Trabajo Fin de Máster

Máster Organización Industrial y Gestión de Empresas

Algoritmos aproximados para minimizar el tiempo total de finalización en talleres de flujo sin esperas

Autor: Fernando Guerrero Ortega

Tutor: Víctor Fernández-Viagas Escudero

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

Sevilla, 2019

Trabajo Fin de Máster
Máster Organización Industrial y Gestión de Empresas

Algoritmos aproximados para minimizar el tiempo total de finalización en talleres de flujo sin esperas

Autor:

Fernando Guerrero Ortega

Tutor:

Víctor Fernández-Viagas Escudero

Profesor Ayudante Doctor

Dpto. Organización Industrial y Gestión de Empresas

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Máster: Algoritmos aproximados para minimizar el tiempo total de finalización en talleres de flujo sin esperas

Autor: Fernando Guerrero Ortega

Tutor: Víctor Fernández-Viagas Escudero

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

Agradecimientos

A todo el personal docente, en todos los niveles, que me ha enseñado a aprender y no sólo a estudiar.

A los profesores que desde el primer momento consiguieron que encontrara todo el interés en una especialidad respecto a la que era bastante escéptico cuando la escogí.

Al tutor de este trabajo, D. Víctor Fernández-Viagas Escudero, quien además de orientarme de forma ejemplar de nuevo, se adaptó a mis cambiantes circunstancias durante todo el tiempo que duró su elaboración.

A ARUS, por permitirme vivir una experiencia irrepetible en un contexto que todos sus miembros sabemos que será imposible de encontrar en nuestras vidas profesionales.

A todos los que mantienen sus principios y siguen escribiendo “sólo” con tilde.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

La programación de operaciones consiste en la toma de decisiones en una industria para distribuir los recursos y tareas en el tiempo, con el objetivo de lograr el mejor resultado posible atendiendo a criterios previamente establecidos. Es crucial para el resultado de una empresa, ya que influye en los plazos y costes de los productos, que a largo plazo determinan el nivel de servicio de la compañía, así como su competitividad en el mercado (Framinan, Leister, Ruiz; 2014).

En este proyecto se considera el caso de una fábrica donde los recursos que deben gestionarse a lo largo del horizonte temporal son las máquinas de la misma, y las tareas consisten en los diferentes productos que en ella se fabrican.

En una planta de fabricación pueden tenerse distintos entornos de trabajo, siendo el más sencillo de ellos el formado por una única máquina. Dentro de las situaciones donde se tienen más máquinas, principalmente se pueden tener los siguientes entornos:

- El primero de ellos es el caso donde las máquinas funcionan en paralelo, y por tanto no es necesario que los trabajos pasen por todas ellas.
- El segundo consiste en los entornos de fabricación de tipo taller, donde los trabajos deben pasar por todas y cada una de las máquinas para considerarse terminados. Se consideran tres subentornos diferentes atendiendo al orden de procesado de los trabajos en cada máquina: *flow shop, job shop, open shop.*

Por último, en plantas de producción de un tamaño considerable, se puede dar el caso de entornos de producción híbridos, donde se tengan entornos diferentes en diversas secciones de las mismas.

El entorno considerado en este proyecto es un *flow shop*, o taller de flujo regular, en el que todos los n trabajos deben seguir la misma ruta de m máquinas, dispuestas en serie, como se muestra en la Figura 1.1. Las industrias con este tipo de entorno suelen fabricar productos que requieran de diversas etapas en su tratamiento o de un ensamblado de sus componentes. En esta clase de situaciones, una etapa concreta de un trabajo dado puede comenzar tan pronto como la anterior haya finalizado, salvo que la máquina en la que se lleve a cabo ya esté procesando otro trabajo, en cuyo caso hay un tiempo de espera hasta que quede libre.

Como explican Ruiz y Stützle (2007), una simplificación habitual de este problema consiste en forzar que el orden de procesado de trabajos sea idéntico en todas las máquinas que componen el taller. El problema resultante se denomina *permutation flow shop problem* (PFSP) y tiene $n!$ posibles soluciones, tantas como secuencias de trabajos distintas se pueden seguir.

A partir del momento en el que el trabajo en la posición j ha terminado de procesarse en la máquina i, dicho trabajo puede comenzar a tratarse en la máquina $i+1$, y la máquina i puede empezar a procesar el trabajo en la posición $j+1$.

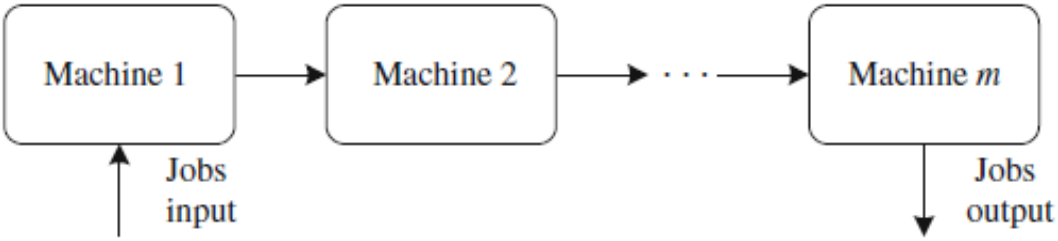

Figura 1.1. Flujo de trabajos en un entorno *flow shop* (Framinan, Leister, Ruiz; 2014).

Para evaluar y comparar las distintas secuencias hay que conocer los tiempos de finalización de los trabajos. Para ello, se parte del tiempo de procesado del primer trabajo en la primera máquina, y a partir del mismo se obtienen los tiempos de inicio del procesado del segundo trabajo en la primera máquina y del primer trabajo en la segunda máquina. Así, siguiendo el esquema mostrado en la Figura 1.2, se van conociendo todos, hasta llegar al tiempo de finalización del procesado del trabajo n en la máquina m, momento en el que se completaría la producción.

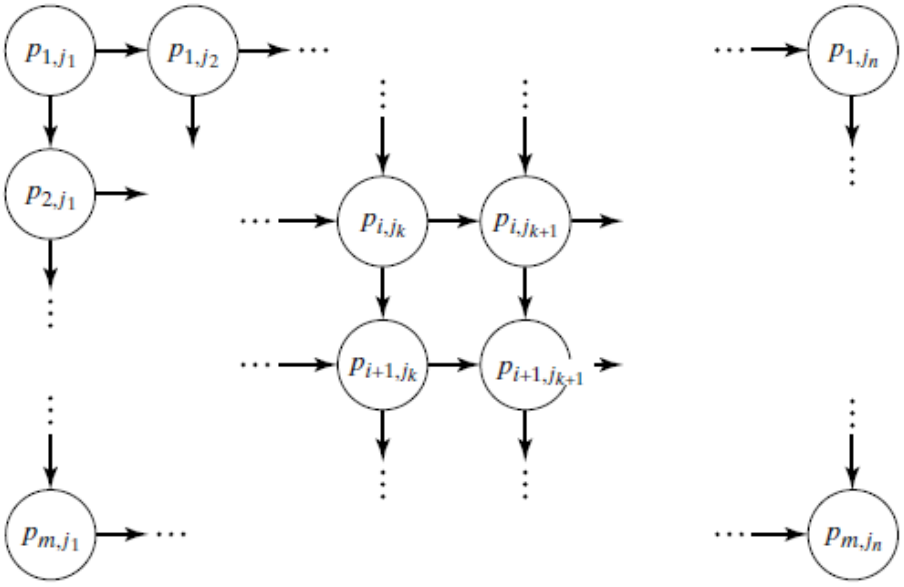

Figura 1.2. Cálculo de los tiempos de finalización de trabajos en un taller de flujo regular (Pinedo; 2012).

Normalmente, a los trabajos se les permite esperar indefinidamente entre etapas si las máquinas posteriores se encuentran ocupadas, pero hay ciertos escenarios donde eso no es admisible, como aquél en el que la capacidad de almacenamiento de trabajos esperando a ser tratados por una máquina sea limitada, o en el que los trabajos deban estar continuamente siendo tratados sin que haya interrupción alguna durante su procesado.

Dos ejemplos de estas situaciones son propuestos por Framinan, Leister y Ruiz (2014), y consisten en el tratamiento del acero, donde éste no debe enfriarse entre cada una de las etapas del proceso de fabricación, o el procesado de alimentos, donde no sólo se debe evitar que se rompa la cadena del frío en caso de tratarse de congelados, sino que también hay que prevenir su posible contaminación.

En este tipo de contextos, denominados *no-wait*, como todos los trabajos se procesan en el mismo orden en cada una de las máquinas y no deben esperar entre dos operaciones consecutivas, el inicio del procesado de un trabajo en la primera máquina podría demorarse con el objeto de que no hubiese interrupciones en su procesado más adelante. La Figura 1.3 muestra un caso con tres máquinas y cuatro trabajos (cuyos tiempos de proceso se adjuntan en la Tabla 1.1) donde no hay ninguna restricción acerca de los tiempos de espera, mientras que en la Figura 1.4 se tiene el mismo caso, sólo que esta vez sí se aplica la restricción *no-wait*, y por ello el trabajo 1 ve cómo sus dos primeras tareas empiezan más tarde de lo que podrían hacerlo, para así asegurarse de que podrá tratarse en la tercera máquina inmediatamente después de finalizar su procesado en la segunda.

p_{ij}		*trabajo (j)*			
		1	***2***	***3***	***4***
máquina (i)	***1***	2	8	5	1
	2	5	9	9	3
	3	6	6	2	8

Tabla 1.1. Tiempos de proceso para un problema genérico con cuatro trabajos y tres máquinas.

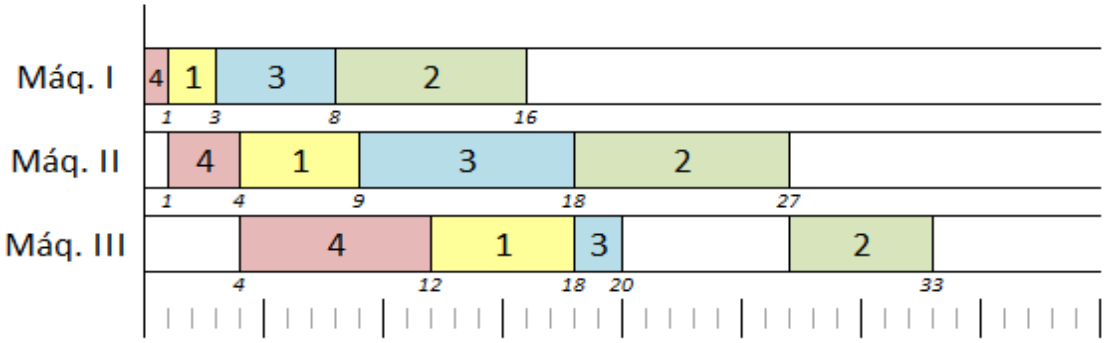

Figura 1.3. Diagrama de Gantt resultado de la secuencia π=[4, 1, 3, 2] para un PFSP con los tiempos de proceso de la Tabla 1.1.

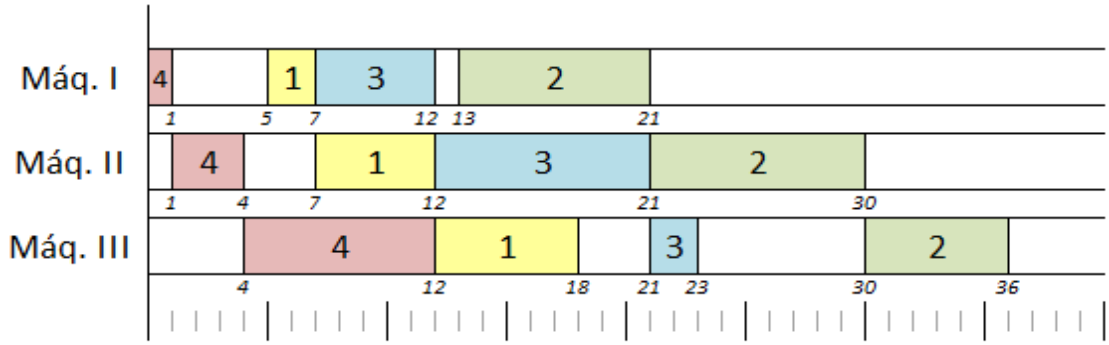

Figura 1.4. Diagrama de Gantt resultado de la secuencia π=[4, 1, 3, 2] para un PFSP con restricción *no-wait* y los tiempos de proceso de la Tabla 1.1.

Por otra parte, se pueden dar casos donde cierta espera no sólo sea admisible, sino necesaria. Por ejemplo, si tras pintar un producto hubiese que esperar a que la pintura se secase para continuar trabajando sobre él se tendría un tiempo mínimo de espera entre etapas (Framinan, Leister, Ruiz; 2014). Aunque estas situaciones no se consideran en este proyecto.

En los siguientes apartados, se detalla en profundidad el problema a tratar en este proyecto, el procedimiento para obtener los resultados de interés, y las conclusiones que se pueden obtener a partir de los mismos.

- En el capítulo 2 se describe más detalladamente el problema a resolver, definiendo las distintas restricciones que se aplican al mismo, así como el modelo matemático que lo caracteriza.
- El capítulo 3 se aborda el estado del arte del problema, donde se revisan las distintas publicaciones que estudian el problema tratado en este documento u otros similares.
- En el capítulo 4 se explica el procedimiento para resolver el problema con el que se obtienen mejores resultados de entre todos los mencionados en el estado del arte.
- En los capítulos del 5 al 7 se presentan las nuevas técnicas y algoritmos que se proponen para la resolución del problema.
- En el capítulo 8 se analizan todos los algoritmos propuestos, comparándose según diversos indicadores.
- Finalmente, se incluyen las conclusiones del trabajo y la bibliografía consultada para la realización del mismo, así como anexos complementarios a temas tratados en otros puntos del documento.

2. DESCRIPCIÓN DEL PROBLEMA

Como se ha dicho anteriormente, el *permutation flow shop problema* (PFSP) es objeto de análisis en gran cantidad de estudios debido a que es un entorno de producción muy común cuando se fabrica en serie. En muchos de ellos además se considera la restricción *no-wait*, que impide a las unidades producidas esperar entre las distintas etapas del proceso de fabricación.

Si bien el uso de talleres de tipo *flow shop* es algo bastante común en procesos de fabricación, la gran mayoría de estos problemas son *NP-hard* (Garey, Johnson, Sethi; 1976), llegando a ser muy complicado alcanzar soluciones cercanas al óptimo para casos con un número de trabajos y/o máquinas lo suficientemente altos, por lo que se usan heurísticas con el fin de intentar llegar a una solución no muy alejada de la óptima de manera relativamente sencilla.

El objetivo que se intenta minimizar en el problema tratado en este proyecto es el tiempo total de finalización, que es la suma del tiempo que permanece en el taller cada una de las unidades que en él se fabrican. Este valor también es comúnmente denominado tiempo total de flujo, pero dichos conceptos sólo representan lo mismo cuando la disponibilidad de los trabajos es completa desde el instante inicial, es decir, cuando éstos no tienen fechas de llegada, como ocurre en el problema estudiado en este proyecto. Minimizar dicho valor es equivalente a reducir el tiempo medio que pasan los trabajos en las distintas etapas del proceso de fabricación (Shyu, Lin, Yin; 2004), por lo que un buen resultado significaría que los clientes tendrían antes sus productos terminados, algo que no ocurre necesariamente cuando el objetivo es el *makespan*, el tiempo de finalización del último trabajo, que es el que más habitualmente se estudia en esta clase de problemas.

Adicionalmente, se consideran las restricciones comunes del entorno *flow shop*, es decir, que una máquina no puede procesar más de un trabajo simultáneamente, y que un trabajo no puede estar asignado a más de una máquina al mismo tiempo.

El propósito de este proyecto es encontrar nuevos métodos de resolución del problema presentado que puedan aportar mejores resultados que los obtenidos con los existentes en la actualidad.

Para describir el modelo del problema, usaremos la notación introducida por Graham et al. (1979), que consiste en describir los tres conceptos previamente mencionados en tres campos, $\alpha|\beta|\gamma$, representando cada uno de ellos:

- α: el entorno del proceso de fabricación, es decir, la disposición de las máquinas y cuáles se deben usar para producir los trabajos.
- β: las distintas restricciones que se aplican al problema.
- γ: el objetivo que determina qué soluciones son más beneficiosas.

Usando dicha notación, el problema a estudiar en este proyecto se expresa como $F_m|prmu, nwt|\sum C_j$. Para expresar las ecuaciones que modelan el problema, se toma como referencia el artículo publicado por Stafford Jr, Tseng y Gupta (2005), en el que se recogen varios modelos para problemas de taller de flujo, siendo uno de ellos el del problema $F_m|prmu|C_{max}$. Basándose en ese modelo, basta con cambiar el objetivo a minimizar y eliminar la holgura de una de las ecuaciones para impedir que haya esperas entre máquinas, y se obtiene el que caracteriza al problema a analizar.

A continuación, se plantean y describen los datos y variables que en él aparecen en la Figura 2.1, en la que se representa el modelo del problema, y se explica el papel de cada una de las restricciones que lo forman, así como el cálculo de la función objetivo.

- p_{ij} : Tiempo que tarda en procesarse el trabajo *j* en la máquina *i*
- C_{ij} : Tiempo de finalización del trabajo *j* en la máquina *i*
- X_{jk} : Variable binaria que vale 1 si el trabajo *j* se procesa antes que el *k* (no necesariamente inmediatamente antes)
- A : Término lo suficientemente grande como para considerar despreciables el resto de sumandos de su lado de la inecuación
- M : Conjunto de todas las máquinas que componen el taller
- N : Conjunto de todos los trabajos que se deben procesar
- m : Número de máquinas que componen el taller
- n : Número de trabajos que deben ser procesados

$$MIN \sum_{j=1}^{n} C_{mj} \quad (0)$$

s. a:

$$C_{1j} \geq p_{1j} \quad \forall j \in N \quad (1)$$

$$C_{ij} - C_{ik} + A \cdot X_{jk} \geq p_{ij} \quad \forall i \in M,\ \forall j \in N,\ \forall k \in N,\ j < k \quad (2)$$

$$C_{ij} - C_{ik} + A\,(X_{jk} - 1) \leq -p_{ik} \quad \forall i \in M,\ \forall j \in N,\ \forall k \in N,\ j < k \quad (3)$$

$$C_{ij} - C_{i-1,j} = p_{ij} \quad \forall i \in M - \{1\},\ \forall j \in N \quad (4)$$

Figura 2.1. Modelo del problema Fm|prmu, nwt|∑Cj.

La función objetivo (0) hace que no sólo se intente que último trabajo se termine lo antes posible, sino que se minimice la suma de los instantes de finalización de cada uno de ellos, que obviamente coincide con el tiempo en el que salen de la última máquina de la línea. Este valor se denomina Tiempo Total de Finalización, más conocido por sus siglas en inglés, TCT (*Total Completion Time*).

La ecuación (1) establece el límite inferior del tiempo de finalización de cada uno de los trabajos en la primera máquina, que debe ser al menos igual al tiempo de proceso del trabajo correspondiente en la primera máquina, es decir, evita que haya trabajos que comiencen a procesarse antes del instante inicial.

Las ecuaciones (2) y (3) son muy similares a la (1), pero para las máquinas a partir de la segunda, por lo que en este caso el "instante inicial" viene marcado por el instante de finalización en la máquina en cuestión del trabajo que se procese con anterioridad.

La ecuación (4) es la que asegura que la restricción *no-wait* se cumpla, es decir, que cada uno de los trabajos sea procesado en cada una de las máquinas sin interrupción alguna.

3. ESTADO DEL ARTE

Se ha llevado a cabo una revisión de artículos publicados sobre entornos de producción en talleres de flujo regular (*flow shop*) en los que se imponen restricciones sobre las esperas de los trabajos entre máquinas, y se han analizado los artículos encontrados que tratan tanto el problema que estudia este trabajo como otros con restricciones u objetivos similares con el propósito de observar y analizar los métodos ya propuestos para resolver dichos problemas.

3.1. Problema Fm|prmu, nwt|∑Cj

Como se comentó en el capítulo anterior, minimizar el tiempo total de finalización no es el objetivo más habitual en este tipo de situaciones. Sin embargo, han sido numerosos los autores que han estudiado este problema.

Los algoritmos aproximados para resolver el problema del que trata este trabajo se pueden dividir en heurísticas constructivas y metaheurísticas (Lin, Ying; 2016).

Dentro de las heurísticas constructivas, Rajendran y Chaudhuri (1990) propusieron dos heurísticas basadas en la inserción de trabajos, que eran más efectivas que otras existentes en su momento. Aldowaisan y Allahverdi (2004) presentaron seis heurísticas usando diferentes métodos de búsqueda, entre las cuales destacaba la PH1(p), que mejoraba la propuesta por Rajendran y Chaudhuri (1990), o el algoritmo genético propuesto por Chen, Neppaili y Alkaber (1996). Framinan, Nagano y Moccellin (2010) propusieron la heurística constructiva FNM, con la que se obtenían mejores resultados que con las anteriores.

Basándose en el procedimiento de Laha y Chakraborty (2009), Gao, Pan, Suganthan y Li (2013) propusieron dos heurísticas constructivas: una de ellas desarrollada a partir de la heurística de desviación típica de Gao, Pan, Li (2011), y otra a partir de la heurística de Bertolissi (2000). Ésta última demostró aportar mejores resultados que la heurística de inserción de trabajos NEH (Nawaz et al., 1983).

Más recientemente, se han propuesto artículos que presentan procedimientos para resolver el problema $F_m|nwt|\sum C_j$, como un algoritmo híbrido de optimización por enjambre de partículas (Akhshabi, Tavakkoli-Modhaddam, Rahnamay-Roodposhti; 2014), o el de Laha y Sapkal (2014), quienes presentaron una heurística LS con la que se obtienen mejores resultados que con las heurísticas PH1(p) y FNM.

Por último, la heurística propuesta por Ye, Li, Abedini, Nault (2017), denominada CFI, aporta mejores resultados que con las heurísticas PH1(p), FNM y LS, empleando además un menor tiempo computacional. Esta heurística consiste en una serie de mejoras sobre una secuencia inicial generada mediante un algoritmo cuyo uso está bastante extendido, como se observa en los artículos publicados por Liu y Reeves (2001), Fernandez-Viagas y Framinan (2015), Fernandez-Viagas, Dios y Framinan (2016) o Framinan y Perez-Gonzalez (2017).

