,89	3,14	3,39
20	100	0,3	1,83	1,83	3,71	5,70
20	100	0,4	1,65	1,65	1,08	3,04
20	100	0,5	1,46	1,47	2,80	2,75
20	100	0,6	0,88	0,88	2,75	2,66

m	n	w	NEH	NEH_{FF}	$PR_c + TB_{LJP}$	NEH_C
20	100	0,7	1,42	1,42	1,37	1,36
20	100	0,8	1,50	1,35	0,86	1,06
20	100	0,9	0,69	0,66	0,56	0,53
20	100	1,0	0,96	0,44	0,74	0,33
10	200	0,0	6,13	6,13	5,37	5,39
10	200	0,1	2,05	2,25	4,73	5,00
10	200	0,2	2,33	2,33	2,52	3,03
10	200	0,3	1,73	1,73	2,55	1,27
10	200	0,4	1,26	1,26	1,31	1,19
10	200	0,5	0,38	0,74	0,94	1,36
10	200	0,6	0,93	0,89	0,42	0,46
10	200	0,7	0,51	0,51	0,64	0,26
10	200	0,8	0,67	0,67	0,25	0,19
10	200	0,9	0,21	0,20	0,41	0,21
10	200	1,0	0,35	0,25	0,44	0,23
20	200	0,0	5,89	5,89	1,34	1,89
20	200	0,1	5,43	5,43	1,94	1,78
20	200	0,2	3,89	3,66	1,38	2,43
20	200	0,3	1,41	1,41	0,77	2,65
20	200	0,4	2,75	3,08	1,36	3,14
20	200	0,5	2,61	2,01	1,17	2,11
20	200	0,6	1,67	1,86	1,38	0,99
20	200	0,7	0,83	0,83	0,93	0,48
20	200	0,8	1,12	0,88	0,53	0,55
20	200	0,9	0,84	0,89	0,82	0,65
20	200	1,0	0,38	0,20	0,49	0,12
20	500	0,0	5,41	5,41	1,46	2,60
20	500	0,1	4,52	4,52	1,57	1,75
20	500	0,2	2,84	2,88	1,35	2,50
20	500	0,3	2,60	1,89	1,86	1,83
20	500	0,4	1,67	1,69	1,47	1,34
20	500	0,5	2,65	2,60	1,41	1,50
20	500	0,6	1,36	1,54	0,76	0,96
20	500	0,7	0,83	0,82	0,60	0,43
20	500	0,8	0,53	0,51	0,36	0,44
20	500	0,9	0,42	0,25	0,31	0,28
20	500	1,0	0,17	0,23	0,29	0,13
Promedio			2,87	2,83	1,82	1,92

ARPD de cada *priority rule* implementada con la TB_C en función de las características de los problemas

Tabla 5 ARPD de cada *priority rule* implementada con la TB_C en función de las características de los problemas.

m	n	w	PR_{NEH} + TB_C	PR_{FF} + TB_C	PR_{LJP} + TB_C	PR_C + TB_C
5	20	0,0	72,12	72,12	129,02	7,21
5	20	0,1	8,60	8,60	18,52	5,01
5	20	0,2	1,61	1,61	9,24	3,46
5	20	0,3	3,24	3,24	5,75	0,42
5	20	0,4	2,17	2,17	2,49	0,81
5	20	0,5	1,62	1,62	1,93	0,96
5	20	0,6	1,04	1,04	1,18	2,02
5	20	0,7	0,51	0,51	1,35	1,05
5	20	0,8	0,56	0,56	1,74	2,80
5	20	0,9	1,27	1,27	1,14	1,50
5	20	1,0	0,60	0,60	1,69	1,35
10	20	0,0	7,94	7,94	8,56	6,61
10	20	0,1	7,45	7,45	9,55	6,59
10	20	0,2	5,48	5,48	6,32	4,20
10	20	0,3	3,94	3,94	4,51	3,16
10	20	0,4	3,21	3,21	4,26	2,97
10	20	0,5	2,45	2,45	3,53	3,02
10	20	0,6	2,20	2,20	3,29	2,54
10	20	0,7	1,00	1,00	1,96	1,08
10	20	0,8	0,91	0,91	2,49	1,96
10	20	0,9	1,08	1,08	1,27	1,03
10	20	1,0	1,17	1,17	0,44	1,11
20	20	0,0	5,11	5,11	2,78	4,50
20	20	0,1	4,93	4,93	2,03	3,97
20	20	0,2	3,99	3,99	1,85	5,28
20	20	0,3	3,84	3,84	3,88	4,12
20	20	0,4	3,84	3,84	2,58	2,83
20	20	0,5	3,65	3,65	2,02	3,36
20	20	0,6	3,50	3,50	0,59	1,75
20	20	0,7	1,94	1,94	0,82	1,75
20	20	0,8	1,51	1,51	0,78	0,90
20	20	0,9	1,74	1,74	0,39	0,90
20	20	1,0	0,88	0,88	0,27	0,63
5	50	0,0	3,70	3,70	8,85	11,71
5	50	0,1	1,20	1,20	4,26	5,53
5	50	0,2	0,98	0,98	2,93	3,90
5	50	0,3	0,78	0,78	2,05	3,07
5	50	0,4	0,80	0,80	0,82	0,87
5	50	0,5	0,62	0,62	0,48	0,83
5	50	0,6	0,50	0,50	1,24	2,03

