Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

# Efecto de los tiempos de espera en talleres de flujo regular

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, 2017

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

# Efecto de los tiempos de espera en talleres de flujo regular


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, 2017

Trabajo Fin de Grado: Efecto de los tiempos de espera en talleres de flujo regular

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, 2017

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 ser uno de los incluidos en el punto anterior, me ha orientado perfectamente desde el primer día, resolviendo todas mis dudas rápida y claramente cada vez que fue necesario.

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.

La programación de operaciones es crucial para el resultado de una empresa, ya que determina 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. (Framiñán, Leister, Ruiz, 2014)

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

En una planta de fabricación de productos 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, se pueden tener los siguientes entornos posibles:

- 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 entornos 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 tienen entornos diferentes en diversas secciones de las mismas.

El entorno considerado para este proyecto es un *flow shop*, o taller de flujo regular, en el que todos los trabajos deben seguir la misma ruta de máquinas, dispuestas en serie (1, 2, … , $m$). 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, la siguiente tarea de un trabajo dado puede comenzar tan pronto como la anterior haya finalizado, salvo que la siguiente máquina 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 hacer que la secuencia de procesado de trabajos en la primera máquina se mantenga en el resto de máquinas de la planta. El problema resultante se denomina *permutation flowshop problem* (PFSP), y tiene $n!$ posibles soluciones (es decir, secuencias a la entrada de la primera máquina).

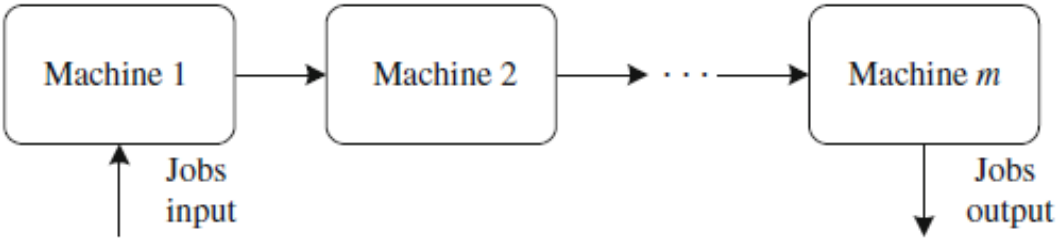


**Figura 1.1** – Flujo de trabajos en un entorno *flow shop* (Framiñán, Leister, Ruiz García, 2014)

A partir del momento en el que el trabajo *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 *j+1*.

Uno de los criterios a usar será el *makespan* (*Cmax*), que se define como el tiempo de finalización del último trabajo en la última máquina. La minimización de esta variable es uno de los objetivos más significativos de este tipo de situaciones, pues delimita el tiempo en el que hay actividad en la fábrica. (Ye, Li, Abedini, 2017)

Para hallar el *makespan*, 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 siguiente esquema, se llegaría al tiempo de finalización del procesado del trabajo *n* en la máquina *m*.