En lo referente a las metaheurísticas, hay muchos estudios que han tratado este problema, como los que presentaron Akhshabi, Tavakkoli-Modhaddam y Rahnamay-Roodposhti (2014) o Zhu y Li (2015). Qi et al. (2016) propusieron un algoritmo de búsqueda local rápida, y los resultados experimentales demuestran que dicho algoritmo es más efectivo que otros propuestos previamente, como el IHA o el DHS. Sin embargo, el tiempo computacional requerido para obtener una solución próxima a la óptima es considerablemente mayor que con las heurísticas constructivas. Esto hace que no sea habitual usar metaheurísticas para la programación de operaciones.

3.2. Problemas similares

En cuanto a estudios sobre problemas similares, la minimización del *makespan* en el problema *flow shop no-wait* ha sido ampliamente estudiada. Algunas metaheurísticas usualmente empleadas para su resolución fueron analizadas por Bewoor, Prakash y Sapkal (2017), como la búsqueda tabú, el algoritmo genético o la optimización por enjambre de partículas. Lin y Ying (2016) presentan otra metaheurística, consistente en tres fases: una heurística constructiva MNEH para lograr una secuencia de trabajos inicial, un algoritmo LHK que mejora dicha secuencia, y un modelo matemático BIP que hace que ésta se aproxime a la óptima. Asimismo, se han propuesto numerosos algoritmos y heurísticas para resolver el problema con el *makespan* como objetivo, como el *Discrete Water Wave* (Zhao, Liu, Zhang, Ma, Zhang; 2018) o el ACO-LS (Riyanto, Santosa; 2015).

Se han estudiado otros objetivos para este tipo de problemas además del *makespan* y el TCT, como el que tratan Ruiz y Allahverdi (2009), que consiste en una combinación del propio *makespan* con el retraso del trabajo que se termina de procesar más tarde respecto a su fecha de entrega.

En otros estudios se ha tratado el mismo problema pero con unas restricciones distintas, como permitir un tiempo máximo de espera entre máquinas (Ye, Zhao, Li, Lei; 2017) o considerar fechas de entrega para los trabajos (Samarghandi, Behroozi; 2017).

4. HEURÍSTICA REFERENCIA

En esta sección, se detalla la heurística con la que se obtienen mejores resultados de entre todas las analizadas en el capítulo anterior, que es la propuesta por Ye, Li, Abedini y Nault (2017), denominada CFI. Básicamente, esta heurística consiste en una primera parte donde se genera una secuencia inicial y una segunda en la que se aplican técnicas de inserción e intercambio de trabajos con el fin de mejorar dicha secuencia. Ambas etapas se detallan a continuación, adjuntándose además su aplicación a un problema de tamaño reducido.

4.1. Algoritmo generador de la secuencia inicial (ISA)

En este algoritmo, llamado ISA por sus siglas en inglés, *Initial Sequence Algorithm*, y cuyo pseudocódigo se adjunta en la Figura 4.1, se ordenan los trabajos basándose en el tiempo ocioso que habría en las máquinas, intentando que sea lo menor posible, especialmente entre el procesado de los primeros trabajos secuenciados.

Para ello, se define un índice para cada uno de los trabajos, denominado ind, que depende de:

- CI: Tiempo ocioso actual, el que tiene lugar antes de procesar el trabajo en cuestión, sumado para todas las máquinas.
- FI: Tiempo ocioso futuro, entre dicho trabajo y el posterior, sumado para todas las máquinas. Sin embargo, como no se puede conocer cuál es el trabajo posterior, se construye artificialmente uno cuyos tiempos de proceso se obtienen a partir de la media de los tiempos en cada una de las máquinas de los trabajos aún no secuenciados.

Entrada: datos del problema (m, n, p_{ij})

Salida: secuencia inicial de trabajos $(secINI)$

Se define U: conjunto de trabajos aún no secuenciados, inicialmente $\{1, 2, \dots, n\}$

$For\ k = 1\ \ to\ (n-1)$

  $For\ \forall\ j \in U$

    Insertar el trabajo j en la posición k de $secINI$, y el trabajo artificial en la $k+1$

    $ind_j = (n-k) * CI + FI$

  Insertar el trabajo con menor ind_jen la posición k de $secINI$, y extraerlo de U

Insertar el único trabajo restante en el conjunto U en la posición n de $secINI$

Figura 4.1. Pseudocódigo del algoritmo ISA.

Con el fin de hacer más entendible este algoritmo, se detalla a continuación cómo se aplica a un problema sencillo de 4 trabajos y 3 máquinas, con los tiempos de proceso indicados en la Tabla 4.1.

p_{ij}		*trabajo (j)*			
		1	***2***	***3***	***4***
máquina (i)	***1***	2	8	5	1
	2	5	9	9	3
	3	6	6	2	8

Tabla 4.1. Tiempos de proceso para un problema genérico con cuatro trabajos y tres máquinas.

En primer lugar, el trabajo 1 se coloca en la primera posición de la secuencia. Posteriormente, se calcula la media de los tiempos de proceso del resto de trabajos para cada una de las máquinas (APT*: AverageProcessing Times*), y esos son los tiempos de proceso del trabajo artificial, que se situaría tras el trabajo 1, obteniendo la situación representada en la Figura 4.2.

$$APT_{234}\ (máquina\ 1) = (8 + 5 + 1)/3 = 4{,}67$$

$$APT_{234}\ (máquina\ 2) = (9 + 9 + 3)/3 = 7$$

$$APT_{234}\ (máquina\ 3) = (6 + 2 + 8)/3 = 5{,}33$$

Nótese que al trabajo artificial también se le impide permanecer en espera entre dos máquinas, como a cualquier otro trabajo real.

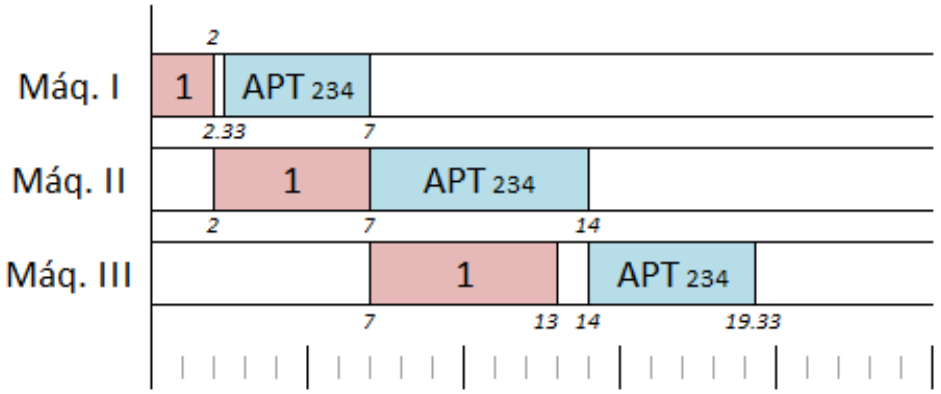

Figura 4.2. Diagrama de Gantt obtenido al secuenciar el trabajo 1 en primer lugar, y posteriormente el trabajo artificial.

En este caso, los tiempos ociosos actuales y futuros para el trabajo 1 se calculan como sigue:

$$CI_1 = (0 - 0) + (2 - 0) + (7 - 0) = 9$$

$$FI_1 = (2{,}33 - 2) + (7 - 7) + (14 - 13) = 1{,}33$$

$$ind_1 = (4 - 1) * 9 + 1{,}33 = 28{,}33$$

Este proceso se repite con cada uno de los trabajos, obteniendo los resultados mostrados en la Tabla 4.2. Como al secuenciar el trabajo 4 es cuando se obtiene un menor valor del índice, es dicho trabajo el que ocupa la primera posición de la secuencia inicial.

		1	2	3	4
k=1	CI	9	25	19	5
	FI	1,33	7	9	3,67
	ind	28,33	82	66	**18,67**
		secINI = [4, ...]			

Tabla 4.2. Cálculo de los tiempos ociosos y del índice para cada uno de los trabajos, y selección del primer trabajo de la secuencia inicial.

En la segunda iteración, se sitúan todos los trabajos restantes en la segunda posición de la secuencia, pues el trabajo 4 está ya fijado en la primera. Por ejemplo, al asignar el trabajo 1 a la segunda posición y secuenciar tras él el trabajo artificial (construido esta vez a partir de los tiempos de proceso de los trabajos 2 y 3), se obtiene el Diagrama de Gantt representado en la Figura 4.3.

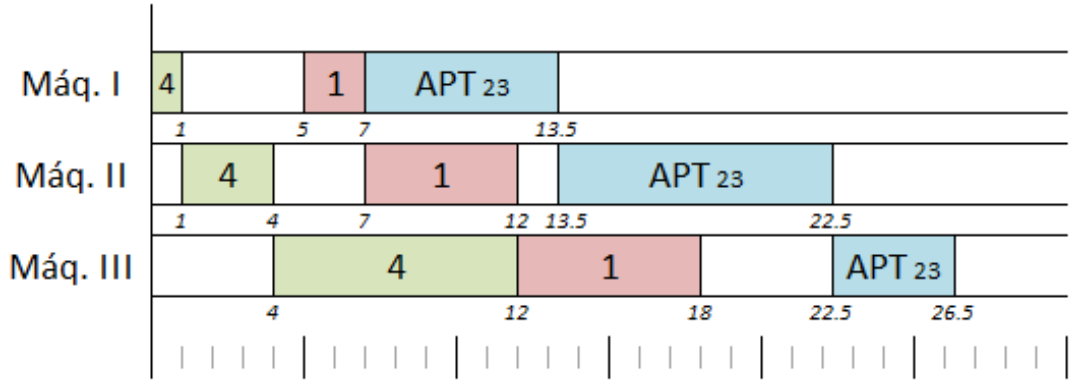

Figura 4.3. Diagrama de Gantt obtenido, una vez se ha fijado el trabajo 4 al inicio de la secuencia, al secuenciar el trabajo 1 en segundo lugar, y posteriormente el trabajo artificial.

$$CI_1 = (5-1) + (7-4) + (12-12) = 7$$

$$FI_1 = (7-7) + (13{,}5-12) + (22{,}5-18) = 6$$

$$ind_1 = (4-2) * 7 + 6 = 20$$

De igual forma, se calculan los índices para el resto de trabajos en la Tabla 4.3, hallando el segundo trabajo de la secuencia inicial. El algoritmo sigue aplicándose hasta que ya sólo quede un trabajo por ser secuenciado, que evidentemente se coloca en la última posición de la secuencia, como se puede observar en la Tabla 4.4, que resume el proceso completo.

		1	2	3	4
k=2	CI	7	11	5	
	FI	6	5,5	9	
	ind	20	27,5	**19**	
		secINI = [4, 3, ...]			

Tabla 4.3. Cálculo de los tiempos ociosos y del índice para cada uno de los trabajos restantes, y selección del segundo trabajo de la secuencia inicial.

		1	2	3	4
k=1	CI	9	25	19	5
	FI	1,33	7	9	3,67
	ind	28,33	82	66	**18,67**
		secINI = [4, ...]			
k=2	CI	7	11	5	
	FI	6	5,5	9	
	ind	20	27,5	**19**	
		secINI = [4, 3, ...]			
k=3	CI	9	8		
	FI	9	9		
	ind	18	**17**		
		secINI = [4, 3, 2, ...]			
		secINI = [4, 3, 2, 1]			

Tabla 4.4. Cálculo de los tiempos ociosos y del índice para cada uno de los trabajos restantes en cada iteración, y obtención de la secuencia inicial.

4.2. Heurística para encontrar mejoras a la secuencia inicial (CFI)

En esta segunda etapa, partiendo de la secuencia inicial obtenida en la anterior, se obtienen nuevas secuencias mediante un proceso iterativo de inserción e intercambio de trabajos, cuyos pasos se describen en la Figura 4.4.

1. Fijar la variable $r = 1$. Se define $secBEST$ como la secuencia de n trabajos con la que se consigue un mejor resultado. Inicialmente, $secBEST$ es la secuencia obtenida mediante ISA.
2. Seleccionar los dos primeros trabajos de $secBEST$, y elegir la secuencia parcial con menor TCT.
3. En primer lugar, insertar el siguiente trabajo de $secBEST$ en cada una de las posibles posiciones, obteniendo nuevas y más largas secuencias parciales, y elegir la mejor de ellas. Posteriormente, intercambiar dos a dos los trabajos de ésta última para generar otras secuencias, y sustituir la anterior por la mejor de ellas en caso de encontrarse alguna que dé un mejor resultado.

4. Repetir el paso 3 hasta que todos los trabajos estén secuenciados, y nombrar a la secuencia resultante de tamaño n $secP$.

5. Desde $j = 1$ hasta $(n-1)$, insertar el trabajo j de $secP$ en las $(n-j)$ posibles posiciones posteriores en la secuencia. Si alguna de las secuencias halladas en este paso (o la propia $secP$) mejora a $secBEST$, actualizar ésta última por la mejor de ellas.

6. Aumentar en 1 el valor de r. Si $r \leq 6$, volver al paso 2.

7. Definir $secCFI = secBEST$, la secuencia finalmente obtenida tras aplicar el algoritmo.

Figura 4.4. Pasos de la heurística CFI.

Se adjunta, al igual que para el algoritmo de generación de la secuencia inicial, la aplicación de esta heurística al mismo ejemplo de 4 trabajos y 3 máquinas. En primer lugar, cada iteración parte de la mejor ecuación hallada hasta el momento, denominada $secBEST$, que en la primera es la obtenida mediante el algoritmo ISA, como muestra la Figura 4.5.

secBEST = secINI	**4**	**3**	**2**	**1**	*95*

Figura 4.5. Paso 1 de la heurística CFI.

La Figura 4.6 representa el segundo paso de la heurística, en el que se forman las dos secuencias posibles de dos trabajos que se pueden construir con los dos primeros trabajos de la secuencia inicial, quedándonos con la mejor.

4	**3**	*29*
3	**4**	*41*

Figura 4.6. Paso 2 de la heurística CFI.

Posteriormente, el paso 3 para el trabajo situado en tercer lugar en la secuencia inicial consiste en insertarlo en las tres posibles posiciones de la secuencia parcial que se obtuvo en el paso anterior. A la mejor de esas tres secuencias se le intercambian dos a dos todos sus trabajos, buscando una nueva secuencia que la mejore, como se observa en la Figura 4.7.

2	**4**	**3**	*90*
4	**2**	**3**	*65*
4	**3**	**2**	*59*

->

3	**4**	**2**	*78*
2	**3**	**4**	*88*
4	**2**	**3**	*65*

Figura 4.7. Paso 3 de la heurística CFI para el tercer trabajo.

A continuación, en la Figura 4.8 se repite el proceso anterior, pero esta vez con el cuarto trabajo de la secuencia inicial.

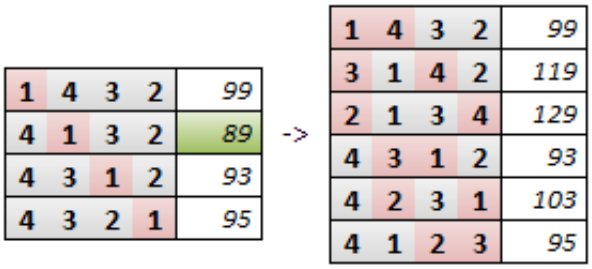

Figura 4.8. Paso 3 de la heurística CFI para el cuarto trabajo.

La mejor secuencia obtenida al finalizar el paso 4 de la heurística es la que se denomina $secP$. En el paso 5 se insertan hacia delante sus trabajos, generando otras $(n-1)+(n-2)+\cdots+1$ secuencias diferentes, como muestra la Figura 4.9.

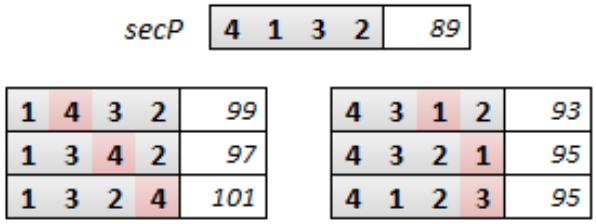

Figura 4.9. Paso 5 de la heurística CFI.

Como se ha encontrado una secuencia con la que se obtiene un resultado mejor que con la que hasta entonces se tenía como la mejor, dicha secuencia se convierte ahora en la referencia. Finalizada la primera iteración, a partir de esta nueva $secBEST$ se repiten los pasos de la heurística a partir del 2 de igual forma hasta realizarlos en seis ocasiones, tras las cuales se termina obteniendo la secuencia resultado de la heurística, a la que se denomina $secCFI$, y que se adjunta en la Figura 4.10.

Figura 4.10. Secuencia resultado de aplicar la heurística CFI a la secuencia obtenida mediante ISA.

Esta heurística realiza seis iteraciones, como se indica en el paso 6 del pseudocódigo mostrado en la Figura 4.4, algo que en ocasiones puede ser redundante, pues si en una iteración no se logra mejorar y la secuencia final es igual a la inicial, toda iteración que se haga posterior a ella solamente va a implicar un aumento en el tiempo computacional empleado, pero nunca una reducción del valor de la función objetivo. Por ello, si se impusiera que no mejorar en una iteración implicara finalizar la heurística, se obtendrían los mismos resultados con un menor tiempo computacional. Al considerar esta modificación, la heurística se denomina CFI interrumpido.

5. TÉCNICAS DE BÚSQUEDA LOCAL

Todos los algoritmos y procesos que se plantean en este trabajo parten del algoritmo ISA como solución inicial. Esto lo debe a que, como ya se explicó en la revisión de la literatura, dicha heurística es una de las mejores y más habitualmente usadas en este tipo de problemas.

Con el objeto de encontrar secuencias que mejoren a la obtenida inicialmente, se consideran tres técnicas distintas de búsqueda local o búsqueda descendiente simple, cada una de ellas con tres criterios de actuación al hallar mejoras. Asimismo, también se considera una técnica basada en la NEH de inserción de trabajos. A continuación, se explican estas diez técnicas, y se aplican al problema reflejado en la Tabla 4.1, partiendo siempre de la misma secuencia inicial: [1, 2, 3, 4], para la cual se obtiene un valor de la función objetivo de 107.

5.1. Intercambio de trabajos

En este tipo de búsqueda, se encuentran nuevas secuencias a partir de intercambios de trabajos dos a dos. Según cómo se actúe al hallar soluciones mejores a la que se tenía previamente, se subdividen estas técnicas en otras tres.

5.1.1. Intercambio de trabajos – BEST (*swapB*)

Partiendo de la secuencia referencia, de tamaño n, se intercambian todos los trabajos dos a dos como se explica en la Figura 5.1.

1. Definir como $secBEST$ a la secuencia de n trabajos de la que se parte.
2. Fijar la variable $i = 1$.
3. Desde $j = i + 1$ hasta n, intercambiar los elementos en las posiciones i y j, obteniendo $(n - i)$ nuevas secuencias.
4. Aumentar en 1 el valor de i. Si $i < n$, volver al paso 2.
5. Sustituir $secBEST$ por la mejor de las $(n-1) + (n-2) + \cdots + 1$ secuencias generadas en esta iteración en caso de que la mejore. Y si es así, volver al paso 2.

Figura 5.1. Pasos de la técnica Intercambio de trabajos (BEST).

En el ejemplo representado en la Figura 5.2, como se tienen cuatro trabajos, se generan seis secuencias nuevas en cada iteración. En la primera de ellas se observa que la mejor de esas secuencias es la obtenida al intercambiar los trabajos 2 y 4, con la que se obtiene un objetivo de 99, que mejora el previo, 107.

La secuencia [1, 4, 3, 2] pasa a ser la nueva referencia, y se generan otras seis secuencias, obteniéndose esta vez una con objetivo 89 al permutar los trabajos 1 y 4.

Se generan otras seis secuencias a partir de la que entonces es referencia, pero ninguna de ellas la mejora, por lo que [4, 1, 3, 2] es la secuencia resultado de aplicar esta técnica de búsqueda local.

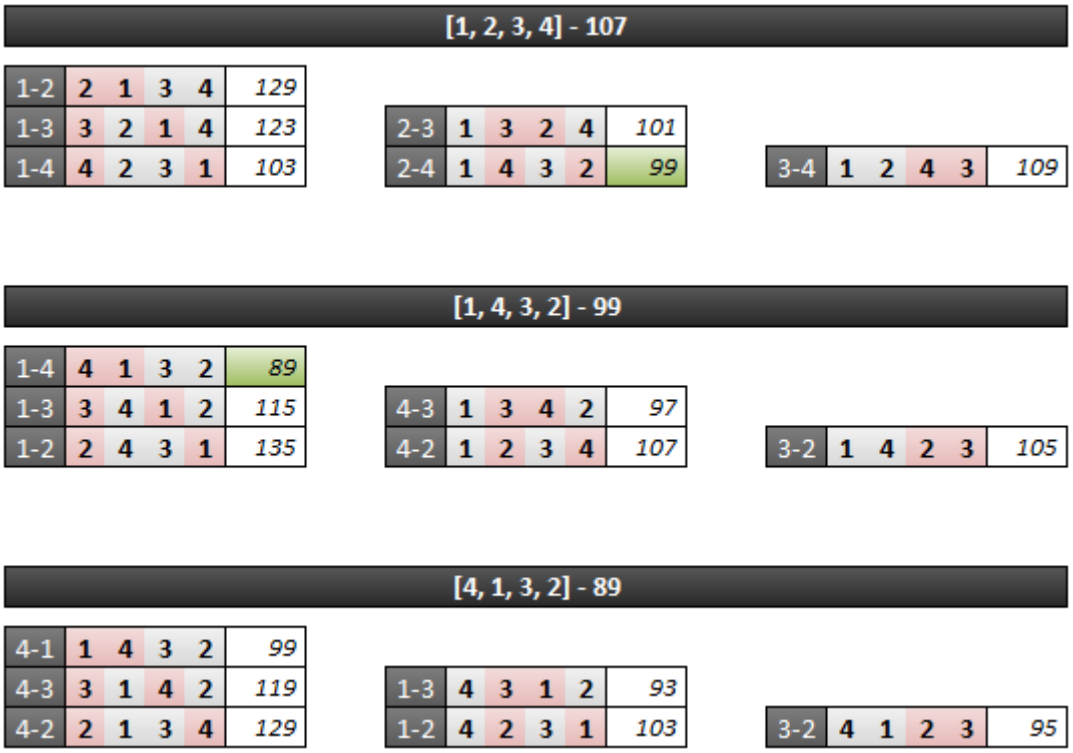

Figura 5.2. Aplicación de la técnica Intercambio de trabajos (BEST) al problema de la Tabla 4.1.