m	n	w	PR_{NEH} + TB_C	PR_{FF} + TB_C	PR_{LJP} + TB_C	PR_C + TB_C
5	50	0,7	0,40	0,40	0,44	1,47
5	50	0,8	0,82	0,82	0,47	0,55
5	50	0,9	0,58	0,58	0,25	0,30
5	50	1,0	0,24	0,24	0,40	0,47
10	50	0,0	13,01	13,01	5,09	10,43
10	50	0,1	7,95	7,95	4,06	10,20
10	50	0,2	6,17	6,17	2,95	7,71
10	50	0,3	3,84	3,84	3,22	4,85
10	50	0,4	3,96	3,96	2,50	3,57
10	50	0,5	3,01	3,01	1,74	3,67
10	50	0,6	2,53	2,53	2,12	2,49
10	50	0,7	2,03	2,03	0,60	2,48
10	50	0,8	0,84	0,84	1,21	0,90
10	50	0,9	0,77	0,77	1,48	0,92
10	50	1,0	0,91	0,91	0,34	0,55
20	50	0,0	7,62	7,62	3,10	1,32
20	50	0,1	5,45	5,45	3,50	1,43
20	50	0,2	3,87	3,87	3,48	3,60
20	50	0,3	3,31	3,31	2,42	3,56
20	50	0,4	4,24	4,24	2,15	3,06
20	50	0,5	2,66	2,66	1,74	2,13
20	50	0,6	1,59	1,59	0,74	1,64
20	50	0,7	2,50	2,50	1,73	1,24
20	50	0,8	2,09	2,09	0,41	1,52
20	50	0,9	1,05	1,05	0,65	0,86
20	50	1,0	0,40	0,40	0,76	0,60
5	100	0,0	21,13	21,13	26,59	5,74
5	100	0,1	1,80	1,80	1,36	1,79
5	100	0,2	1,14	1,14	1,22	0,38
5	100	0,3	0,55	0,55	0,68	0,90
5	100	0,4	0,36	0,36	0,64	0,43
5	100	0,5	0,41	0,41	0,42	0,37
5	100	0,6	0,71	0,71	0,61	0,11
5	100	0,7	0,32	0,32	0,68	0,23
5	100	0,8	0,31	0,31	0,24	0,42
5	100	0,9	0,10	0,10	0,13	0,13
5	100	1,0	0,08	0,08	0,09	0,16
10	100	0,0	6,82	6,82	2,92	3,06
10	100	0,1	4,00	4,00	2,03	2,27
10	100	0,2	2,45	2,45	1,95	2,10
10	100	0,3	2,62	2,62	1,16	1,63
10	100	0,4	1,97	1,97	0,74	1,40
10	100	0,5	1,17	1,17	1,64	0,55
10	100	0,6	0,80	0,80	0,69	0,74