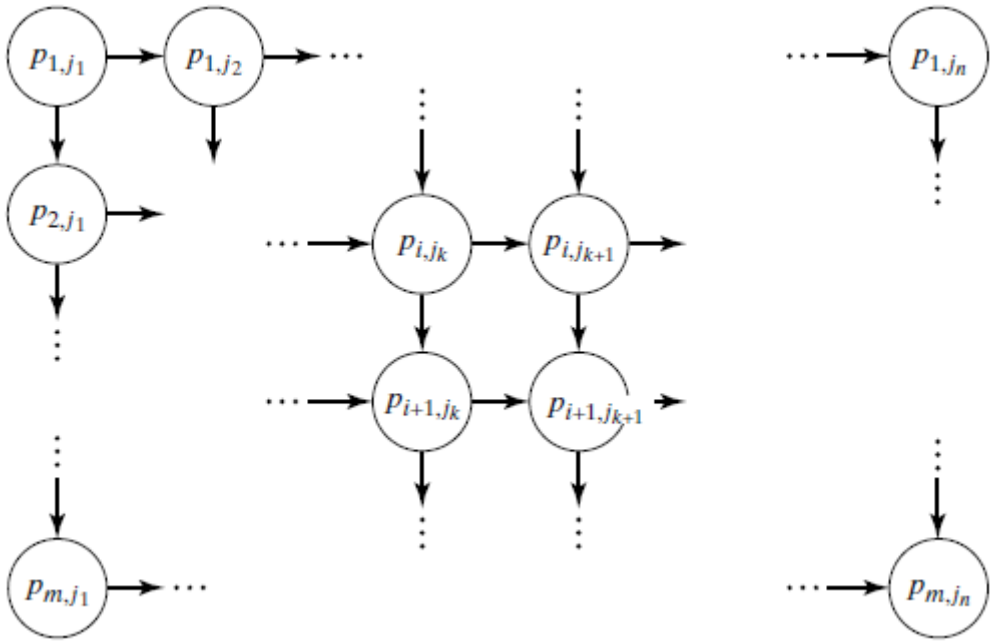


**Figura 1.2** – Cálculo del *makespan*, de manera genérica, en un taller de flujo regular (Pinedo, 2012)

# 1. Introducción

En este proyecto, atendiendo a distintos criterios relacionados con el tiempo de finalización de los trabajos, se estudia cómo afecta a cada uno de ellos aplicar diversas restricciones en el proceso de fabricación. En concreto, se consideran restricciones referentes al tiempo de espera máximo de un trabajo al pasar de una máquina a la siguiente.

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 donde 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 los propuestos por Framiñán, 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, y 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.

El uso de talleres de tipo *flow shop* donde no se permita esperar a los trabajos entre dos máquinas consecutivas (es decir, con restricciones *no-wait*) es bastante común en procesos de fabricación. Sin embargo, la minimización del *makespan* en estos casos es un problema no polinomial (Ye, Li, Abedini, Nault, 2017), 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 para intentar llegar a una solución no muy distinta a la óptima de manera relativamente sencilla.

En procesos de fabricación *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 cuatro máquinas y cinco trabajos 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 que se aplica una restricción *no-wait*, y por ello el trabajo 4 ve cómo sus tres primeras tareas empiezan más tarde de lo que podrían hacerlo, para así asegurarse de que podrá tratarse en la cuarta máquina inmediatamente después de finalizar su procesado en la tercera.

| $p_{ij}$ | | trabajo (j) | | | | |
|---|---|---|---|---|---|---|
| | | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** |
| máquina (i) | ***1*** | 4 | 5 | 7 | 4 | 5 |
| | ***2*** | 3 | 5 | 4 | 3 | 3 |
| | ***3*** | 5 | 4 | 3 | 4 | 3 |
| | ***4*** | 6 | 3 | 7 | 3 | 3 |

**Tabla 1.1** – Tiempos de proceso para un problema con cinco trabajos y cuatro máquinas (Elaboración propia)

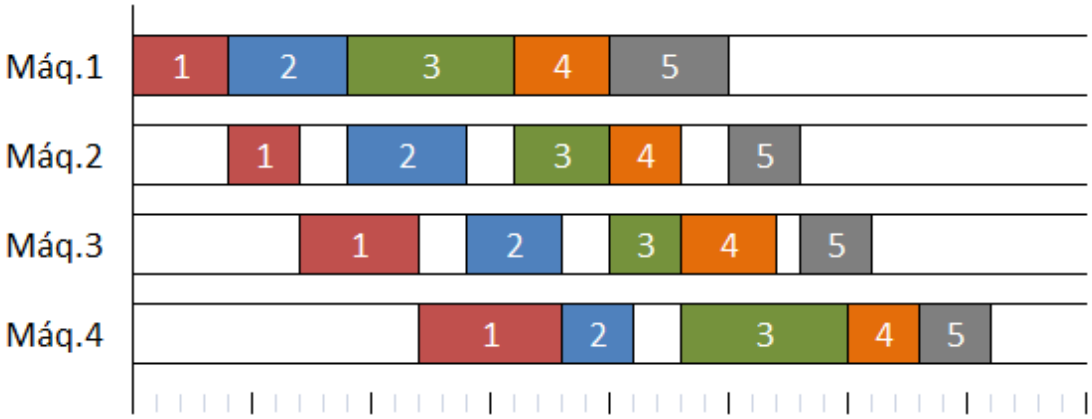


**Figura 1.3** – Diagrama de Gantt resultado de la secuencia π={1,2,3,4,5} para un PFSP sin más restricciones, con los tiempos de proceso de la tabla 1.1 (Elaboración propia)