5.1.2. Intercambio de trabajos – FIRST (*swapF*)

En este caso, se van obteniendo nuevas secuencias de igual forma a partir de la referencia, pero se deja de buscar en el momento en el que una de ellas ya sea mejor que la referencia, pasando directamente a la siguiente iteración.

Como se observa en la Figura 5.3, cuando se halla la secuencia [4, 2, 3, 1], que mejora a la referencia inicial, se da por finalizada la primera iteración y se reinicia el proceso de generación de nuevas secuencias, por lo que en esta técnica no tienen por qué generarse esas seis secuencias en cada una de las iteraciones.

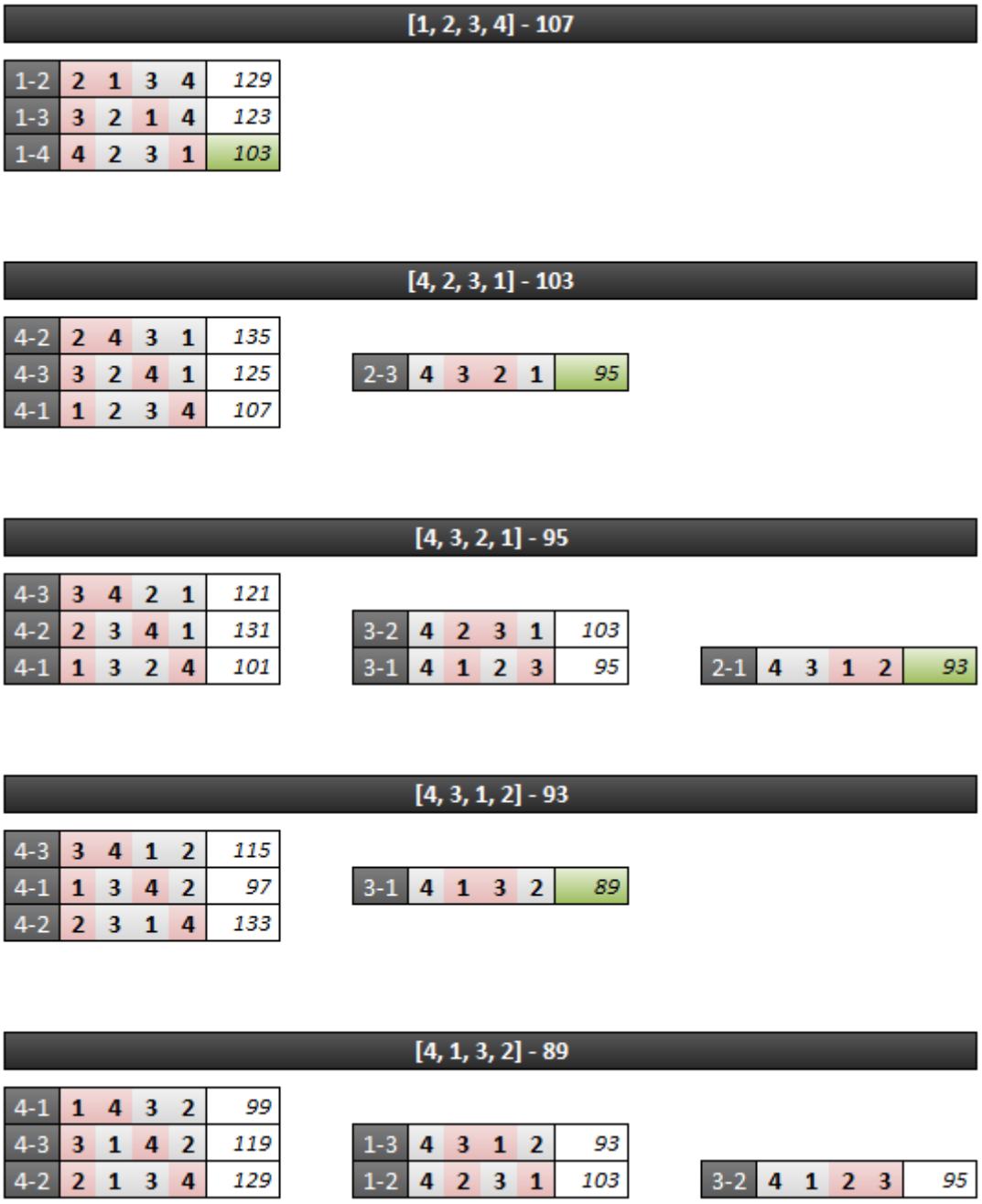

Figura 5.3. Aplicación de la técnica Intercambio de trabajos (FIRST) al problema de la Tabla 4.1.

5.1.3. Intercambio de trabajos – FIRST BEST (*swapFB*)

Este criterio combina características de los dos anteriores. Por un lado, se vuelven a generar seis secuencias en cada iteración, pero por otro, las secuencias generadas que mejoran a la referencia se convierten en las nuevas referentes incluso antes de dar por finalizada la iteración en cuestión, aplicándose sobre ellas los intercambios restantes.

En la Figura 5.4 se aprecia que en la primera iteración, la primera secuencia hallada que mejora a la referencia es la obtenida al intercambiar 1 y 4. Pues [4, 2, 3, 1] se convierte en la nueva referencia, y en ella se intercambian los trabajos que aún no habían sido intercambiados en la original. Como al intercambiar 2 y 3 se vuelve a obtener una mejor secuencia, [4, 3, 2, 1], los intercambios restantes se realizan sobre ella, hasta haber obtenido las seis nuevas secuencias y ya sí dar por finalizada la iteración.

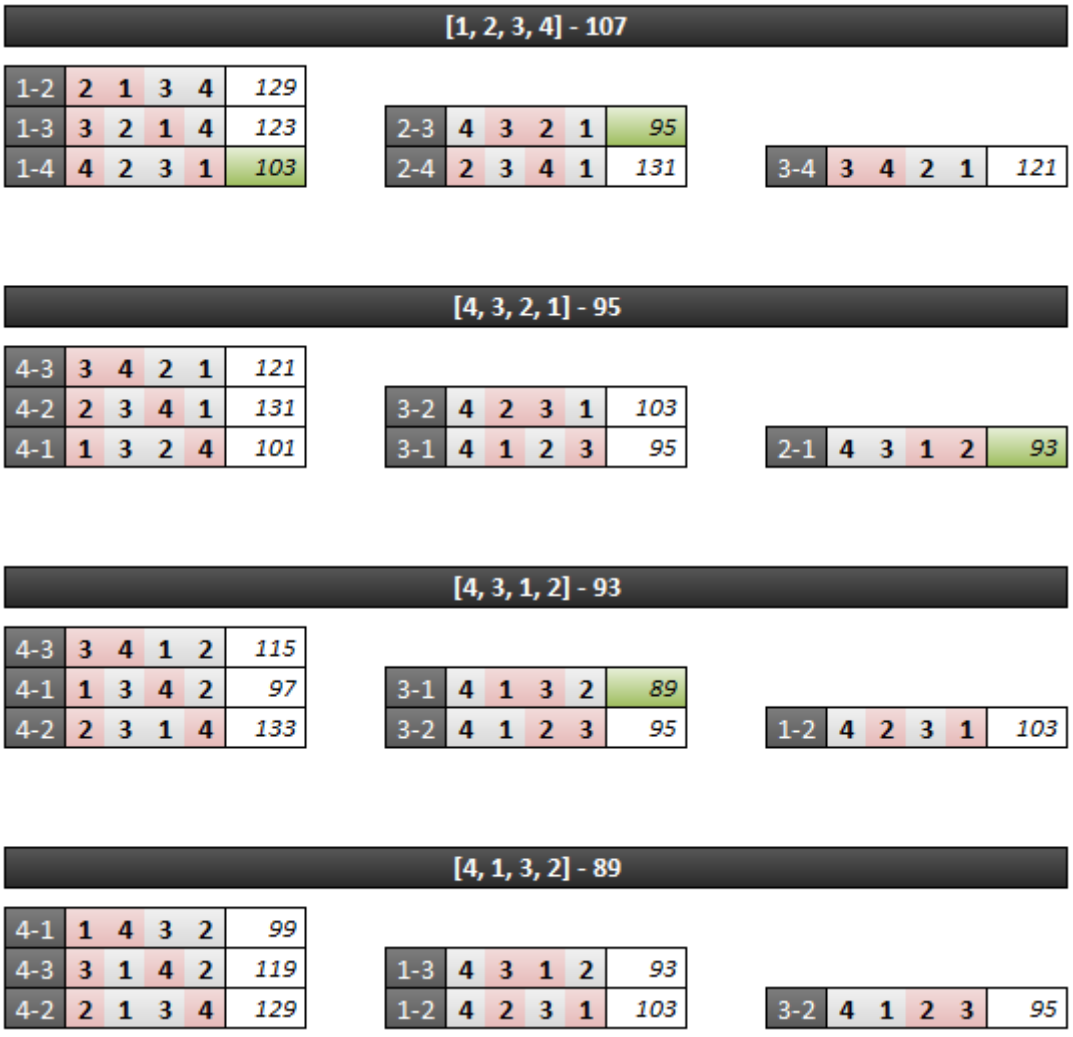

Figura 5.4. Aplicación de la técnica Intercambio de trabajos (FIRST BEST) al problema de la Tabla 4.1.

5.2. Inserción completa de trabajos

En este tipo de búsqueda, se generan nuevas secuencias a partir de la inserción de cada uno de los trabajos en el resto de posiciones. Análogamente, según cómo se actúe al hallar soluciones mejores a las que se tenían previamente, se subdividen estas técnicas en otras tres.

5.2.1. Inserción completa de trabajos – BEST (*insB*)

Partiendo de la secuencia referencia, cada uno de los trabajos se coloca en las otras $(n-1)$ posibles posiciones de la secuencia respetando el orden del resto de trabajos, obteniendo $n*(n-1)$ nuevas secuencias.

Si la mejor de las secuencias generadas mejora a la que hasta entonces aportaba un mejor resultado, ésta se convierte en la secuencia referencia, y se repite el proceso.

La Figura 5.5 muestra un problema en el que ninguna de las doce nuevas secuencias generadas en la tercera iteración consigue mejorar a la secuencia referencia, por lo que ésa es la que se obtiene empleando esta técnica.

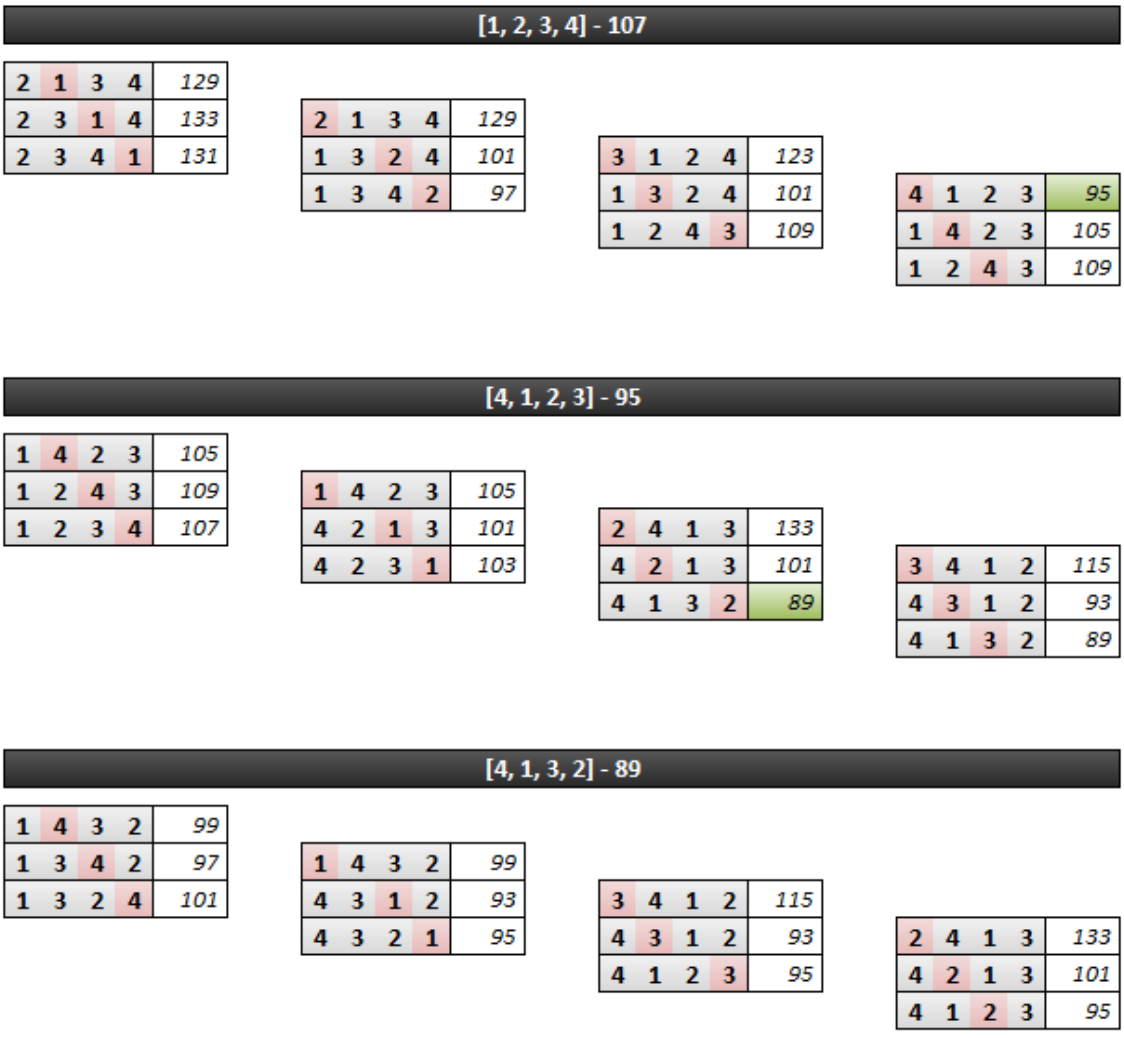

Figura 5.5. Aplicación de la técnica Inserción de trabajos (BEST) al problema de la Tabla 4.1.

5.2.2. Inserción completa de trabajos – FIRST (*insF*)

En este caso, se van obteniendo nuevas secuencias de igual forma a partir de la que se tiene como referencia, pero se deja de buscar en el momento en el que una de ellas ya sea mejor que la que se tenía, pasando directamente a la siguiente iteración.

Como se observa en la Figura 5.6, cuando se halla la secuencia [1, 3, 2, 4], que mejora a la referencia inicial, se da por finalizada la primera iteración y se reinicia el proceso de generación de nuevas secuencias, por lo que en esta técnica no tienen por qué generarse esas doce secuencias en cada una de las iteraciones.

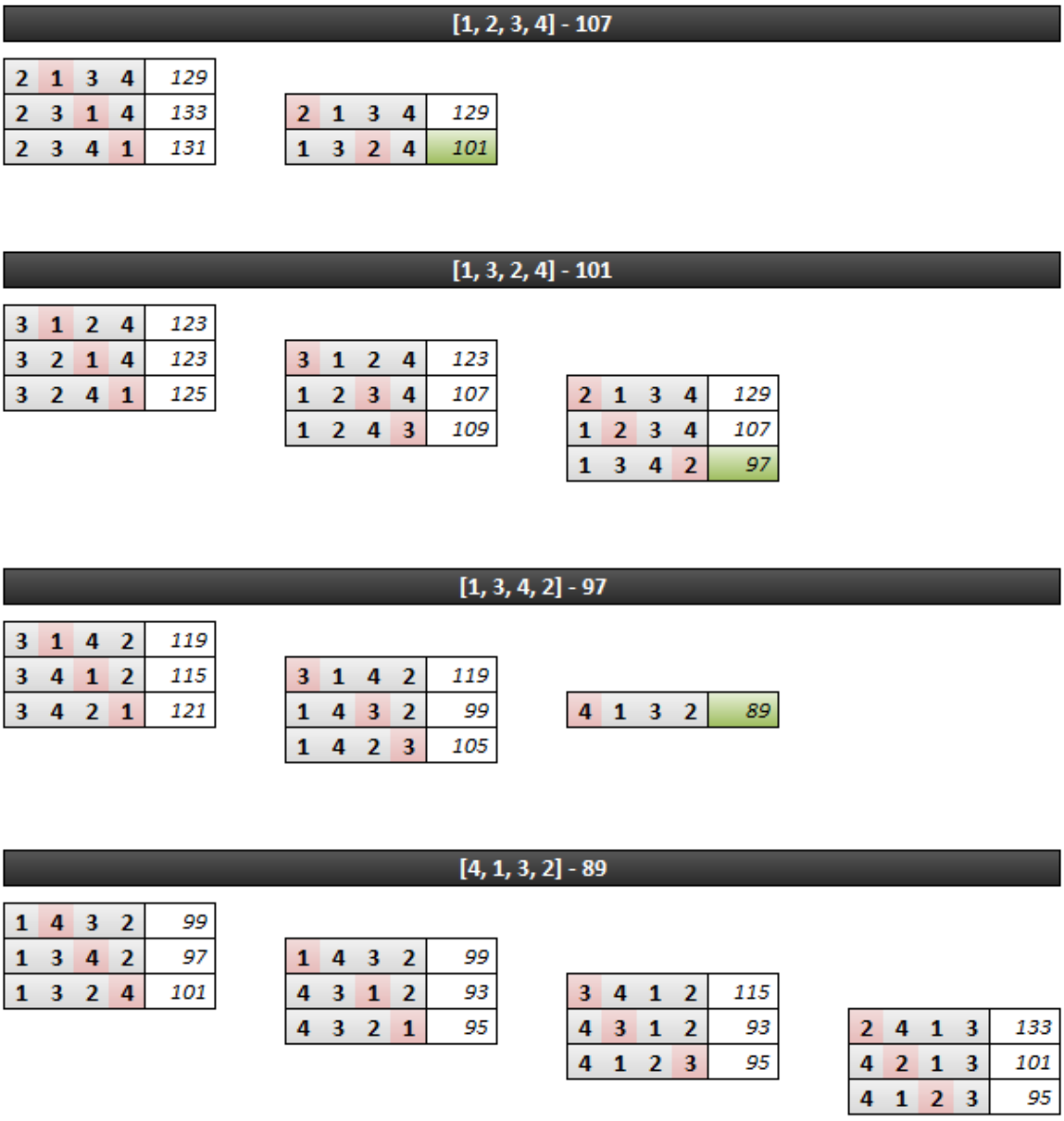

Figura 5.6. Aplicación de la técnica Inserción de trabajos (FIRST) al problema de la Tabla 4.1.

5.2.3. Inserción completa de trabajos – FIRST BEST (*insFB*)

A diferencia de lo que sucedía con los intercambios de trabajos, no se genera la misma cantidad de secuencias por iteración que en caso BEST, pues en éste se hallan n^2 nuevas soluciones en cada una de ellas, como puede apreciarse en el problema de la Figura 5.7.

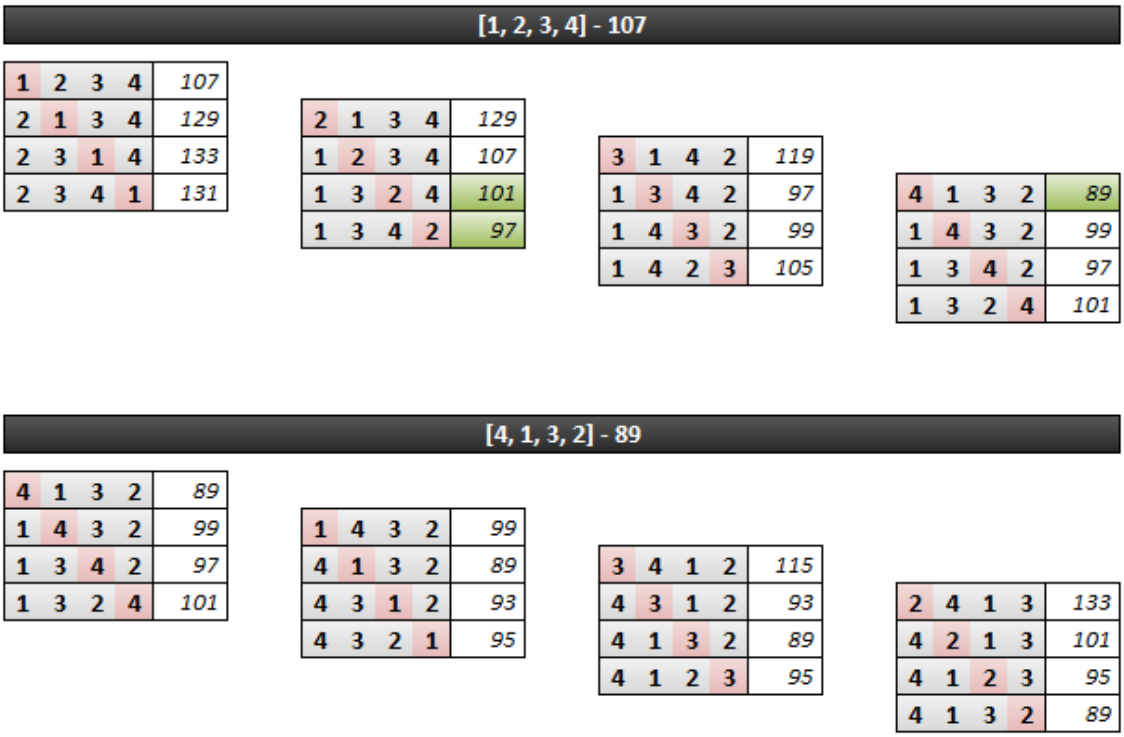

Figura 5.7. Aplicación de la técnica Inserción de trabajos (FIRST BEST) al problema de la Tabla 4.1.