m	n	w	PR_{NEH} + TB_C	PR_{FF} + TB_C	PR_{LJP} + TB_C	PR_C + TB_C
10	100	0,7	0,71	0,71	0,75	0,77
10	100	0,8	0,70	0,70	0,47	0,85
10	100	0,9	0,39	0,39	0,51	0,68
10	100	1,0	0,39	0,39	0,29	0,43
20	100	0,0	2,85	2,85	4,31	3,21
20	100	0,1	3,26	3,26	4,63	3,27
20	100	0,2	1,80	1,80	3,96	3,49
20	100	0,3	1,40	1,40	2,08	5,26
20	100	0,4	1,62	1,62	1,83	3,39
20	100	0,5	1,83	1,83	4,07	3,09
20	100	0,6	0,91	0,91	2,09	2,72
20	100	0,7	1,53	1,53	1,70	1,59
20	100	0,8	1,48	1,48	1,14	1,19
20	100	0,9	0,85	0,85	0,81	0,72
20	100	1,0	0,59	0,59	0,69	0,33
10	200	0,0	6,17	6,17	8,06	4,96
10	200	0,1	3,58	3,58	1,25	5,39
10	200	0,2	2,11	2,11	1,07	2,82
10	200	0,3	2,43	2,43	0,92	2,05
10	200	0,4	0,99	0,99	1,09	1,46
10	200	0,5	0,68	0,68	0,49	1,45
10	200	0,6	1,16	1,16	0,24	0,83
10	200	0,7	0,64	0,64	0,46	0,38
10	200	0,8	0,66	0,66	0,50	0,18
10	200	0,9	0,35	0,35	0,32	0,27
10	200	1,0	0,20	0,20	0,18	0,20
20	200	0,0	6,38	6,38	1,72	1,38
20	200	0,1	3,50	3,50	3,06	0,75
20	200	0,2	2,33	2,33	1,37	0,97
20	200	0,3	2,66	2,66	0,83	3,35
20	200	0,4	3,19	3,19	0,88	2,87
20	200	0,5	1,96	1,96	1,00	2,00
20	200	0,6	1,91	1,91	1,15	0,89
20	200	0,7	1,15	1,15	1,02	0,55
20	200	0,8	1,00	1,00	1,04	1,00
20	200	0,9	0,73	0,73	0,91	0,49
20	200	1,0	0,34	0,34	0,19	0,22
20	500	0,0	4,11	4,11	2,55	1,78
20	500	0,1	5,03	5,03	0,47	2,26
20	500	0,2	3,54	3,54	2,65	2,38
20	500	0,3	2,32	2,32	0,99	1,86
20	500	0,4	1,58	1,58	1,23	1,52
20	500	0,5	2,03	2,03	0,82	1,16
20	500	0,6	1,79	1,79	1,00	1,25
20	500	0,7	0,99	0,99	0,57	0,56
20	500	0,8	0,68	0,68	0,22	0,50
20	500	0,9	0,18	0,18	0,18	0,19
20	500	1,0	0,12	0,12	0,12	0,12
Promedio			2,95	2,95	3,17	2,21

Bibliografía

[1] Weibo Liu, Yan Jin & Mark Price, *A new Nawaz–Enscore–Ham-based heuristic for permutation flow-shop problems with bicriteria of makespan and machine idle time*, Engineering Optimization, 2016.

[2] Víctor Fernández-Viagas Escudero, José Manuel Framiñán Torres, *The permutation flowshop scheduling problems análisis, solution procedures and problem extensions*, Departamento de Organización Industrial y Gestión de Empresas, Universidad de Sevilla, 2016.

[3] Fernández-Viagas, V. & J. M. Framiñán, *On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem*, 2014.

[4] Paz Pérez González, José Manuel Framiñán Torres, Víctor Fernández-Viagas Escudero, *Programación de operaciones: 4º GITI*, Departamento de Organización Industrial y Gestión de Empresas, Universidad de Sevilla, 2019.

[5] Paz Pérez González, *Sistemas de producción integrados*, Departamento de Organización Industrial y Gestión de Empresas, Universidad de Sevilla, 2008.

[6] Análisis de la complejidad de los algoritmos, *https://www.cs.us.es/~jalonso/cursos/i1m-19/temas/tema-28.html#%C3%B3rdenes-de-complejidad*, 2020.

[7] Teoría de la complejidad algorítmica, *https://www.monografias.com/trabajos27/complejidad-algoritmica/complejidad-algoritmica.shtml*, 2020.

[8] Michael L. Pinedo, *Scheduling Theory, Algorithms, and Systems*, Springer, 2008.

[9] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, *Optimization and aproximation in deterministic sequencing and scheduling*, 1979.

[10] Uday K. Chakraborty (Ed.), *Computational Intelligence in Flow Shop and Job Shop Scheduling*, Springer, 2009.

[11] Kathrin Maassen, Paz Pérez Gonzalez, Lisa C. Günther, *Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling*, Computers and Operations Research, 2020.

[12] E. Taillard, *Benchmarks for Basic Scheduling Problems*, Eur. J. Oper. Res., vol. 64, no. 2, pp. 278–285, *http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html*, 1993.