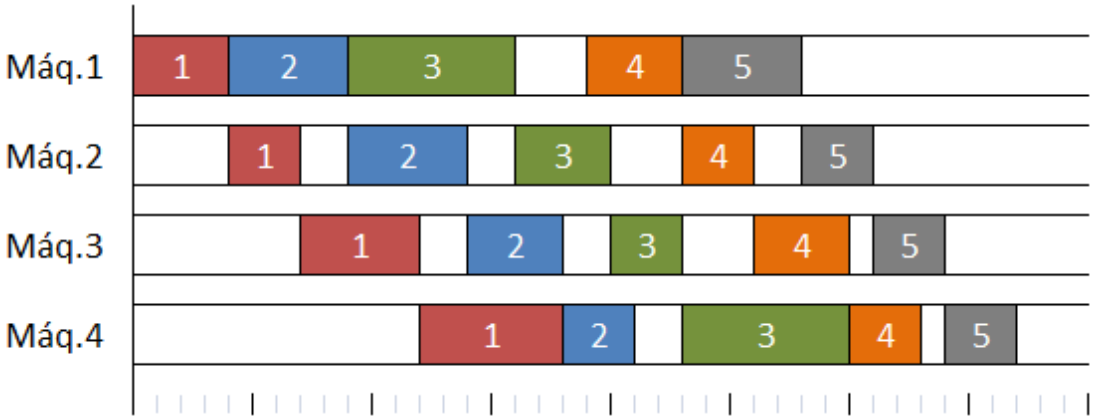


**Figura 1.4** – Diagrama de Gantt resultado de la secuencia π={1,2,3,4,5} para un PFSP con restricción *no-wait*, con los tiempos de proceso de la tabla 1.1 (Elaboración propia)

No sólo se tratarán los casos extremos donde o no se admitan esperas o se consientan esperas indefinidas, sino que se verán casos donde, permitiéndose una cierta espera, ésta esté limitada. Por ejemplo, para el ejemplo anterior del tratamiento de alimentos se podría tener el caso en el que esperar cinco minutos no llegase a romper la cadena del frío, y por tanto una espera inferior a ese umbral entre etapa y etapa se considerase asumible.

También 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 (Framiñán, Leister, Ruiz, 2014). Aunque estas situaciones no se considerarán en este proyecto.

# 1. Introducción

En los siguientes apartados, se detallará 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 describirá más detalladamente el problema a resolver, definiendo las distintas restricciones que se aplicarán al mismo, así como los modelos matemáticos que los caracterizan.

- En el capítulo 3 se explicarán los pasos que se siguen para obtener los resultados de cada problema, y los motivos por los que obtener dichos resultados resulta de interés para el objetivo del trabajo.

- En el capítulo 4 se indicarán los experimentos que se han llevado a cabo para obtener los resultados que se nombraron en el capítulo previo, justificando el porqué se han considerado los más apropiados para este estudio.

- En el capítulo 5 se describirán los programas informáticos construidos y usados con el fin de obtener los resultados descritos en el capítulo 3.

- El capítulo 6 se compondrá de la evaluación computacional, es decir, los resultados obtenidos al ejecutar los programas, y una primera interpretación de los mismos.

- En el capítulo 7 se expondrán las conclusiones que se pueden sacar a partir de la evaluación computacional realizada y expuesta previamente.

- Por último, se adjuntará la bibliografía utilizada para la elaboración del documento, así como anexos con información que no se consideró relevante como para incluirse en el mismo.

# 2 DESCRIPCIÓN DEL PROBLEMA

El proyecto consiste en el estudio del efecto de ciertas restricciones a un proceso de producción dado, donde dichas restricciones serán las que definan los distintos problemas.

Los problemas consisten en un entorno de trabajo de tipo taller, *flow shop*, con *m* máquinas, donde los *n* trabajos que la fábrica produce deben pasar por todas las máquinas siguiendo el orden que éstas, dispuestas en serie, marcan.

Se consideran dos criterios a la hora de evaluar las distintas secuencias de producción, ambos relacionados con el tiempo de finalización de la producción de los trabajos. Uno de ellos es minimizar el *makespan*, que es el tiempo en el que se finaliza el último de los trabajos, mientras que el otro trata de minimizar el tiempo total de finalización, que se define como la suma de los tiempos de finalización de cada uno de los trabajos procesados.

Se aplica la restricción de permutación en todos los casos a tratar, que hace que el orden de procesado de los trabajos sea idéntico en cada una de las máquinas presentes en el proceso, convirtiéndose así el problema en un PFSP (*permutation flow shop problem*).

Se considera una segunda clase de restricciones, que es donde reside el núcleo del proyecto. Estas restricciones pueden limitar el tiempo de espera máximo de los trabajos entre que salen de una máquina e inician su actividad en la posterior. Se parte del caso en el que esta restricción no se aplica, y a partir de ahí se comienza a considerar un tiempo máximo de espera, que se ve reducido de forma escalonada, hasta llegar a tener una restricción *no-wait*, que obliga a los trabajos a pasar de una máquina a la siguiente de forma inmediata, no permitiéndose tiempo de espera alguno entre ellas.