5.3. Inserción hacia delante de trabajos

En este tipo de búsqueda, se encuentran nuevas secuencias a partir de la inserción de cada trabajo sólo en las posiciones posteriores de la secuencia. Según cómo se actúe al hallar soluciones mejores a las que se tenían previamente, se subdividen estas técnicas en otras tres.

5.3.1. Inserción hacia delante de trabajos – BEST (*insforB*)

Partiendo de la secuencia referencia, cada uno de los trabajos se inserta en todas las posiciones posteriores de la secuencia respetando el orden del resto de trabajos. El trabajo en la posición j puede insertarse en $(n-j)$ posiciones distintas, generando otras tantas secuencias. En total, en cada iteración se generan $(n-1)+(n-2)+\cdots+1$ nuevas secuencias.

La Figura 5.8 muestra un problema en el que, al haber cuatro trabajos, se generan seis secuencias en cada iteración, y en el que la secuencia resultado de la primera ya no se mejora en la segunda, por lo que ésa es la secuencia resultado de aplicar esta técnica.

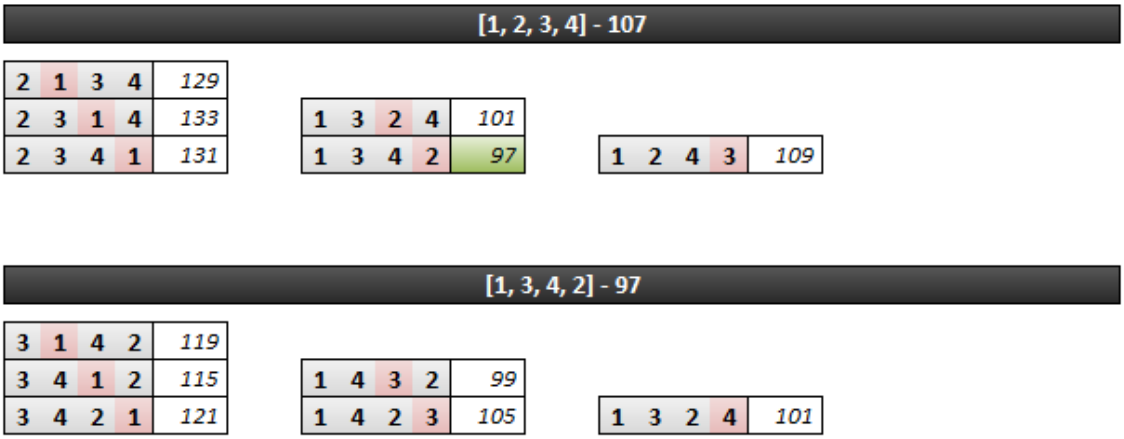

Figura 5.8. Aplicación de la técnica Inserción hacia delante de trabajos (BEST) al problema de la Tabla 4.1

5.3.2. Inserción hacia delante de trabajos – FIRST (*insforF*)

En este caso, se van obteniendo nuevas secuencias de igual forma a partir de la que se tiene como referencia, pero se deja de buscar en el momento de que una de ellas ya sea mejor que la que se tenía, pasando directamente a la siguiente iteración.

Como se observa en la Figura 5.9, cuando se halla la secuencia [1, 3, 2, 4], que mejora a la referencia inicial, se da por finalizada la primera iteración y se reinicia el proceso de generación de nuevas secuencias, por lo que en esta técnica no tienen por qué generarse esas seis secuencias en cada una de las iteraciones.

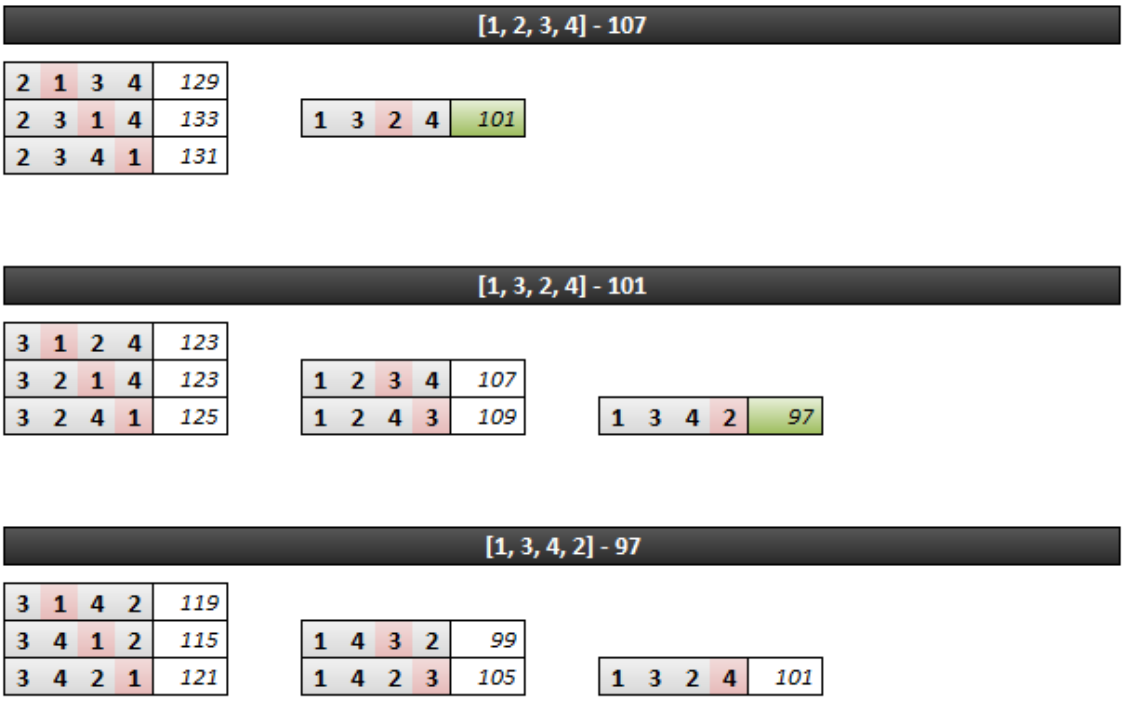

Figura 5.9. Aplicación de la técnica Inserción hacia delante de trabajos (FIRST) al problema de la Tabla 4.1

5.3.3. Inserción hacia delante de trabajos – FIRST BEST (*insforFB*)

En este caso, el número de secuencias nuevas que se generan en cada iteración no es fijo, pues depende de la posición que ocupen los trabajos que aún no han sido insertados en el resto de posiciones cada vez que se encuentre una nueva secuencia referencia.

El mínimo de secuencias nuevas que se generan en cada iteración es seis, pero si se hallan soluciones mejores a la que se tenía hasta ese instante, puede ocurrir que se obtengan más de seis secuencias, como sucede en la primera iteración del ejemplo mostrado en la Figura 5.10.

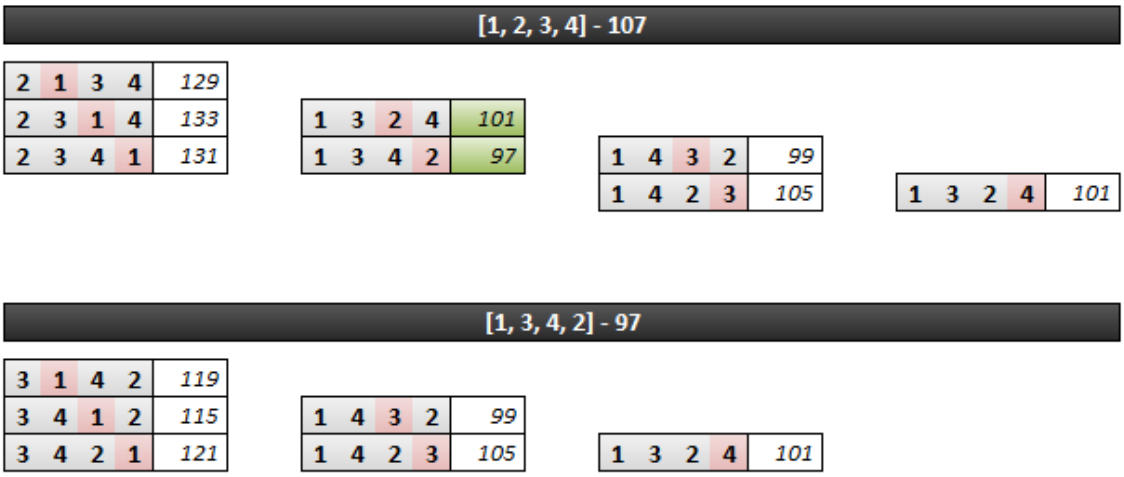

Figura 5.10. Aplicación de la técnica Inserción hacia delante de trabajos (FIRST BEST) al problema de la Tabla 4.1.

5.4. Técnica de búsqueda local basada en la NEH ($\mathbf{NEH_{LOC}}$*)*

Se trata de un procedimiento basado en una heurística constructiva habitual en la programación de operaciones como la NEH. En ella, partiendo de una secuencia inicial, se insertan uno a uno sus trabajos en todas las posibles posiciones de una nueva secuencia, quedándose con la mejor secuencia parcial obtenida en cada momento, hasta haber insertado todos los trabajos y obtenido una nueva secuencia final.

Los pasos para aplicar la técnica de inserción de trabajos NEH estándar se explican en la Figura 5.11, mostrándose además una aplicación de la misma en el ejemplo representado en la Figura 5.12.

1. Se consideran los dos primeros trabajos de la secuencia inicial ($secINI$), de tamaño n, y la mejor de las dos secuencias parciales que pueden formar dichos trabajos se convierte en la secuencia resultado provisional ($secRES$).

2. Se crea la variable i, que inicialmente toma el valor 3.

3. Se toma el siguiente trabajo de $secINI$ (el que está en la posición i), y se inserta en todas las posibles posiciones de $secRES$, obteniendo i secuencias parciales de tamaño i. La mejor de todas ellas es la nueva $secRES$.

4. Si $i < n$, se aumenta en 1 unidad el valor de i y se vuelve el paso 3.

Figura 5.11. Pasos de la heurística NEH.

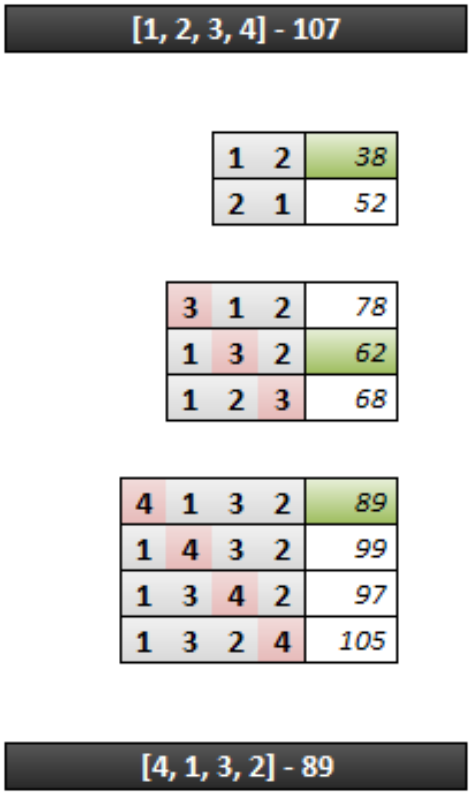

Figura 5.12. Aplicación de la técnica de inserción de trabajos NEH al problema de la Tabla 4.1.

Dependiendo del problema y de la secuencia inicial desde la que se parta, pudiera suceder que la secuencia final fuese peor que la inicial, algo que nunca ocurre con las nueve técnicas anteriores. Esto sucede porque la NEH se trata de una heurística constructiva, así que para asegurarse de que este procedimiento propuesto sea siempre un método de mejora, se compara la secuencia inicial con la obtenida finalmente, y se mantiene la primera en caso de que la segunda no la mejore. Al añadirle esta modificación a la técnica *NEH*, se denomina NEH_{LOC} al procedimiento descrito en este párrafo.

6. HEURÍSTICAS PROPUESTAS

Partiendo del algoritmo de generación de secuencia inicial (ISA) descrito en el punto 4, se combinan algunas de las técnicas de mejora local explicadas en el capítulo anterior. Según cuáles de ellas se apliquen y en qué orden se haga, se plantean distintas heurísticas, cuya evaluación se trata en el capítulo 8.

Las dos primeras combinan las técnicas insFB y NEH_{LOC} en los dos posibles órdenes, mientras que las dos siguientes aplican en primer lugar la NEH_{LOC} y posteriormente las insFB y swapFB, también en los dos órdenes posibles. Se presenta también una quinta heurística en la que dichas técnicas se aplican en múltiples ocasiones.

6.1. Heurística CML-I

Esta primera heurística planteada consiste en, partiendo de la secuencia inicial, realizar una combinación de dos de las técnicas de búsqueda local explicadas en el capítulo anterior.

1. Se obtiene la secuencia π_{ini} mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica insFB a π_{ini}, obteniendo la secuencia π_I.
3. Se aplica la técnica NEH_{LOC} a π_I, obteniendo la secuencia π_{fin}.

Figura 6.1. Pasos de la heurística CML-I.

6.2. Heurística CML-II

En esta heurística se invierte respeto a la CML-I el orden de las dos técnicas de búsqueda local aplicadas tras la obtención de la secuencia inicial.

1. Se obtiene la secuencia π_{ini} mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica NEH_{LOC} a π_{ini}, obteniendo la secuencia π_I.
3. Se aplica la técnica insFB a π_I, obteniendo la secuencia π_{fin}.

Figura 6.2. Pasos de la heurística CML-II.

6.3. Heurística CML-III

Esta heurística incluye una nueva técnica de búsqueda local en comparación a las dos anteriores, como es la swapFB, que se aplica en último lugar. Es idéntica a la CML-I, salvo por precisamente ese último paso.

1. Se obtiene la secuencia π_{ini} mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica NEH_{LOC} a π_{ini}, obteniendo la secuencia π_I.
3. Se aplica la técnica insFB a π_I, obteniendo la secuencia π_{II}.
4. Se aplica la técnica swapFB a π_{II}, obteniendo la secuencia π_{fin}.

Figura 6.3. Pasos de la heurística CML-III.

6.4. Heurística CML-IV

Esta heurística invierte respeto a la CML-III el orden de los pasos 3 y 4, es decir, de las dos técnicas de búsqueda local aplicadas tras la obtención de la secuencia inicial y la aplicación de la NEH_{LOC}.

1. Se obtiene la secuencia π_{ini} mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica NEH_{LOC} a π_{ini}, obteniendo la secuencia π_I.
3. Se aplica la técnica swapFB a π_I, obteniendo la secuencia π_{II}.
4. Se aplica la técnica insFB a π_{II}, obteniendo la secuencia π_{fin}.

Figura 6.4. Pasos de la heurística CML-IV.

6.5. Heurística CML-V

Esta heurística, siendo similar a la CML-III, contempla en varios puntos del proceso distintas alternativas, según qué técnica vaya logrando mejores resultados.

1. Se obtiene la secuencia π_{ini} mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica a π_{ini} la técnica NEH_{LOC}, obteniendo π_I, y a ésta se le aplica la insFB, obteniendo la secuencia π_{II}.
3. Se aplica la técnica insFB a π_{ini}, obteniendo la secuencia π_{III}.
4. Se comparan las secuencias π_{II} y π_{III}, conservando solamente la mejor, renombrada como π_{IV}.
5. Se repiten los pasos 2,3 y 4, siendo en este caso π_{IV} la nueva secuencia inicial.
6. Se aplica la técnica swapFB a π_{IV}, obteniendo π_{fin}.

Figura 6.5. Pasos de la heurística CLM-V.

7. METAHEURÍSTICAS PROPUESTAS

Las heurísticas presentadas en el capítulo anterior pretendían encontrar buenas soluciones utilizando tiempos computacionales pequeños, con el fin de obtener soluciones rápidas que además pudieran usarse como soluciones iniciales de metaheurísticas más complejas. Sin embargo, no dejan de ser algoritmos deterministas que siempre emplean el mismo tiempo y llegan a idénticos resultados.

Con el fin de ampliar las posibilidades de búsquedas de óptimos locales, se plantea un nuevo algoritmo de generación de secuencias iniciales, basado en explicado en el capítulo 4 pero que incluye cierta aleatoriedad. Esto, además, hace que las metaheurísticas que se plantean en este capítulo puedan llegar a mejores resultados si se ejecutan durante más tiempo, a diferencia de lo que ocurría con todas las propuestas hasta este punto.

7.1. Algoritmo de generación de secuencia iniciales alternativas (ISA-GRASP)

Dada la complejidad del problema, se presenta un procedimiento de búsqueda GRASP (*Greedy Randomized Adaptative Search Procedure*) para resolverlo. Este tipo de procedimientos fueron propuestos por primera vez por Feo y Resende (1989), y desde entonces se han aplicado frecuentemente en problemas de optimización. Festa y Resende (2001) realizaron una revisión de los distintos usos de estos algoritmos.

El algoritmo ISA-GRASP es similar al ISA original ya explicado en el capítulo 4, con la diferencia de que en lugar de seleccionar el trabajo con menor valor del índice en cada iteración, escoge aleatoriamente uno de los dos elementos con menor índice. Es decir, se adapta parte del algoritmo ISA para integrarlo en una GRASP.

Entrada: datos del problema (m, n, p_{ij})

Salida: secuencia inicial de trabajos $(secINI)$

Se define U: conjunto de trabajos aún no secuenciados, inicialmente $\{1, 2, \dots, n\}$

$For\ k = 1\ \ to\ (n-1)$

$For\ \forall\ j \in U$

Insertar el trabajo j en la posición k de $secINI$, y el trabajo artificial en la $k+1$

$ind_j = (n-k) * CI + FI$

Identificar los dos trabajos con menor ind_j, y seleccionar uno de ellos aleatoriamente

Insertar el trabajo seleccionado en la posición k de $secINI$, y extraerlo de U

Insertar el único trabajo restante en el conjunto U en la posición n de $secINI$

Figura 7.1. Pseudocódigo del algoritmo ISA-GRASP.

Al aplicar el algoritmo ISA en el problema ejemplo con los tiempos de proceso que se muestran en la Tabla 4.1, cuando se quería hallar el primer elemento de la secuencia inicial se calculaba el índice correspondiente a cada uno de ellos a partir de sus tiempos ociosos actuales y futuros, y se seleccionaba el trabajo 4 por ser el de menor índice, como demuestra la Tabla 4.2 o la Tabla 7.1.

		1	2	3	4
k=1	CI	9	25	19	5
	FI	1,33	7	9	3,67
	ind	**28,33**	82	66	**18,67**

secINI = [4, ...] o secINI = [1, ...]

Tabla 7.1. Cálculo de los tiempos ociosos y del índice para cada uno de los trabajos, y selección del primer trabajo de una secuencia inicial con aleatoriedad.

Sin embargo, el ISA-GRASP escoge aleatoriamente entre los dos trabajos con menor índice, es decir, entre 4 y 1. A priori no hay evidencia que asegure que este algoritmo vaya a lograr mejores resultados que el ISA, pero sí va a permitir obtener secuencias iniciales distintas en cada aplicación, algo en lo que se basan las metaheurísticas propuestas a continuación.

Las metaheurísticas propuestas en este capítulo se basan en la repetición de una serie de pasos, partiendo en cada iteración de una secuencia inicial diferente, lo que teóricamente permite llegar a múltiples óptimos locales, pudiendo elegir el mejor de todos ellos. No obstante, todas ellas tienen unos primeros pasos en los que se trabaja con la secuencia inicial original obtenida mediante el algoritmo ISA, para así asegurar rápidamente una referencia a la cual intentar mejorar.

7.2. Metaheurística MH-I

En esta primera metaheurística, a cada secuencia inicial generada mediante el algoritmo ISA-GRASP se le aplica la técnica NEH_{LOC}. Y tras alcanzar el criterio de parada definido, a la mejor secuencia obtenida hasta el momento se le aplican otras dos técnicas de búsquedas local más: insFB y swapFB.

1. Se obtiene la secuencia π_0 mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica NEH_{LOC} a π_0, obteniendo la secuencia π_I.
3. Se aplica la técnica insFB a π_I, obteniendo la secuencia π_{II}.
4. Se define una secuencia referencia π_{ref}, por ahora igual a π_{II}.
5. Si π_{II} no mejora a π_I, se salta directamente al paso 9.

6. Se aplica la técnica NEH_{LOC} a π_{II}, obteniendo la secuencia π_{III}, que pasa a ser π_{ref}.
7. Si π_{III} no mejora a π_{II}, se salta directamente al paso 9.
8. Se aplica la técnica insFB a π_{III}, obteniendo la secuencia π_{IV}, que pasa a ser π_{ref}.
9. Se aplica la técnica swapFB a π_{ref}, obteniendo la secuencia π_{LB}.
10. Determinar el valor de la variable x, que determina el criterio de parada.
11. Fijar la variable$i = 1$.
12. Se genera una secuencia inicial usando el algoritmo ISA-GRASP,π_{ini-i}.
13. Se aplica a π_{ini-i} la técnica NEH_{LOC}, obteniendo la secuencia π_{neh-i}.
14. Aumentar en 1 el valor de i. Si $i \leq x$, volver al paso 12.
15. Se conserva sólo la mejor secuencia de todas las x secuencias obtenidas en el paso 13, descartándose las restantes. La secuencia obtenida se renombra como π_{neh}
16. Se aplica la técnica insFB a la secuencia π_{neh}, obteniendo la secuencia π_{ins}.
17. Se aplica la técnica swapFB a la secuencia π_{ins}, obteniendo la secuencia π_{swap}.
18. Se comparan las secuencias π_{swap} y π_{LB}, siendo la mejor de ellasπ_{fin}.

Figura 7.2. Pasos de la metaheurística MH-I.

El criterio de parada escogido es el número de iteraciones realizadas, es decir, el parámetro x representa la cantidad de secuencias iniciales generadas mediante el algoritmo ISA-GRASP. Es posible usar algún otro, como el tiempo computacional empleado, pero por simplicidad se decide usar el número de iteraciones.

7.3. Metaheurística MH-II

La diferencia con respecto a la anterior es que en ésta la técnica insFB es aplicada en cada una de las iteraciones, antes de descartar ninguna secuencia. Finalmente, a la mejor de todas las secuencias que se han obtenido se le aplica la swapFB, de igual forma que en MH-I.

1. Se obtiene la secuencia π_0 mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica NEH_{LOC} a π_0, obteniendo la secuencia π_I.
3. Se aplica la técnica insFB a π_I, obteniendo la secuencia π_{II}.
4. Se define una secuencia referencia π_{ref}, por ahora igual a π_{II}.
5. Si π_{II} no mejora a π_I, se salta directamente al paso 9.

6. Se aplica la técnica NEH_{LOC} a π_{II}, obteniendo la secuencia π_{III}, que pasa a ser π_{ref}.
7. Si π_{III} no mejora a π_{II}, se salta directamente al paso 9.
8. Se aplica la técnica insFB a π_{III}, obteniendo la secuencia π_{IV}, que pasa a ser π_{ref}.
9. Se aplica la técnica swapFB a π_{ref}, obteniendo la secuencia π_{LB}.
10. Determinar el valor de la variable x, que determina el criterio de parada.
11. Fijar la variable $i = 1$.
12. Se genera una secuencia inicial usando el algoritmo ISA-GRASP, π_{ini-i}.
13. Se aplica a π_{ini-i} la técnica NEH_{LOC}, obteniendo la secuencia π_{neh-i}.
14. Se aplica a π_{neh-i} la técnica insFB, obteniendo la secuencia π_{ins-i}.
15. Aumentar en 1 el valor de i. Si $i \leq x$, volver al paso 12.
16. Se conserva sólo la mejor secuencia de todas las x secuencias obtenidas en el paso 14, descartándose las restantes. La secuencia obtenida se renombra como π_{ins}.
17. Se aplica la técnica swapFB a la secuencia π_{ins}, obteniendo la secuencia π_{swap}.
18. Se comparan las secuencias π_{swap} y π_{LB}, siendo la mejor de ellas π_{fin}.

Figura 7.3. Pasos de la metaheurística MH-II.

7.4. Metaheurística MH-III

Se propone, basándose en las dos heurísticas anteriores, una tercera metaheurística que combina características de ambas.

1. Se obtiene la secuencia π_0 mediante el algoritmo de secuencia inicial (ISA).
2. Se aplica la técnica NEH_{LOC} a π_0, obteniendo la secuencia π_I.
3. Se aplica la técnica insFB a π_I, obteniendo la secuencia π_{II}.
4. Se define una secuencia referencia π_{ref}, por ahora igual a π_{II}.
5. Si π_{II} no mejora a π_I, se salta directamente al paso 9.
6. Se aplica la técnica NEH_{LOC} a π_{II}, obteniendo la secuencia π_{III}, que pasa a ser π_{ref}.
7. Si π_{III} no mejora a π_{II}, se salta directamente al paso 9.
8. Se aplica la técnica insFB a π_{III}, obteniendo la secuencia π_{IV}, que pasa a ser π_{ref}.

9. Se aplica la técnica swapFB a π_{ref}, obteniendo la secuencia π_{LB}.
10. Determinar el valor de los parámetros que determinan los criterios de parada: x, z.
11. Calcular el valor de y mediante la fórmula $y = x * z/100$. Se redondea y al entero superior en caso de que no lo sea ya.
12. Fijar la variable $i = 1$.
13. Se genera una secuencia inicial usando el algoritmo ISA-GRASP, π_{ini-i}.
14. Se aplica a π_{ini-i} la técnica NEH_{LOC}, obteniendo la secuencia π_{neh-i}.
15. Aumentar en 1 el valor de i. Si $i \leq x$, volver al paso 13.
16. Se conservan sólo las mejores y secuencias de todas las x secuencias obtenidas en el paso 14, descartándose las restantes. Las secuencias resultantes se renombran como $\pi_{neh-1}, \pi_{neh-2}, \dots, \pi_{neh-y}$.
17. Fijar la variable $j = 1$.
18. Se aplica la técnica insFB a la secuencia π_{neh-j}, obteniendo la secuencia π_{ins-j}.
19. Se aplica la técnica swapFB a la secuencia π_{ins-j}, obteniendo la secuencia π_{swap-j}.
20. Aumentar en 1 el valor de j. Si $j \leq y$, volver al paso 17.
21. Se conserva sólo la mejor secuencia de todas las y secuencias obtenidas en el paso 19, descartándose las restantes. Ésta se compara con π_{LB}, y la mejor es π_{fin}.

Figura 7.4. Pasos de la metaheurística MH-III.

Esta heurística es parametrizable por partida doble, ya que los valores de x y de z se pueden modificar dependiendo de si urge encontrar una solución o de si lo prioritario es minimizar el valor de la función objetivo. Aumentar los valores de los parámetros hará que se emplee mayor tiempo computacional, y teóricamente que se obtengan mejores resultados, aunque en este caso no puede asegurarse, ya que depende de las secuencias iniciales generadas en cada iteración.

La única relación entre los parámetros es que, por definición, $0 < z \leq 100$, y por consiguiente, $0 < y \leq x$.

En el caso particular de que los parámetros valiesen $x \geq 1$; $y = 1$, se obtendría el mismo resultado que siguiendo MH-I. Y si valieran $x \geq 1$; $y = x$, el logrado por MH-II.

8. EVALUACIÓN COMPUTACIONAL

Para comparar los distintos algoritmos de obtención de secuencias se usan las instancias de Taillard (1993), que son comúnmente usadas a la hora de probar la eficacia y eficiencia de heurísticas para problemas con entornos *flow shop*. Dichas instancias consisten en 120 problemas de 12 tamaños distintos, habiendo 10 instancias para cada uno de ellos, repartidas como se observa en la Tabla 8.1. En ellas, el número de trabajos comprende valores entre 20 y 500, el número de máquinas varía entre 5 y 20, y los tiempos de proceso de cada trabajo en cada máquina valen entre 1 y 99 unidades temporales.

Instancias	Trabajos (n)	Máquinas (m)
1-10	20	5
11-20	20	10
21-30	20	20
31-40	50	5
41-50	50	10
51-60	50	20
61-70	100	5
71-80	100	10
81-90	100	20
91-100	200	10
101-110	200	20
111-120	500	20

Tabla 8.1. Tamaño de los problemas que componen las instancias de Taillard.

8.1. Heurística referencia. ISA y CFI.

Como se comentó en el capítulo 5, todos los algoritmos que se proponen en este documento parten del ISA. Por tanto, se recopilan los TCT de las secuencias generadas por dicho algoritmo en cada una de las 120 instancias, que se muestran en la Tabla 8.2, para así tenerlas como referencia.

El principal indicador que se usa a la hora de evaluar la eficacia del resto de algoritmos es la mejora relativa de la secuencia final con respecto a la inicialmente generada por el algoritmo ISA, normalmente conocido como RPD (*Relative Percentage Deviation*). Es un indicador adimensional que no se ve afectado por los distintos tamaños de los problemas, y que se define de la siguiente forma:

$$\mathrm{RPD} = \frac{TCT\ (secuencia\ Final) - TCT\ (secuencia\ ISA)}{TCT\ (secuencia\ ISA)}$$

Algoritmo generador de la secuencia inicial (ISA)							
Instancia	TCT	*Instancia*	TCT	*Instancia*	TCT	*Instancia*	TCT
1	15 870	*31*	78 311	*61*	320 218	*91*	1 568 595
2	18 301	*32*	91 451	*62*	318 160	*92*	1 558 916
3	16 423	*33*	82 033	*63*	312 558	*93*	1 566 316
4	18 977	*34*	88 530	*64*	303 687	*94*	1 546 953
5	16 608	*35*	87 274	*65*	316 949	*95*	1 528 339
6	17 583	*36*	86 646	*66*	320 095	*96*	1 546 865
7	16 709	*37*	81 144	*67*	317 500	*97*	1 596 678
8	16 258	*38*	85 177	*68*	317 461	*98*	1 555 315
9	17 387	*39*	77 872	*69*	340 128	*99*	1 539 822
10	15 947	*40*	87 938	*70*	324 282	*100*	1 568 148
11	25 944	*41*	117 560	*71*	423 886	*101*	2 024 311
12	26 816	*42*	113 827	*72*	407 911	*102*	2 056 391
13	23 782	*43*	109 756	*73*	416 218	*103*	2 074 396
14	23 135	*44*	117 359	*74*	436 926	*104*	2 080 928
15	24 097	*45*	119 860	*75*	410 710	*105*	2 088 352
16	23 026	*46*	116 905	*76*	416 273	*106*	2 085 433
17	23 667	*47*	120 837	*77*	407 244	*107*	2 109 200
18	24 314	*48*	119 050	*78*	419 447	*108*	2 085 308
19	24 010	*49*	114 701	*79*	424 132	*109*	2 061 292
20	26 167	*50*	119 669	*80*	442 377	*110*	2 061 271
21	39 769	*51*	175 780	*81*	577 587	*111*	11 673 180
22	38 376	*52*	165 586	*82*	575 473	*112*	11 909 296
23	38 833	*53*	163 959	*83*	573 713	*113*	11 626 667
24	40 222	*54*	166 759	*84*	580 371	*114*	11 804 810
25	40 148	*55*	170 161	*85*	563 822	*115*	11 846 478
26	39 266	*56*	165 451	*86*	571 252	*116*	11 847 434
27	40 425	*57*	170 646	*87*	575 672	*117*	11 758 172
28	38 020	*58*	171 276	*88*	590 482	*118*	11 798 447
29	40 479	*59*	171 870	*89*	579 781	*119*	11 780 457
30	39 205	*60*	173 877	*90*	593 609	*120*	11 789 891

Tabla 8.2. TCT de las secuencias generadas por el algoritmo ISA en las 120 instancias.

Por cómo se ha definido, RPD nunca puede ser positivo, ya que la secuencia final es siempre al menos tan buena como la inicial. Un mayor valor absoluto en el indicador representa una reducción más acentuada del TCT al seguir el algoritmo en cuestión.

Por ejemplo, para la heurística CFI, se muestran en la Tabla 8.3 los valores del TCT y del indicador de mejora relativa para cada una de las 120 instancias.

En la Tabla 8.3 se observa que, por lo general, aplicar la heurística de mejora CFI a la secuencia obtenida mediante el algoritmo ISA hace que se reduzca el TCT. Destaca la instancia número 6, en la que la heurística CFI hace que el valor de la función objetivo se reduzca en más de un 11%. Por otra parte, todas las instancias en las que $\mathrm{RPD} = 0.00$ son casos en los que CFI no consigue mejorar la secuencia inicial que aporta ISA.

ISA + CFI								
Instancia	TCT	RPD	*Instancia*	TCT	RPD	*Instancia*	TCT	RPD
1	15 674	-1.24	*41*	115 065	-2.12	*81*	570 200	-1.28
2	17 545	-4.13	*42*	113 827	0.00	*82*	572 966	-0.44
3	15 998	-2.59	*43*	108 379	-1.25	*83*	570 868	-0.50
4	17 970	-5.31	*44*	114 637	-2.32	*84*	575 480	-0.84
5	15 781	-4.98	*45*	118 171	-1.41	*85*	562 705	-0.20
6	15 626	-11.13	*46*	116 149	-0.65	*86*	571 252	0.00
7	15 930	-4.66	*47*	118 699	-1.77	*87*	572 473	-0.56
8	16 068	-1.17	*48*	116 933	-1.78	*88*	584 048	-1.09
9	16 385	-5.76	*49*	112 024	-2.33	*89*	574 629	-0.89
10	15 463	-3.04	*50*	116 519	-2.63	*90*	585 756	-1.32
11	25 205	-2.85	*51*	175 780	0.00	*91*	1 520 177	-3.09
12	26 804	-0.04	*52*	163 344	-1.35	*92*	1 530 470	-1.82
13	22 975	-3.39	*53*	161 056	-1.77	*93*	1 536 835	-1.88
14	22 622	-2.22	*54*	164 654	-1.26	*94*	1 496 185	-3.28
15	23 721	-1.56	*55*	170 161	0.00	*95*	1 528 339	0.00
16	22 785	-1.05	*56*	164 271	-0.71	*96*	1 525 817	-1.36
17	21 965	-7.19	*57*	169 630	-0.60	*97*	1 552 524	-2.77
18	24 282	-0.13	*58*	171 018	-0.15	*98*	1 544 485	-0.70
19	23 550	-1.92	*59*	167 103	-2.77	*99*	1 522 429	-1.13
20	24 954	-4.64	*60*	170 748	-1.80	*100*	1 538 367	-1.90
21	38 597	-2.95	*61*	320 218	0.00	*101*	2 024 311	0.00
22	37 930	-1.16	*62*	302 455	-4.94	*102*	2 052 846	-0.17
23	38 602	-0.59	*63*	299 423	-4.20	*103*	2 057 213	-0.83
24	38 802	-3.53	*64*	285 096	-6.12	*104*	2 080 928	0.00
25	39 571	-1.44	*65*	296 549	-6.44	*105*	2 088 352	0.00
26	38 716	-1.40	*66*	293 211	-8.40	*106*	2 083 017	-0.12
27	40 048	-0.93	*67*	305 351	-3.83	*107*	2 063 940	-2.15
28	37 774	-0.65	*68*	298 483	-5.98	*108*	2 072 675	-0.61
29	39 267	-2.99	*69*	312 283	-8.19	*109*	2 050 876	-0.51
30	38 338	-2.21	*70*	305 699	-5.73	*110*	2 048 037	-0.64
31	77 126	-1.51	*71*	418 583	-1.25	*111*	11 646 936	-0.22
32	84 418	-7.69	*72*	397 544	-2.54	*112*	11 909 296	0.00
33	79 449	-3.15	*73*	410 687	-1.33	*113*	11 626 667	0.00
34	84 622	-4.41	*74*	427 295	-2.20	*114*	11 801 140	-0.03
35	86 164	-1.27	*75*	404 143	-1.60	*115*	11 846 478	0.00
36	81 774	-5.62	*76*	400 052	-3.90	*116*	11 840 446	-0.06
37	79 860	-1.58	*77*	403 426	-0.94	*117*	11 758 172	0.00
38	82 741	-2.86	*78*	414 310	-1.22	*118*	11 798 447	0.00
39	77 092	-1.00	*79*	419 365	-1.12	*119*	11 763 442	-0.14
40	85 219	-3.09	*80*	431 535	-2.45	*120*	11 789 891	0.00

Tabla 8.3. TCT y RPD de las secuencias generadas por el algoritmo CFI en las 120 instancias.

Una vez se ha definido el indicador que refleja la eficacia de los algoritmos, no se debe ignorar el tiempo computacional empleado en llevarlos a cabo. Los tiempos se miden para comparar la eficiencia de las heurísticas, que se programan en lenguaje C usando el entorno de desarrollo *Code::Blocks* en un *Toshiba Satellite Pro* con las características que aparecen en la Figura 8.1.

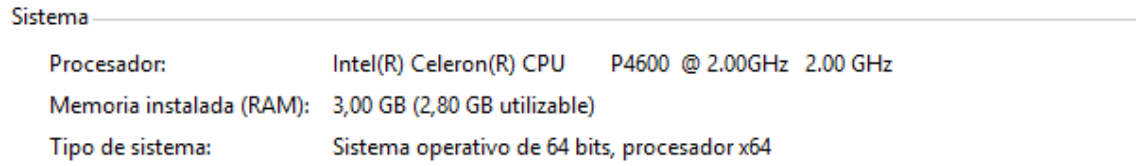

Figura 8.1. Características técnicas del ordenador en el que se han ejecutado los algoritmos.

Sin embargo, tratando los resultados como se muestran en la Tabla 8.5 sería muy difícil extraer conclusiones. Es por ello que se agrupan los resultados de las instancias con problemas de igual tamaño, de forma que se pueda apreciar más fácilmente cómo funciona cada algoritmo en función del mismo. En la Tabla 8.4 se presentan los resultados de dicha forma, indicándose el valor medio de ambos indicadores para las diez instancias de cada tamaño de problema. Nótese que cuando se hace referencia al valor medio del indicador RPD para una serie de instancias, se le denomina ARPD (*Average Relative Percentage Deviation*).

ISA + CFI		
Instancias	**ARPD**	**Tiempo**
1-10	**-4.40**	0.009
11-20	**-2.50**	0.012
21-30	**-1.79**	0.012
31-40	**-3.22**	0.079
41-50	**-1.63**	0.106
51-60	**-1.04**	0.107
61-70	**-5.38**	0.881
71-80	**-1.86**	1.017
81-90	**-0.71**	1.120
91-100	**-1.79**	14.722
101-110	**-0.50**	15.531
111-120	**-0.05**	580.325
TOTAL	**-2.07**	51.160

Tabla 8.4. ARPD y tiempo computacional medio de la heurística CFI agrupado por instancias.

ISA + CFI								
Instancia	RPD	Tiempo	*Instancia*	RPD	Tiempo	*Instancia*	RPD	Tiempo
1	-1.24	0.000	*41*	-2.12	0.109	*81*	-1.28	1.109
2	-4.13	0.000	*42*	0.00	0.093	*82*	-0.44	1.125
3	-2.59	0.015	*43*	-1.25	0.093	*83*	-0.50	1.109
4	-5.31	0.000	*44*	-2.32	0.078	*84*	-0.84	1.093
5	-4.98	0.015	*45*	-1.41	0.109	*85*	-0.20	1.156
6	-11.13	0.015	*46*	-0.65	0.109	*86*	0.00	1.109
7	-4.66	0.015	*47*	-1.77	0.187	*87*	-0.56	1.140
8	-1.17	0.000	*48*	-1.78	0.078	*88*	-1.09	1.125
9	-5.76	0.015	*49*	-2.33	0.109	*89*	-0.89	1.125
10	-3.04	0.015	*50*	-2.63	0.093	*90*	-1.32	1.109
11	-2.85	0.015	*51*	0.00	0.125	*91*	-3.09	14.734
12	-0.04	0.015	*52*	-1.35	0.109	*92*	-1.82	14.750
13	-3.39	0.015	*53*	-1.77	0.093	*93*	-1.88	14.657
14	-2.22	0.000	*54*	-1.26	0.109	*94*	-3.28	14.688
15	-1.56	0.015	*55*	0.00	0.109	*95*	0.00	14.735
16	-1.05	0.015	*56*	-0.71	0.093	*96*	-1.36	14.641
17	-7.19	0.031	*57*	-0.60	0.109	*97*	-2.77	14.703
18	-0.13	0.000	*58*	-0.15	0.124	*98*	-0.70	14.766
19	-1.92	0.015	*59*	-2.77	0.093	*99*	-1.13	14.719
20	-4.64	0.000	*60*	-1.80	0.109	*100*	-1.90	14.828
21	-2.95	0.031	*61*	0.00	1.015	*101*	0.00	15.391
22	-1.16	0.015	*62*	-4.94	0.097	*102*	-0.17	15.485
23	-0.59	0.015	*63*	-4.20	0.984	*103*	-0.83	15.531
24	-3.53	0.015	*64*	-6.12	0.937	*104*	0.00	15.563
25	-1.44	0.000	*65*	-6.44	0.968	*105*	0.00	15.563
26	-1.40	0.015	*66*	-8.40	0.984	*106*	-0.12	15.578
27	-0.93	0.015	*67*	-3.83	0.953	*107*	-2.15	15.516
28	-0.65	0.000	*68*	-5.98	0.984	*108*	-0.61	15.625
29	-2.99	0.000	*69*	-8.19	0.953	*109*	-0.51	15.578
30	-2.21	0.015	*70*	-5.73	0.937	*110*	-0.64	15.484
31	-1.51	0.062	*71*	-1.25	1.000	*111*	-0.22	580.632
32	-7.69	0.078	*72*	-2.54	1.046	*112*	0.00	581.585
33	-3.15	0.062	*73*	-1.33	1.046	*113*	0.00	583.565
34	-4.41	0.062	*74*	-2.20	1.000	*114*	-0.03	580.269
35	-1.27	0.109	*75*	-1.60	0.984	*115*	0.00	578.628
36	-5.62	0.078	*76*	-3.90	1.046	*116*	-0.06	579.593
37	-1.58	0.078	*77*	-0.94	1.031	*117*	0.00	577.099
38	-2.86	0.093	*78*	-1.22	1.015	*118*	0.00	582.775
39	-1.00	0.078	*79*	-1.12	1.000	*119*	-0.14	578.101
40	-3.09	0.093	*80*	-2.45	1.000	*120*	0.00	581.003

Tabla 8.5. RPD y tiempo computacional empleado por la heurística CFI en las 120 instancias.

Como se podría esperar, en las instancias de mayor tamaño se requiere un tiempo computacional superior, sufriendo éste un crecimiento exponencial proporcional al número de trabajos a procesar. Por otra parte, el número de máquinas tiene una influencia mucho menor, como se aprecia, por ejemplo, en los resultados de las instancias 91-100 (de 200 trabajos y 10 máquinas) y 101-110 (de 200 trabajos y 20 máquinas).

Sobre la mejora relativa con respecto a la secuencia ISA, se observa una clara relación con el número de máquinas en el taller, pues en instancias donde hay más máquinas no se mejora tanto la secuencia inicial como cuando éste es menor, sucediendo esto para instancias con distintas cantidades de trabajos a procesar. Por tanto, se demuestra que el algoritmo ISA original aporta mejores soluciones cuando el número de máquinas en el taller es mayor.

A continuación, en la Tabla 8.6 se muestran los resultados obtenidos con la heurística CFI interrumpido, un proceso con el que, como ya se explicó al final del capítulo 4, se logran exactamente los mismos valores del TCT pero empleando un menor tiempo computacional que con CFI, pues se da por finalizado cuando se encuentra el óptimo local.

ISA + CFI interrumpido		
Instancias	**ARPD**	**Tiempo**
1-10	**-4.40**	0.011
11-20	**-2.50**	0.018
21-30	**-1.79**	0.017
31-40	**-3.22**	0.067
41-50	**-1.63**	0.067
51-60	**-1.04**	0.098
61-70	**-5.38**	0.732
71-80	**-1.86**	0.628
81-90	**-0.71**	0.642
91-100	**-1.79**	8.734
101-110	**-0.50**	5.657
111-120	**-0.05**	136.391
TOTAL	**-2.07**	12.755

Tabla 8.6. ARPD y tiempo computacional medio de la heurística CFI interrumpido agrupado por instancias.

Evidentemente, la mejora media del TCT respecto al de la secuencia inicial se mantiene en el 2.07 %, pero el tiempo computacional empleado se reduce hasta la cuarta parte de lo que era con la heurística CFI original.

8.2. Técnicas de búsqueda local

Con el fin de presentar los datos de forma clara, se adjunta directamente la Tabla 8.7, en la que se observa la mejora relativa y el tiempo computacional medio para las 120 instancias al aplicar a la secuencia inicial cada una de las diez técnicas de búsqueda local explicadas en el capítulo 5. En el anexo B se pueden consultar los resultados de cada técnica por separado para los distintos tamaños de problema.

ISA + Técnicas de búsqueda local						
	swap		*ins*		*insfor*	
	ARPD	**Tiempo**	**ARPD**	**Tiempo**	**ARPD**	**Tiempo**
BEST	**-0.33**	1.050	**-1.45**	7.253	**-0.56**	1.480
FIRST	**-0.36**	0.712	**-1.47**	11.220	**-0.52**	2.308
FIRST BEST	**-0.34**	0.582	**-1.43**	1.273	**-0.54**	0.621

NEH LOC	
ARPD	**Tiempo**
-1.01	0.300

Tabla 8.7. ARPD y tiempo computacional medio de las diez técnicas de búsqueda local consideradas.

Con la tabla anterior, queda demostrado que los diferentes criterios de actuación al encontrar una secuencia vecina mejor (BEST, FIRST, FIRST BEST) no influyen en el valor de la función objetivo de la secuencia final, pero sí en el tiempo que se tarda en hallarla, siendo las FIRST BEST claramente las más rápidas en todos los casos.

Por otra parte, las técnicas de inserción completa de trabajos parecen ser las que mayores mejoras consiguen. Es por ello que siempre se incluye la técnica insFB en los algoritmos presentados a partir del capítulo 6. También se considera la swapFB al tratarse de una técnica de búsqueda diferente que puede hallar otros óptimos locales sin ralentizar excesivamente el proceso.

Por su parte, la NEH_{LOC} demuestra ser una heurística extremadamente rápida y que también logra grandes mejoras, por lo que es el tercero de los diez métodos de búsquedas locales que se usan para construir los algoritmos propuestos, no usándose ninguno de los siete restantes.

8.3. Heurísticas propuestas

En primer lugar, se comparan los resultados obtenidos con las dos primeras heurísticas propuestas, que obviamente emplean el mismo tiempo computacional, pues se componen de las mismas técnicas, sólo que aplicadas en distinto orden.

CML - I			CML - II		
Instancias	ARPD	Tiempo	*Instancias*	ARPD	Tiempo
1-10	**-3.68**	0.008	*1-10*	**-3.67**	0.000
11-20	**-2.08**	0.018	*11-20*	**-2.41**	0.003
21-30	**-1.43**	0.021	*21-30*	**-1.71**	0.005
31-40	**-2.19**	0.032	*31-40*	**-2.43**	0.009
41-50	**-1.34**	0.039	*41-50*	**-1.38**	0.015
51-60	**-0.87**	0.054	*51-60*	**-0.91**	0.025
61-70	**-3.34**	0.112	*61-70*	**-3.71**	0.089
71-80	**-1.57**	0.132	*71-80*	**-1.71**	0.092
81-90	**-0.71**	0.179	*81-90*	**-0.79**	0.145
91-100	**-1.16**	0.818	*91-100*	**-1.32**	0.848
101-110	**-0.66**	1.073	*101-110*	**-0.79**	0.951
111-120	**-0.36**	13.138	*111-120*	**-0.37**	13.231
TOTAL	**-1.62**	1.302	*TOTAL*	**-1.77**	1.284

Tabla 8.8. ARPD y tiempo computacional medio de las heurísticas propuestas CML-I y CML-II.

Se observa en la Tabla 8.8 que para la mayoría de tamaños de problema, y por tanto también en término medio, la segunda heurística consigue resultados claramente mejores que la primera, por lo que se demuestra que la técnica NEH_{LOC} debe aplicarse antes que la insFB, información que se tiene en cuenta al construir los algoritmos posteriores.

Sabiendo que la NEH_{LOC} ha de aplicarse en primer lugar, en la Tabla 8.9 se estudia en qué orden es mejor aplicar las otras dos técnicas consideradas: insFB y swapFB.

De nuevo, el tiempo computacional empleado por ambas heurísticas es idéntico, y se demuestra que es ligeramente más favorable aplicar la insFB antes que la swapFB, de ahí que en todos los algoritmos posteriores la swapFB sea la última técnica que se aplique.

La quinta heurística propuesta necesita más tiempo para ejecutarse en comparación a las anteriores porque aplica en dos ocasiones la técnica NEH_{LOC}, cuatro veces la insFB y una la swapFB. Aunque atendiendo a la Tabla 8.10, los resultados mejoran con respecto a las anteriores.

CML - III		
Instancias	ARPD	Tiempo
1-10	-3.76	0.008
11-20	-2.41	0.025
21-30	-1.78	0.026
31-40	-2.76	0.042
41-50	-1.48	0.049
51-60	-0.92	0.056
61-70	-4.13	0.153
71-80	-1.74	0.134
81-90	-0.80	0.206
91-100	-1.42	1.082
101-110	-0.84	1.154
111-120	-0.40	15.880
TOTAL	-1.87	1.568

CML - IV		
Instancias	ARPD	Tiempo
1-10	-3.93	0.006
11-20	-1.82	0.008
21-30	-1.78	0.009
31-40	-2.55	0.017
41-50	-1.45	0.021
51-60	-0.96	0.032
61-70	-4.13	0.117
71-80	-1.73	0.143
81-90	-0.83	0.168
91-100	-1.40	1.079
101-110	-0.85	1.171
111-120	-0.38	16.074
TOTAL	-1.82	1.570

Tabla 8.9. ARPD y tiempo computacional medio de las heurísticas propuestas CML-III y CML-IV.

CML - V		
Instancias	ARPD	Tiempo
1-10	-4.08	0.008
11-20	-2.43	0.029
21-30	-1.75	0.022
31-40	-2.90	0.064
41-50	-1.56	0.070
51-60	-1.12	0.076
61-70	-4.64	0.303
71-80	-2.07	0.303
81-90	-0.96	0.351
91-100	-1.74	2.494
101-110	-0.94	2.683
111-120	-0.40	41.601
TOTAL	-2.05	4.000

Tabla 8.10. ARPD y tiempo computacional medio de la heurística propuesta CML-V.

8.4. Metaheurísticas propuestas

Se muestran en la Tabla 8.11 los resultados obtenidos por el algoritmo MH-I realizando 5, 10, 50 y 100 iteraciones.

MH - I	x=5	
Instancias	ARPD	Tiempo
1-10	-4.60	0.040
11-20	-3.09	0.056
21-30	-2.12	0.064
31-40	-3.45	0.082
41-50	-1.71	0.117
51-60	-1.04	0.122
61-70	-4.42	0.329
71-80	-2.16	0.375
81-90	-0.85	0.525
91-100	-1.61	2.495
101-110	-0.82	2.831
111-120	-0.41	39.285
TOTAL	-2.19	3.860

MH - I	x=10	
Instancias	ARPD	Tiempo
1-10	-4.52	0.028
11-20	-3.13	0.045
21-30	-2.17	0.064
31-40	-3.37	0.097
41-50	-1.80	0.120
51-60	-1.23	0.185
61-70	-4.72	0.382
71-80	-2.15	0.470
81-90	-1.01	0.706
91-100	-1.54	2.890
101-110	-0.94	3.742
111-120	-0.41	46.193
TOTAL	-2.25	4.577

MH - I	x=50	
Instancias	ARPD	Tiempo
1-10	-4.69	0.039
11-20	-3.40	0.067
21-30	-2.41	0.123
31-40	-3.37	0.168
41-50	-2.18	0.234
51-60	-1.35	0.526
61-70	-5.17	0.743
71-80	-2.29	1.079
81-90	-1.10	2.131
91-100	-1.90	6.045
101-110	-1.07	10.209
111-120	-0.42	104.430
TOTAL	-2.45	10.483

MH - I	x=100	
Instancias	ARPD	Tiempo
1-10	-4.73	0.031
11-20	-3.41	0.062
21-30	-2.46	0.156
31-40	-3.43	0.215
41-50	-2.23	0.395
51-60	-1.51	0.932
61-70	-5.14	1.182
71-80	-2.35	1.870
81-90	-1.14	3.960
91-100	-1.99	10.246
101-110	-1.07	18.549
111-120	-0.42	175.353
TOTAL	-2.49	17.746

Tabla 8.11. ARPD y tiempo computacional medio de la metaheurística propuesta MH-I para x=5, x=10, x=50, x=100.

Como se podía deducir, aplicar un mayor número de iteraciones hace que el tiempo computacional empleado crezca. Y también se logran mejores resultados, algo que realmente no tiene por qué ser siempre así, ya que éstos dependen de las secuencias iniciales generadas en cada caso.

La principal debilidad de esta heurística es que descarta muchas secuencias que podrían aportar buenos resultados una vez se les aplica las técnicas insFB y swapFB, pero que no sobreviven al paso 15 del proceso (Figura 7.2), ya que tras él sólo se conserva una secuencia, de la cual no hay ninguna certeza de que vaya a mejorar considerablemente tras los pasos 16 y 17.

La heurística MH-II no descarta tantas secuencias de forma prematura, y teóricamente asegura una secuencia final al menos igual de buena que la obtenida con MH-I a igual número de iteraciones (algo que realmente no tiene por qué cumplirse siempre debido a la aleatoriedad existente al generar las secuencias iniciales). Sin embargo, emplea un tiempo considerablemente mayor, pues la técnica insFB se aplica tantas veces como secuencias iniciales se generen, mientras que en la MH-I apenas se aplicaba en una ocasión.

MH - II	x=5	
Instancias	ARPD	Tiempo
1-10	-4.83	0.053
11-20	-3.13	0.048
21-30	-2.38	0.068
31-40	-3.45	0.103
41-50	-1.79	0.140
51-60	-1.08	0.172
61-70	-4.67	0.809
71-80	-2.28	0.795
81-90	-0.90	0.901
91-100	-1.61	6.370
101-110	-0.86	6.311
111-120	-0.41	111.565
TOTAL	-2.28	10.611

MH - II	x=10	
Instancias	ARPD	Tiempo
1-10	-4.89	0.090
11-20	-3.29	0.037
21-30	-2.44	0.060
31-40	-3.54	0.173
41-50	-1.95	0.190
51-60	-1.24	0.243
61-70	-5.07	1.398
71-80	-2.29	1.354
81-90	-1.07	1.489
91-100	-1.67	11.569
101-110	-0.97	11.806
111-120	-0.41	209.324
TOTAL	-2.40	19.811

Tabla 8.12. ARPD y tiempo computacional medio de la metaheurística propuesta MH-II para x=5 y x=10.

El algoritmo MH-III, al consistir en un híbrido entre los dos anteriores, consigue resultados no tan buenos como los de MH-II a igualdad de secuencias iniciales distintas generadas, pero en un tiempo claramente menor.

MH - III	x = 20 ; y = 2	
Instancias	% Δ TCT ISA	Tiempo
1-10	-4.77	0.009
11-20	-3.21	0.018
21-30	-2.41	0.040
31-40	-3.51	0.082
41-50	-2.01	0.122
51-60	-1.37	0.232
61-70	-5.05	0.545
71-80	-2.31	0.667
81-90	-1.14	1.111
91-100	-1.70	4.509
101-110	-1.00	6.181
111-120	-0.41	77.848
TOTAL	-2.41	7.614

MH - III	x = 30 ; y = 5	
Instancias	% Δ TCT ISA	Tiempo
1-10	-5.04	0.023
11-20	-3.40	0.060
21-30	-2.41	0.081
31-40	-3.73	0.171
41-50	-2.13	0.222
51-60	-1.45	0.389
61-70	-5.18	1.014
71-80	-2.38	1.142
81-90	-1.23	1.781
91-100	-1.95	8.183
101-110	-1.03	10.280
111-120	-0.41	144.738
TOTAL	-2.53	14.007

MH - III	x = 50 ; y = 5	
Instancias	% Δ TCT ISA	Tiempo
1-10	-4.98	0.020
11-20	-3.44	0.040
21-30	-2.47	0.090
31-40	-3.63	0.164
41-50	-2.36	0.256
51-60	-1.42	0.529
61-70	-5.31	1.186
71-80	-2.39	1.428
81-90	-1.19	2.453
91-100	-1.95	9.470
101-110	-1.09	13.349
111-120	-0.42	172.098
TOTAL	-2.55	16.757

MH - III	x = 100 ; y = 10	
Instancias	% Δ TCT ISA	Tiempo
1-10	-5.04	0.039
11-20	-3.48	0.070
21-30	-2.48	0.165
31-40	-3.82	0.310
41-50	-2.48	0.482
51-60	-1.65	1.023
61-70	-5.38	2.167
71-80	-2.46	2.696
81-90	-1.36	4.750
91-100	-2.04	18.014
101-110	-1.15	25.535
111-120	-0.42	329.029
TOTAL	-2.65	32.023

Tabla 8.13. ARPD y tiempo computacional medio de la metaheurística propuesta MH-III para cuatro combinaciones de parámetros distintas.

Observando las Tablas 8.12 y 8.13, se aprecia lo importante del hecho de que en la MH-III sólo se intente profundizar en las iteraciones que son más prometedoras tras los primeros pasos, pues lo que ralentiza tanto a la MH-II es que se aplican las técnicas insFB y swapFB x veces cada una, mientras que en la MH-III sólo en y ocasiones.

8.5. Parametrización de la metaheurística propuesta MH-III

Al ser un algoritmo doblemente parametrizable, se plantean numerosas combinaciones de los dos parámetros: x, z. No obstante, se da el caso de que distintas combinaciones resulten en el mismo problema, ya que, como se observa en la Tabla 8.14, cualquier valor de z menor de 20 cuando $x = 5$ hace que sólo se cuente con la mejor de las cinco secuencias obtenidas para la parte final del algoritmo. Por ello, a pesar de plantear hasta 42 posibles combinaciones, realmente sólo se tienen 34 distintas.

Valor de y		z					
		5	10	15	20	25	30
x	5	1				2	
	10	1		2		3	
	15	1	2	3		4	5
	20	1	2	3	4	5	6
	30	2	3	5	6	8	9
	50	3	5	8	10	13	15
	100	5	10	15	20	25	30

Tabla 8.14. Valores propuestos para los parámetros del problema: *x*, *z*; así como el valor de *y* en cada caso.

En las Tablas 8.15 y 8.16 se observa el ARPD y tiempo computacional empleado por la metaheurística MH-III para cada uno de los 34 casos planteados.

ARPD		z					
		5	10	15	20	25	30
x	5	-2.19				-2.24	
	10	-2.25		-2.32		-2.35	
	15	-2.29	-2.37	-2.40		-2.43	-2.45
	20	-2.33	-2.41	-2.44	-2.46	-2.49	-2.50
	30	-2.43	-2.47	-2.53	-2.53	-2.56	-2.56
	50	-2.51	-2.55	-2.58	-2.60	-2.61	-2.61
	100	-2.61	-2.65	-2.66	-2.67	-2.68	-2.68

Tabla 8.15. ARPD de la metaheurística propuesta MH-III para los 34 casos planteados según *x, z.*

Tiempo		z					
		5	10	15	20	25	30
x	5	3.9				5.5	
	10	4.6		6.2		7.7	
	15	5.3	7.0	8.4		10.0	11.7
	20	6.1	7.6	9.2	11.0	12.5	14.4
	30	9.0	10.6	14.0	15.6	18.6	20.4
	50	13.4	16.8	21.6	24.7	29.3	33.0
	100	24.0	32.0	41.4	47.8	56.1	66.3

Tabla 8.16. Tiempo computacional medio empleado por la metaheurística propuesta MH-III para los 34 casos planteados según *x, z.*

Como ya se comentó en el capítulo 7 al definir el algoritmo, se demuestra que aumentar el valor de los parámetros (generar más secuencias iniciales distintas y/o guardar una mayor cantidad de ellas de cara a la parte final del algoritmo) incrementa notablemente el tiempo computacional que emplea la metaheurística, aunque también logra mejorar los resultados obtenidos. Esta relación se plasma en la Figura 8.2, en la que se representan los resultados de los 34 problemas planteados.

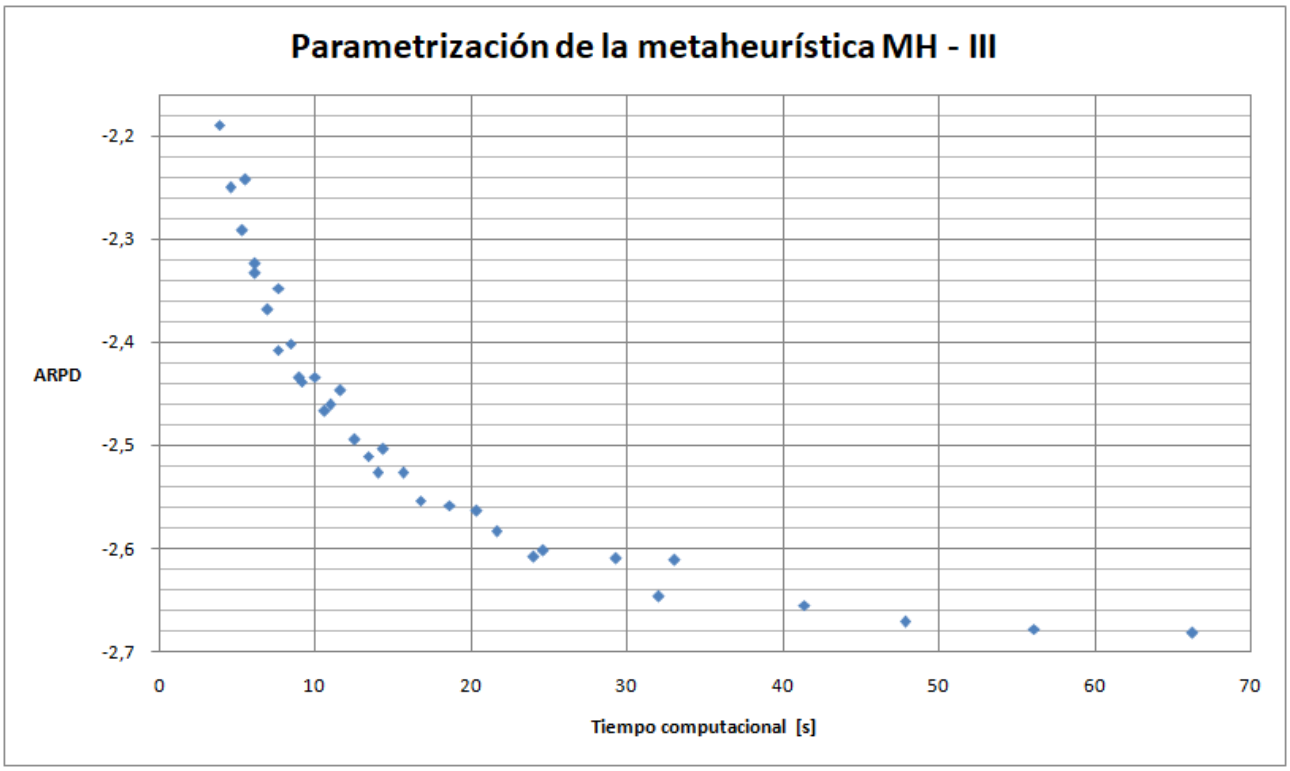

Figura 8.2. Representación gráfica del ARPD y tiempo computacional de la metaheurística propuesta MH-III para los 34 problemas planteados.

Se observa que es relativamente fácil mejorar los resultados de los problemas más sencillos, ya que cuando los parámetros toman un valor pequeño, generar más secuencias (aumentar la x) y/o no descartar tantas antes de tiempo (aumentar la z) hace que haya más posibilidades de llegar a distintos óptimos locales sin que el tiempo computacional se vea excesivamente afectado. Sin embargo, a medida que se van aumentando los parámetros resulta más y más difícil seguir mejorando, creciendo enormemente el tiempo computacional empleado apenas logrando mejoras ínfimas.

Se adjunta en la Figura 8.3 la frontera eficiente de todos los casos planteados de la heurística MH-III excepto los cuatro que emplean un mayor tiempo computacional (correspondientes a problemas con $x = 100$; $z = 15, 20, 25, 30$), que no se incluyen en la gráfica con el fin de facilitar su interpretación. No obstante, los puntos omitidos forman parte de la frontera eficiente, pues no hay ningún otro caso que logre un mayor ARPD en menos tiempo.

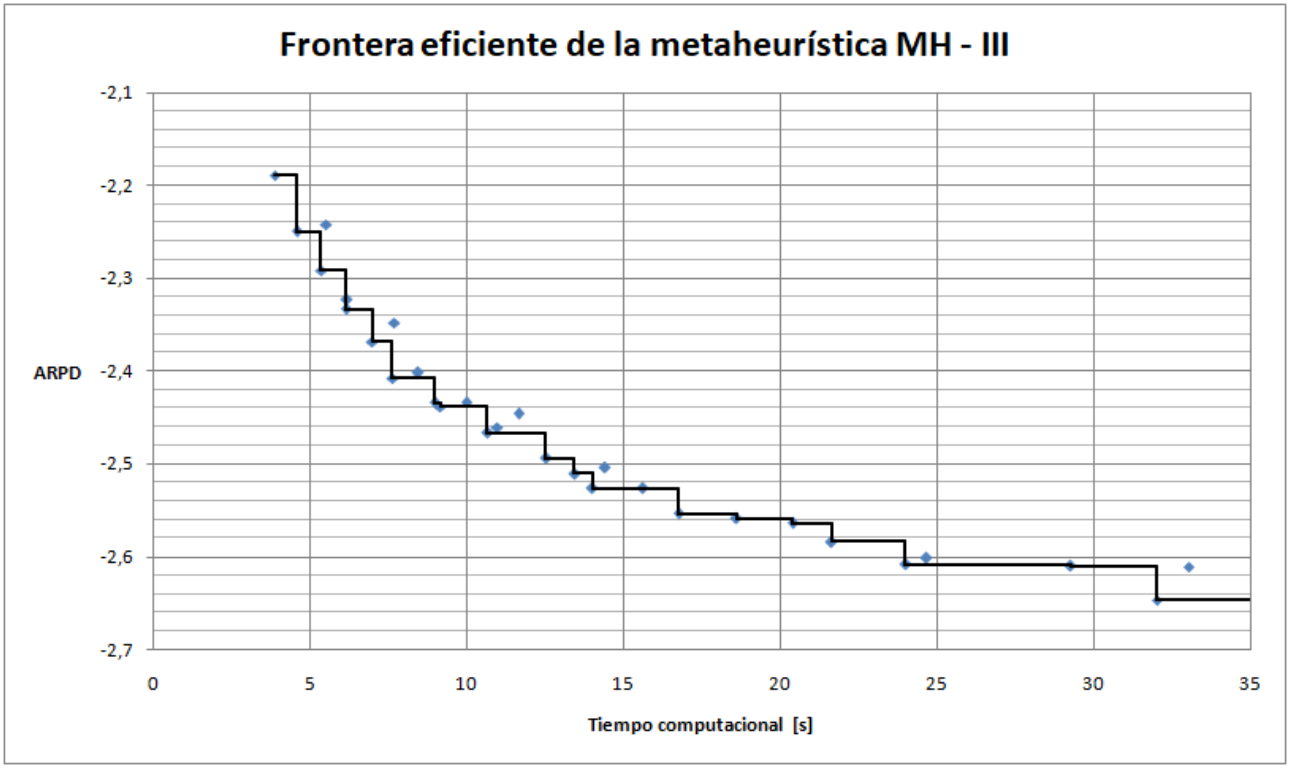

Figura 8.3. Representación gráfica de la frontera eficiente de la metaheurística propuesta MH-III para los 34 problemas planteados.

Es posible representar la frontera eficiente aproximada de la metaheurística uniendo los vértices de la frontera eficiente real, ya que ésta se obtuvo solamente con los valores de los parámetros que se consideraron. Sin embargo, en caso de realizar una parametrización completa (algo irreal e innecesario dado el volumen de información), la frontera eficiente resultante sería más parecida a la representada en la Figura 8.4.

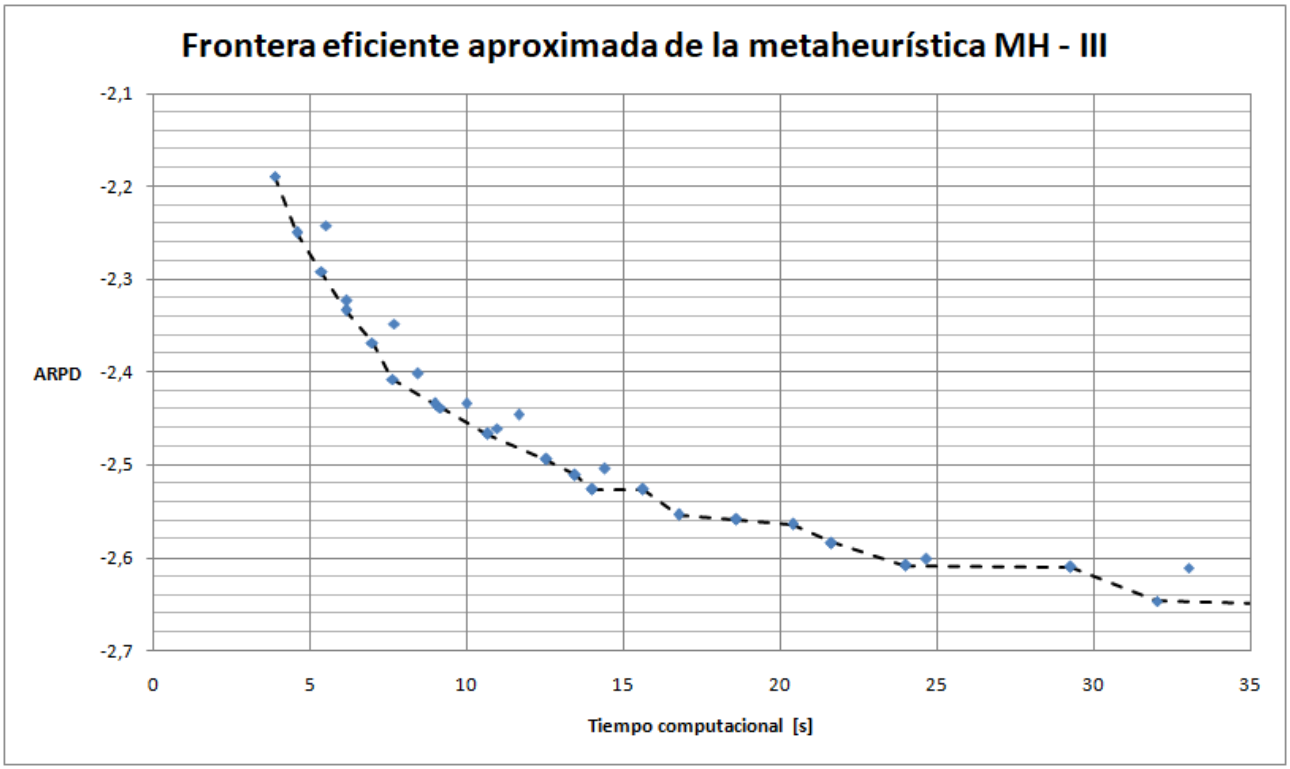

Figura 8.4. Representación gráfica de la frontera eficiente aproximada de la metaheurística propuesta MH-III para los 34 problemas planteados.

8.6. Comparación general de algoritmos

Una vez se obtienen los resultados logrados por todos los algoritmos, así como el tiempo computacional que emplean, se comparan entre ellos con el fin de hallar cuáles forman la frontera eficiente general. Se consideran las metaheurísticas MH-I y MH-II con los valores de los parámetros que aparecen en la sección 8.4, así como la frontera eficiente aproximada de la MH-III que se muestra en la Figura 8.4. Asimismo, también se incluye en la Tabla 8.17 muestran los resultados que aportan las cinco heurísticas propuestas en el capítulo 6, y también el algoritmo referencia en la literatura.

Algoritmos propuestos (excepto MH-III)		
Algoritmo	**ARPD**	**Tiempo**
CFI interrumpido	**-2.07**	12.755
CML - I	**-1.62**	1.302
CML - II	**-1.77**	1.284
CML - III	**-1.87**	1.568
CML - IV	**-1.82**	1.570
CML - V	**-2.05**	4.000
MH - I (x = 5)	**-2.19**	3.860
MH - I (x = 10)	**-2.25**	4.577
MH - I (x = 50)	**-2.45**	10.483
MH - I (x = 100)	**-2.49**	17.292
MH - II (x = 5)	**-2.28**	10.611
MH - II (x = 10)	**-2.40**	19.811

Tabla 8.17. ARPD y tiempo computacional de los algoritmos propuestos que aparecen en la Figura 8.5.

Tras representar los resultados de los distintos algoritmos en la Figura 8.5, la primera conclusión que se puede extraer de ella es que los algoritmos propuestos mejoran a la heurística referencia en términos de resultados obtenidos y/o tiempo computacional empleado.

Las heurísticas CML-II y CML-III se consideran eficientes porque en un tiempo muy reducido son capaces de encontrar resultados bastante aceptables, y porque todos los algoritmos que logran mejores resultados que ellas requieren de un tiempo computacional considerablemente superior. Por otra parte, las heurísticas CML-I y CML-IV no son eficientes, porque empleando el mismo tiempo computacional que CML-II y CML-III respectivamente, no consiguen obtener tan buenos resultados.

La heurística CML-V, si bien sería eficiente en caso de considerar sólo las heurísticas, deja de serlo al considerar la metaheurística MH-I, pues cuando se aplica con el parámetro $x = 5$ logra mejores resultados que la anterior en un tiempo menor.

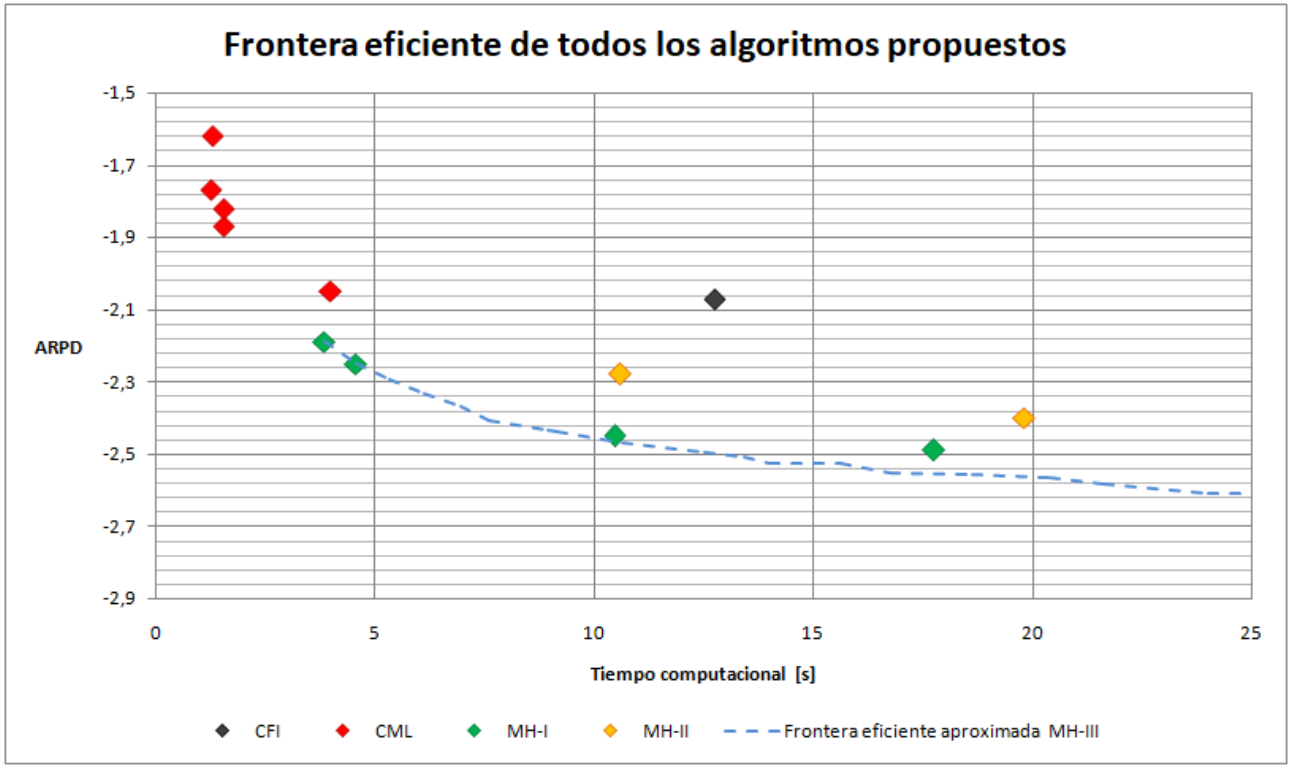

Figura 8.5. Representación gráfica del ARPD y tiempo computacional del algoritmo referencia, de los propuestos, y comparación ante la frontera eficiente aproximada de la metaheurística propuesta MH-III.

MH-II claramente no es eficiente frente a MH-I y MH-III, que consiguen mejores resultados incluso en menos tiempo. Como ya se comentó previamente, a MH-II le lastra el hecho de que no se descarte ninguna secuencia hasta el final del proceso, lo que dispara el tiempo que necesita para ejecutarse.

Por último, se observa que, aunque aumentando el valor del parámetro x se consigue que la metaheurística MH-I dé mejores resultados, no es eficiente frente a la MH-III, ya que cuando sus parámetros hacen que el problema tarde el mismo tiempo en resolverse, los resultados que da ésta última son mejores. Esto demuestra que jugando con los dos parámetros de MH-III adecuadamente, aumentando también ligeramente la z a medida que se hace crecer la x, se consigue llegar a resultados para los que usando MH-I (donde no existe el parámetro z) se necesitaría emplear muchísimo más tiempo.

En la Tabla 8.18 se adjuntan los resultados que se obtienen con la metaheurística MH-III cuando los valores de los parámetros son $x = 20$, $z = 25$, es decir, $y = 5$. Como se podía ver en la Tabla 8.15, éstos no son los mejores que se han obtenido, pero se escogen esos valores porque así el tiempo computacional empleado es similar al de la heurística referencia, de forma que la comparación de sus ARPD sea más representativa.

ISA + CFI interrumpido			MH - III	x = 20 ; y = 5		
Instancias	ARPD	Tiempo	*Instancias*	ARPD	Tiempo	Diferencia
1-10	-4.40	0.011	*1-10*	-5.03	0.020	-0.63
11-20	-2.50	0.018	*11-20*	-3.33	0.043	-0.83
21-30	-1.79	0.017	*21-30*	-2.41	0.092	-0.62
31-40	-3.22	0.067	*31-40*	-3.67	0.198	-0.45
41-50	-1.63	0.067	*41-50*	-2.07	0.217	-0.44
51-60	-1.04	0.098	*51-60*	-1.45	0.334	-0.41
61-70	-5.38	0.732	*61-70*	-5.23	0.964	0.15
71-80	-1.86	0.628	*71-80*	-2.36	1.011	-0.50
81-90	-0.71	0.642	*81-90*	-1.20	1.451	-0.49
91-100	-1.79	8.734	*91-100*	-1.77	7.592	0.02
101-110	-0.50	5.657	*101-110*	-1.01	8.850	-0.51
111-120	-0.05	136.391	*111-120*	-0.41	129.469	-0.36
TOTAL	-2.07	12.755	*TOTAL*	-2.49	12.520	

Tabla 8.18. ARPD y tiempo computacional empleado por la heurística referencia y MH-III (x=20, y=5) agrupado por instancias.

Se observa como para la mayoría de tamaños de problema MH-III mejora holgadamente a la heurística referencia, siendo el valor absoluto de su ARPD 0,42 puntos porcentuales mayor empleando el mismo tiempo computacional. No es éste el mejor valor logrado del mismo, pues se ha conseguido llegar hasta -2,68. Sin embargo, para ello el tiempo computacional es más de cinco veces superior, y como ya se ha explicado, mejoras adicionales implicarían un aumento exponencial del tiempo computacional, que sólo sería asumible en caso de que fuese vital reducir el valor de la función objetivo aunque fuese mínimamente.

En la Tabla 8.19 se muestra la misma comparación que en la 8.18, sólo que en esta ocasión pudiendo comparar el TCT obtenido con ambos métodos en cada una de las 120 instancias. Promediando la mejora porcentual de los resultados conseguidos por esta particularización de la MH-III con respecto a los de la heurística referencia, se concluye que éstos son un 0,43% mejores.

No obstante, aunque en estas dos tablas se aprecia que generalmente se obtienen mejores resultados aplicando la heurística MH-III, es necesario justificar mediante un análisis estadístico que esa diferencia en los valores del TCT se debe realmente al uso de dicho algoritmo. Dicho análisis, así como los resultados y conclusiones extraídas del mismo, se adjunta en el anexo A.

Comparación de resultados por instancia: mejora de MH-III (x = 20 ; y = 5) sobre ISA + CFI interrumpido

Instancia	ISA + CFI	MH-III	Mejora	*Instancia*	ISA + CFI	MH-III	Mejora	*Instancia*	ISA + CFI	MH-III	Mejora
1	15 674	15 674	0.00%	*41*	115 065	115 592	0.46%	*81*	570 200	567 612	-0.45%
2	17 545	17 431	-0.65%	*42*	113 827	113 827	0.00%	*82*	572 966	573 942	0.17%
3	15 998	15 892	-0.66%	*43*	108 379	107 307	-0.99%	*83*	570 868	569 140	-0.30%
4	17 970	18 076	0.59%	*44*	114 637	114 696	0.05%	*84*	575 480	567 759	-1.34%
5	15 781	15 411	-2.34%	*45*	118 171	117 183	-0.84%	*85*	562 705	556 818	-1.05%
6	15 626	15 501	-0.80%	*46*	116 149	114 612	-1.32%	*86*	571 252	567 237	-0.70%
7	15 930	15 746	-1.16%	*47*	118 699	117 915	-0.66%	*87*	572 473	574 365	0.33%
8	16 068	15 955	-0.70%	*48*	116 933	117 181	0.21%	*88*	584 048	584 475	0.07%
9	16 385	16 385	0.00%	*49*	112 024	112 204	0.16%	*89*	574 629	566 560	-1.40%
10	15 463	15 329	-0.87%	*50*	116 519	114 642	-1.61%	*90*	585 756	584 266	-0.25%
11	25 205	25 205	0.00%	*51*	175 780	174 259	-0.87%	*91*	1 520 177	1 534 523	0.94%
12	26 804	26 575	-0.85%	*52*	163 344	163 619	0.17%	*92*	1 530 470	1 528 400	-0.14%
13	22 975	22 961	-0.06%	*53*	161 056	160 771	-0.18%	*93*	1 536 835	1 539 558	0.18%
14	22 622	22 461	-0.71%	*54*	164 654	163 021	-0.99%	*94*	1 496 185	1 503 285	0.47%
15	23 721	23 269	-1.91%	*55*	170 161	169 137	-0.60%	*95*	1 528 339	1 523 730	-0.30%
16	22 785	22 185	-2.63%	*56*	164 271	162 832	-0.88%	*96*	1 525 817	1 518 629	-0.47%
17	21 965	21 939	-0.12%	*57*	169 630	169 859	0.13%	*97*	1 552 524	1 562 306	0.63%
18	24 282	24 158	-0.51%	*58*	171 018	169 921	-0.64%	*98*	1 544 485	1 538 158	-0.41%
19	23 550	23 501	-0.21%	*59*	167 103	167 626	0.31%	*99*	1 522 429	1 517 392	-0.33%
20	24 954	24 597	-1.43%	*60*	170 748	169 756	-0.58%	*100*	1 538 367	1 533 893	-0.29%
21	38 597	38 597	0.00%	*61*	320 218	314 185	-1.88%	*101*	2 024 311	2 016 176	-0.40%
22	37 930	37 729	-0.53%	*62*	302 455	305 474	1.00%	*102*	2 052 846	2 046 931	-0.29%
23	38 602	38 350	-0.65%	*63*	299 423	301 771	0.78%	*103*	2 057 213	2 052 857	-0.21%
24	38 802	38 802	0.00%	*64*	285 096	288 312	1.13%	*104*	2 080 928	2 063 129	-0.86%
25	39 571	39 211	-0.91%	*65*	296 549	297 375	0.28%	*105*	2 088 352	2 060 845	-1.32%
26	38 716	38 597	-0.31%	*66*	293 211	295 661	0.84%	*106*	2 083 017	2 061 905	-1.01%
27	40 048	39 734	-0.78%	*67*	305 351	307 277	0.63%	*107*	2 063 940	2 066 699	0.13%
28	37 774	37 027	-1.98%	*68*	298 483	297 488	-0.33%	*108*	2 072 675	2 055 530	-0.83%
29	39 267	39 228	-0.10%	*69*	312 283	313 755	0.47%	*109*	2 050 876	2 047 256	-0.18%
30	38 338	37 931	-1.06%	*70*	305 699	302 239	-1.13%	*110*	2 048 037	2 046 224	-0.09%
31	77 126	77 750	0.81%	*71*	418 583	416 751	-0.44%	*111*	11 646 936	11 580 787	-0.57%
32	84 418	84 124	-0.35%	*72*	397 544	396 019	-0.38%	*112*	11 909 296	11 871 461	-0.32%
33	79 449	80 211	0.96%	*73*	410 687	409 535	-0.28%	*113*	11 626 667	11 602 435	-0.21%
34	84 622	83 633	-1.17%	*74*	427 295	427 980	0.16%	*114*	11 801 140	11 765 074	-0.31%
35	86 164	85 570	-0.69%	*75*	404 143	403 001	-0.28%	*115*	11 846 478	11 793 864	-0.44%
36	81 774	81 967	0.24%	*76*	400 052	399 212	-0.21%	*116*	11 840 446	11 809 385	-0.26%
37	79 860	79 703	-0.20%	*77*	403 426	402 718	-0.18%	*117*	11 758 172	11 738 157	-0.17%
38	82 741	80 304	-2.95%	*78*	414 310	411 465	-0.69%	*118*	11 798 447	11 765 490	-0.28%
39	77 092	76 562	-0.69%	*79*	419 365	420 057	0.17%	*119*	11 763 442	11 681 991	-0.69%
40	85 219	84 691	-0.62%	*80*	431 535	418 364	-3.05%	*120*	11 789 891	11 747 580	-0.36%

Tabla 8.19. TCT de las secuencias obtenidas mediante ISA+CFI y MH-III (x=20, y=5) en las 120 instancias, y mejora porcentual de MH-III respecto a la heurística referencia.

9. CONCLUSIONES

Tratar de encontrar buenas soluciones para problemas de taller de flujo regular es algo habitualmente analizado, pues la adecuada secuenciación de los trabajos tratados en dichos escenarios es de vital importancia competitiva y económica para las empresas que gestionan esta clase de situaciones.

En este caso concreto, en el que el problema particular que se estudia es un *permutation flow shop problem* (PFSP) al que se le aplica la restricción *no-wait* y en el que se intenta minimizar el tiempo total de finalización (TCT), se realiza una revisión previa de otros artículos que tratan dicho problema, con el fin de identificar los procesos y métodos de resolución referentes hasta el momento.

Una vez identificada la heurística referencia, se ejecuta para 120 instancias estandarizadas habituales a la hora de comparar distintos procesos, guardándose los resultados obtenidos con ellas así como el tiempo empleado en hallarlos. Con esta información, ya se está en disposición de probar los algoritmos propuestos y poder compararlos.

Esta heurística referencia consiste en una rápida generación de una secuencia inicial de trabajos, paso del que parten todos los algoritmos propuestos, pues se demuestra que es un muy buen punto de partida; y posteriormente en un proceso iterativo de mejora de dicha secuencia inicial, que si bien consigue mejorarla generalmente, emplea demasiado tiempo, y es eso lo que se trata de evitar con los algoritmos propuestos.

En primer lugar, se plantean técnicas básicas de búsqueda local, cuyos resultados permiten decidir con cuáles de ellas se plantean los algoritmos posteriores.

Las heurísticas, que simplemente combinan algunas de las técnicas anteriores, ya se acercan bastante a los resultados de la heurística referencia, empleando un tiempo mucho menor.

Por último, se plantean unas metaheurísticas basadas en procedimientos GRASP, que además son parametrizables, por lo que permiten decidir en cada situación si centrarse en lograr el mejor resultado posible o simplemente en lograr uno bueno en poco tiempo. Basándose en las conclusiones extraídas al analizar los resultados de todos los algoritmos planteados previamente, se propone el algoritmo MH-III, con el que se mejora un 0,43% los resultados obtenidos por la heurística referencia empleando el mismo tiempo que ella. Mediante un análisis estadístico, se demuestra que dicha diferencia es significativa, probando así que realmente se encuentran mejores secuencias usando la metaheurística propuesta en este documento.

REFERENCIAS

Framinan, J. M., Leisten, R., Ruiz, R. (2014) "Manufacturing Scheduling Systems. An integrated view on Models, Methods and Tools". *Editorial Springer*.

Ruiz, R. and Stüztle, T. (2007) "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem". *European Journal of Operational Research*.

Pinedo, M. (2012) "Scheduling.Theory, Algorithms, and Systems". *Editorial Springer.*

Shyu, S. J., Lin, B. M., Yin, P.-Y. (2004) "Application of ant colony optimization for no-wait flowshop scheduling problems to minimize the total completion time". *Computers & Industrial Engineering.*

Garey, M. R., Johnson, D. S., Sethi, R. (1976) "The complexity of flowshop and jobshop scheduling". *Mathematics of operations research, 1(2), 117-119*

Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. (1979) "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium. Editorial Elseiver.*

Stafford Jr, E. F., Tseng, F. T., Gupta, J. N. D. (2005) "Comparative evaluation of MILP flowshop models". *Journal of the Operational Research Society, 56, 88-101.*

Lin, S.-W., Ying, K.-C. (2016) "Optimization of makespan for no-wait flowshop scheduling programs using different metaheuristics". *Omega, 64, 115 - 125.*

Rajendran, C., Chaudhuri, D. (1990) "Heuristics algorithms for continuous flow-shop problem". *Naval Research Logistics (NRL), 37(5), 695 - 705.*

Aldowaisan, T., Allahverdi, A. (2004) "New heuristics for m-machine no-wait flowshop to minimize total completion time". *Omega, 32(5), 345 - 352.*

Chen, C.-L., Neppalli, R. V., Aljaber, N. (1996) "Genetic algorithms applied to the continuous flow shop problem". *Computers & Industrial Engineering, 30(4), 919 - 929.*

Framinan, J. M., Nagano, M. S., Moccellin, J. V. (2010) "An efficient heuristic for total flowtime minimization in no-wait flowshops". *The International Journal of Advanced Manufacturing Technology, 46 (9 - 12), 1049 – 1057.*

Laha, D., Chakraborty, U. K. (2009) "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling". *The International Journal of Advanced Manufacturing Technology, 41 (1 - 2), 97 – 109.*

Gao, K., Pan, Q., Suganthan, P, Li, J. (2013) "Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization". *The International Journal of Advanced Manufacturing Technology, 66 (9 - 12), 1563 – 1572.*

Gao, K., Pan, Q., Li, J. (2011) "Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion". *The International Journal of Advanced Manufacturing Technology, 56 (5 - 8), 683 – 692.*

Bertolissi, E. (2000) "Heuristic algorithm for scheduling in the no-wait flow-shop". *Journal of Materials Processing Technology, 107(1), 459 – 465.*

Nawaz, M., Enscore, E. E., Ham, I. (1983) "A heuristic algorithm for the m.machine, n-job flow-shop sequencing problem". *Omega, 11(1), 91 – 95.*

Akhshabi, M., Tavakkoli-Moghaddam, R., Rahnamay-Roodposhti, F. (2014) "A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time". *The International Journal of Advanced Manufacturing Technology, 70(5 – 8), 1181 – 1188.*

Laha, D., Sapkal, S. U. (2014) "An improved heuristic to minimize total flow time for scheduling in the m-machine no-wait flow shop". *Computers & Industrial Engineering, 67, 36 – 43.*

Ye, H., Li, W., Abedini, A., Nault, B. (2017) "An effective and efficient heuristic for no-wait flow shop production to minimize total completion time". *Computers & Industrial Engineering, 108, 57 – 69.*

Liu, J., Reeves, C. R. (2001) "Constructive and composite heuristic solutions to the $P||\sum C_i$ scheduling problem". *European Journal of Operational Research, 132, 439 – 452.*

Fernandez-Viagas, V., Framinan, J. M. (2015) "A new set of high-performing heuristics to minimise flowtime in permutation flowshops". *Computers and Operations Research, 53, 68 – 80.*

Fernandez-Viagas, V., Dios, M., Framinan, J. M. (2016) "Efficient constructive and composite heuristics for the Permutation Flowshop to minimise total earliness and tardiness". *Computers and Operations Research, 75, 38 – 48.*

Framinan, J. M., Perez-Gonzalez, P. (2017) "The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic". *Computers and Operations Research, 88, 237 – 246.*

Zhu, X., Li, X. (2015) "Iterative search method for total flowtime minimization no-wait flowshop problem". *International Journal of Machine Learning and Cybernetics, 6(5), 747 – 761.*

Qi, X., Wang, H., Zhu, H., Zhang, J., Chen, F., Yang, J. (2016) "Fast local neighborhood search algorithm for the no-wait flow shop scheduling with total flow time minimization". *International Journal of Production Research, 54(16), 4957 – 4972.*

Bewoor, L. A., Prakash, V. C., Sapkal, S. U. (2017) "Comparative analysis of metaheuristics approaches for makespan minimization for no wait flow shop scheduling problem". *International Journal of Electrical and Computer Engineering.*

Lin, S.-W., Ying, K.-C. (2016) "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics". *Omega, 64, 115 – 125.*

Zhao, F., Liu, H., Zhang, Y., Ma, W., Zhang, C. (2018) "A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem". *Expert Systems with Applications.*

Riyanto, O. A. W., Santosa, B. (2015) "ACO-LS algorithm for solving no-wait flow shop scheduling problem". *Communications in Computer and Information Science.*

Ruiz, R., Allahverdi, A. (2009) "New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness". *International Journal of Production Research.*

Ye, S., Zhao, N., Li, K., Lei, C. (2017) "Efficient heuristics for solving non-permutation flow-shop scheduling problems with maximal and minimal time lags". *Computers & Industrial Engineering.*

Samarghandi, H., Behroozi, M. (2017) "On the exact solution of the no-wait flow shop problem with due date constraints". *Computers and Operations Research.*

Feo, T. A., Resende, M. G. C. (1989) "A probabilistic heuristic for a computationally difficult set covering problem". *Operations Research Letters, vol. 8, no. 2, pp. 67-71.*

Festa, P., Resende, M. G. C. (2001) "GRASP: An Annotated Bibliography". *Kluwer Academic Publishers, pp. 325-367, Boston, Mass, USA.*

ANEXOS

A. Análisis estadístico. Contraste de hipótesis.

Atendiendo a las Tablas 8.18 y 8.19, se aprecia que generalmente se obtienen mejores resultados aplicando la metaheurística MH-III que con la heurística referencia CFI, pues en 90 de las 120 instancias el TCT de la solución hallada con la primera es menor, siendo éste, en término medio, un 0,43 % inferior.

No obstante, con el fin de poder asegurar con una cierta confianza que realmente se logran mejores soluciones con la metaheurística MH-III, se realiza un análisis estadístico usando el software SPSS. Dicho análisis consiste en una prueba t de Student para muestras relacionadas, que en este caso consisten en los 120 valores del indicador RPD para cada una de las instancias de Taillard, tanto para la heurística CFI como para la MH-III. Se puede observar que los valores de la variable RPD_CFI en la Tabla A.1 son los que aparecían en las Tablas 8.3 y 8.5.

	RPD_CFI	RPD_MHIII
1	-1,24	-1,24
2	-4,13	-4,75
3	-2,59	-3,23
4	-5,31	-4,75
5	-4,98	-7,21
6	-11,13	-11,84
7	-4,66	-5,76
8	-1,17	-1,86
9	-5,76	-5,76
10	-3,04	-3,88

Tabla A.1. Extracto de las dos muestras a las que se realiza el análisis estadístico.

Los resultados del análisis permiten aceptar o rechazar las hipótesis planteadas a continuación:

Hipótesis nula (H_0): La metaheurística MH-III no logra encontrar soluciones con un TCT menor que las que se obtenían usando la heurística referencia CFI.

Hipótesis alternativa (H_1): Siguiendo la metaheurística MH-III se obtienen secuencias mejores que las halladas usando la heurística referencia CFI.

$$H_0 : \mu_{RPD\_MHIII} \geq \mu_{RPD\_CFI}$$

$$H_1 : \mu_{RPD\_MHIII} < \mu_{RPD\_CFI}$$

Cabe recordar que, según cómo se definió el RPD, un valor más negativo implica una secuencia final con menor TCT, y por tanto, mejor.

Por cómo se ha definido, se trata de un t-Test unilateral, ya que sólo se busca confirmar que con la metaheurística MH-III se obtienen resultados mejores que con CFI. De ahí que, como se aprecia en la Figura A.1, se define un porcentaje del intervalo de confianza del 97,5 % aunque se busque un nivel de confianza del 95 %.

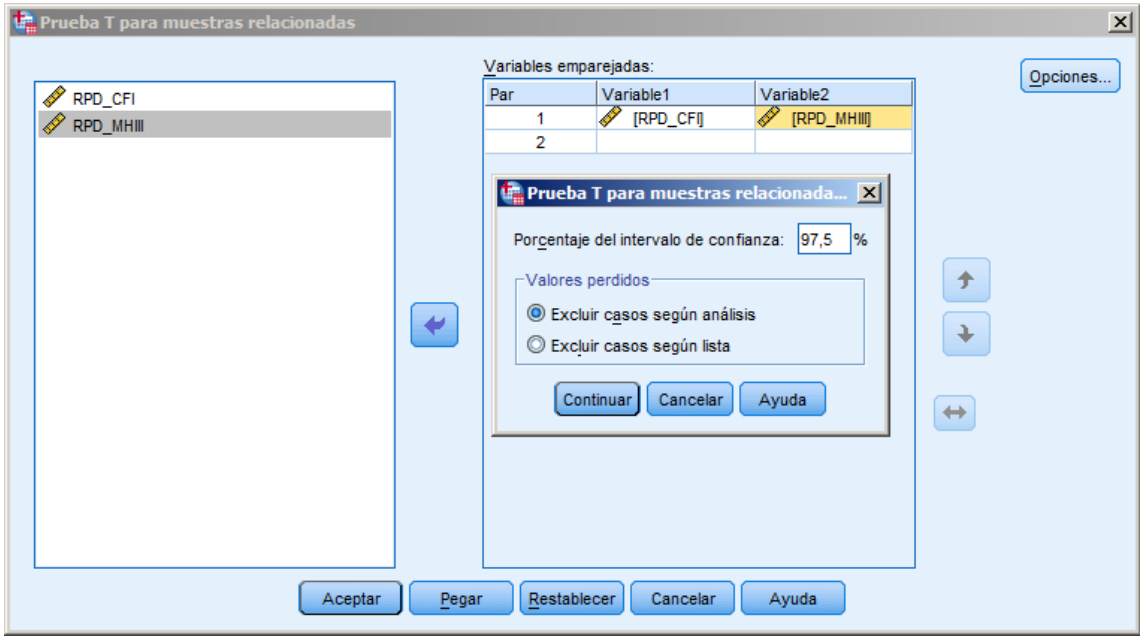

Figura A.1. Configuración del t-Test para muestras relacionadas efectuado.

Tras llevar a cabo el análisis, se obtienen los resultados presentados en la Figura A.2.

Estadísticas de muestras emparejadas

		Media	N	Desviación estándar	Media de error estándar
Par 1	RPD_CFI	-2,0717	120	2,12025	,19355
	RPD_MHIII	-2,4942	120	2,08409	,19025

Correlaciones de muestras emparejadas

		N	Correlación	Sig.
Par 1	RPD_CFI & RPD_MHIII	120	,940	,000

Prueba de muestras emparejadas

		Diferencias emparejadas					t	gl	Sig. (bilateral)
		Media	Desviación estándar	Media de error estándar	97,5% de intervalo de confianza de la diferencia				
					Inferior	Superior			
Par 1	RPD_CFI - RPD_MHIII	,42242	,73068	,06670	,27100	,57384	6,333	119	,000

Figura A.2. Resultados del t-Test para muestras relacionadas efectuado.

La primera tabla muestra, como ya se observaba en la Tabla 8.18, que la media del RPD para el MH-III es sensiblemente superior (en valor absoluto) a la de la heurística referencia.

En la segunda se confirma que ambas variables están muy relacionadas, cuya interpretación es que, como ambos algoritmos parten de la misma secuencia inicial (la que determina el algoritmo ISA), si dicha secuencia no es excesivamente buena, ambos la van a mejorar holgadamente, aunque no terminen llegando a la misma secuencia final.

La tercera tabla es la que confirma que sí hay una diferencia sustancial entre las mejoras logradas por ambos indicadores, y la que permite afirmar con un 95% de confianza que el algoritmo MH-III encuentra soluciones con un menor TCT que la heurística CFI. El valor de la t de 6,333 (claramente mayor que el presente en la Tabla A.2 para los casos con 120 grados de libertad y α en torno al 2,5 %), así como la no inclusión del cero en el intervalo de confianza de la diferencia entre los RPD, confirma el rechazo de la hipótesis nula, y por tanto la aceptación de la alternativa.

Degrees of freedom	Significance level					
	20% (0.20)	10% (0.10)	5% (0.05)	2% (0.02)	1% (0.01)	0.1% (0.001)
1	3.078	6.314	12.706	31.821	63.657	636.619
2	1.886	2.920	4.303	6.965	9.925	31.598
3	1.638	2.353	3.182	4.541	5.841	12.941
4	1.533	2.132	2.776	3.747	4.604	8.610
5	1.476	2.015	2.571	3.365	4.032	6.859
6	1.440	1.943	2.447	3.143	3.707	5.959
7	1.415	1.895	2.365	2.998	3.499	5.405
8	1.397	1.860	2.306	2.896	3.355	5.041
9	1.383	1.833	2.262	2.821	3.250	4.781
10	1.372	1.812	2.228	2.764	3.169	4.587
11	1.363	1.796	2.201	2.718	3.106	4.437
12	1.356	1.782	2.179	2.681	3.055	4.318
13	1.350	1.771	2.160	2.650	3.012	4.221
14	1.345	1.761	2.145	2.624	2.977	4.140
15	1.341	1.753	2.131	2.602	2.947	4.073
16	1.337	1.746	2.120	2.583	2.921	4.015
17	1.333	1.740	2.110	2.567	2.898	3.965
18	1.330	1.734	2.101	2.552	2.878	3.922
19	1.328	1.729	2.093	2.539	2.861	3.883
20	1.325	1.725	2.086	2.528	2.845	3.850
21	1.323	1.721	2.080	2.518	2.831	3.819
22	1.321	1.717	2.074	2.508	2.819	3.792
23	1.319	1.714	2.069	2.500	2.807	3.767
24	1.318	1.711	2.064	2.492	2.797	3.745
25	1.316	1.708	2.060	2.485	2.787	3.725
26	1.315	1.706	2.056	2.479	2.779	3.707
27	1.314	1.703	2.052	2.473	2.771	3.690
28	1.313	1.701	2.048	2.467	2.763	3.674
29	1.311	1.699	2.043	2.462	2.756	3.659
30	1.310	1.697	2.042	2.457	2.750	3.646
40	1.303	1.684	2.021	2.423	2.704	3.551
60	1.296	1.671	2.000	2.390	2.660	3.460
120	1.289	1.658	1.980	2.158	2.617	3.373
∞	1.282	1.645	1.960	2.326	2.576	3.291

Tabla A.2. Tabla de la distribución t de Student.

B. Resultados detallados de las técnicas de búsqueda local

En la Tabla 8.7 se indicaba el ARPD y el tiempo computacional empleado medio de cada una de las diez técnicas de búsqueda local para las 120 heurísticas. A continuación se adjuntan los resultados obtenidos por cada una de ellas para cada uno de los grupos de instancias de Taillard, dependiendo del tamaño de las mismas.

ISA + swapBEST		
Instancias	ARPD	Tiempo
1-10	-1.13	0.028
11-20	-0.24	0.043
21-30	-0.42	0.011
31-40	-0.59	0.029
41-50	-0.22	0.036
51-60	-0.21	0.050
61-70	-0.49	0.175
71-80	-0.15	0.115
81-90	-0.27	0.189
91-100	-0.14	0.751
101-110	-0.10	0.851
111-120	-0.04	10.322
TOTAL	-0.33	1.050

ISA + swapFIRST		
Instancias	ARPD	Tiempo
1-10	-1.32	0.018
11-20	-0.17	0.032
21-30	-0.36	0.043
31-40	-0.66	0.046
41-50	-0.22	0.053
51-60	-0.20	0.057
61-70	-0.62	0.132
71-80	-0.17	0.106
81-90	-0.26	0.134
91-100	-0.15	0.617
101-110	-0.11	0.667
111-120	-0.04	6.637
TOTAL	-0.36	0.712

ISA + swapFIRSTBEST		
Instancias	ARPD	Tiempo
1-10	-1.39	0.015
11-20	-0.17	0.040
21-30	-0.36	0.104
31-40	-0.56	0.086
41-50	-0.22	0.076
51-60	-0.17	0.101
61-70	-0.53	0.135
71-80	-0.16	0.167
81-90	-0.28	0.205
91-100	-0.14	0.437
101-110	-0.10	0.517
111-120	-0.04	5.100
TOTAL	-0.34	0.582

Tabla B.1. ARPD y tiempo computacional medio, agrupados por instancias, de las técnicas de búsqueda local de intercambio de trabajos.

ISA + insBEST		
Instancias	ARPD	Tiempo
1-10	-3.10	0.028
11-20	-1.85	0.045
21-30	-1.19	0.056
31-40	-2.09	0.100
41-50	-1.01	0.096
51-60	-0.82	0.098
61-70	-3.00	0.407
71-80	-1.47	0.417
81-90	-0.71	0.336
91-100	-1.12	3.197
101-110	-0.65	3.545
111-120	-0.35	78.717
TOTAL	-1.45	7.253

ISA + insFIRST		
Instancias	ARPD	Tiempo
1-10	-2.96	0.014
11-20	-1.68	0.017
21-30	-1.29	0.017
31-40	-2.08	0.070
41-50	-1.03	0.057
51-60	-0.85	0.075
61-70	-3.16	0.621
71-80	-1.51	0.442
81-90	-0.74	0.364
91-100	-1.15	4.890
101-110	-0.76	4.650
111-120	-0.37	123.427
TOTAL	-1.47	11.220

ISA + insFIRSTBEST		
Instancias	ARPD	Tiempo
1-10	-3.12	0.017
11-20	-1.65	0.032
21-30	-1.32	0.031
31-40	-2.07	0.065
41-50	-1.15	0.046
51-60	-0.68	0.062
61-70	-2.94	0.187
71-80	-1.40	0.223
81-90	-0.71	0.233
91-100	-1.08	0.854
101-110	-0.65	1.093
111-120	-0.36	12.433
TOTAL	-1.43	1.273

Tabla B.2. ARPD y tiempo computacional medio, agrupados por instancias, de las técnicas de búsqueda local de inserción completa de trabajos.

ISA + insforBEST		
Instancias	ARPD	Tiempo
1-10	-1.31	0.015
11-20	-0.49	0.043
21-30	-0.25	0.035
31-40	-0.89	0.045
41-50	-0.34	0.045
51-60	-0.20	0.064
61-70	-1.59	0.142
71-80	-0.63	0.140
81-90	-0.17	0.145
91-100	-0.44	0.806
101-110	-0.25	0.932
111-120	-0.14	15.355
TOTAL	-0.56	1.480

ISA + insforFIRST		
Instancias	ARPD	Tiempo
1-10	-1.13	0.026
11-20	-0.33	0.037
21-30	-0.24	0.034
31-40	-0.84	0.062
41-50	-0.25	0.068
51-60	-0.21	0.076
61-70	-1.63	0.271
71-80	-0.62	0.190
81-90	-0.15	0.190
91-100	-0.45	1.333
101-110	-0.21	1.106
111-120	-0.13	24.305
TOTAL	-0.52	2.308

ISA + insforFIRSTBEST		
Instancias	ARPD	Tiempo
1-10	-1.45	0.015
11-20	-0.33	0.015
21-30	-0.24	0.015
31-40	-0.80	0.034
41-50	-0.30	0.043
51-60	-0.20	0.057
61-70	-1.59	0.120
71-80	-0.63	0.117
81-90	-0.17	0.145
91-100	-0.42	0.408
101-110	-0.23	0.565
111-120	-0.14	5.914
TOTAL	-0.54	0.621

Tabla B.3. ARPD y tiempo computacional medio, agrupados por instancias, de las técnicas de búsqueda local de inserción hacia delante de trabajos.

ISA + NEH$_{LOC}$		
Instancias	ARPD	Tiempo
1-10	-2.90	0.026
11-20	-1.32	0.045
21-30	-1.06	0.048
31-40	-1.35	0.053
41-50	-0.89	0.062
51-60	-0.61	0.087
61-70	-2.33	0.097
71-80	-0.84	0.112
81-90	-0.23	0.145
91-100	-0.49	0.234
101-110	-0.15	0.393
111-120	-0.01	2.293
TOTAL	-1.01	0.300

Tabla B.4. ARPD y tiempo computacional medio, agrupados por instancias, de la técnica de búsqueda local NEH$_{LOC}$.