No se considera ningún otro tipo de restricciones (llegada de trabajos, fechas de entrega, tiempos de *setup*, *buffer*…), aunque obviamente sí 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.

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, $\alpha|\beta|\gamma$, siendo cada uno de ellos:

- $\alpha$: el entorno del proceso de fabricación, es decir, la disposición de las máquinas y cuáles se deberán usar para producir los trabajos.

- $\beta$: las distintas restricciones que se aplican al problema.

- $\gamma$: el objetivo en el que hay que basarse para decidir qué solución es más beneficiosa.

Por ejemplo, una notación $R4|r_j, d_j|\sum U_j$ hace referencia a un problema donde se tienen cuatro máquinas paralelas con velocidades de procesado no relacionadas entre ellas y donde los trabajos sólo tienen que pasar por una de dichas máquinas, aunque éstos no están disponibles en su totalidad desde el inicio, pues cada uno tiene una fecha de llegada dada, y en el que se intenta que el número de total de trabajos que finalicen después de su fecha de entrega sea el menor posible.

Como se ha explicado en el capítulo 1, en este proyecto se analizarán problemas donde los trabajos puedan esperar indefinidamente, donde el paso de los trabajos de una máquina a otra haya de ser inmediato, y donde los tiempos de espera estén acotados superiormente. Es decir, que se considerarán tres posibles conjuntos de restricciones que formen el campo β. Asimismo, se tratarán dos objetivos a minimizar dependiendo del problema: el *makespan* y el tiempo total de finalización de los trabajos, lo que hace que se puedan tener dos campos γ distintos. El campo α, por otra parte, siempre será el mismo, porque el único entorno con el que se va a trabajar es un taller de flujo regular, aunque pueda variar el número de máquinas que conformen el taller. Todo ello que se tengan seis tipos de problemas diferentes en este proyecto, según las restricciones que se les apliquen y el objetivo que traten de minimizar.

A continuación se listan las variables que se usan para el modelado de los problemas descritos, así como sus datos. La variable $e_{ij}$ y el dato $emax$ sólo se necesitan a la hora de modelar los problemas donde las esperas, permitiéndose, están limitadas.

Notación de las variables y datos del problema

$p_{ij}$ : Tiempo que tarda en procesarse el trabajo *j* en la máquina *i*

$C_{ij}$ : Tiempo de finalización del trabajo *j* en la máquina *i*

$C\_max$ : *Makespan*, tiempo de finalización del último trabajo en la última máquina

$X_{jk}$ : 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

$e_{ij}$ : Tiempo que espera el trabajo *j* antes de entrar en la máquina *i*

$emax$ : Tiempo máximo que puede estar esperando un trabajo entre que termina su actividad en una máquina y entra en la siguiente

## *2.1 Problemas donde se minimiza el makespan*

En primer lugar, se describirán los problemas en los que el objetivo es lograr el menor valor posible de *Cmax*, que serán tres, según el conjunto de restricciones que se les apliquen.

### 2.1.1 Problemas con esperas ilimitadas

Este primer problema se denota *Fm|prmu|Cmax* usando la notación α|β|γ, que significa:

- Entorno: taller de flujo regular (*flow shop*) formado por *m* máquinas.
- Restricciones: permutación, es decir, todos los trabajos se procesan en el mismo orden en todas las máquinas.
- Objetivo: minimizar el *makespan.*

Pérez, Fernández-Viagas, Framiñán (2016) expresan este problema mediante el siguiente modelo de programación lineal entera mixta (MILP):

MIN $C\_max$

s.a:

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

$$C_{ij} - C_{i-1,j} \geq p_{ij} \quad \forall i \in M - \{1\}, \ \forall j \in N$$

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

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

$$C\_max \geq C_{mj} \quad \forall j \in N$$

Este modelo es la base de la que parten los modelos que describen los otros dos problemas para este objetivo, pues éstos se forman a partir de añadir una restricción al problema con esperas ilimitadas.

### 2.1.2 Problemas con esperas no admisibles

La restricción que se tiene en este problema es una bastante común en ciertos tipos de industria, la restricción *no-wait* o *nwt*), que obliga a los trabajos a pasar de una máquina a otra sin detenerse. Estos problemas se denotan *Fm*|*prmu*,*nwt*|*Cmax*.

MIN $C\_max$

s.a:

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

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

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

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

$$C\_max \geq C_{mj} \quad \forall j \in N$$

Se aprecia que en la segunda restricción ya no se trata de una inecuación, sino de una ecuación, pues debido a la ausencia de tiempos de espera, la diferencia entre el tiempo de finalización de procesado de un trabajo en una máquina *i* y en su predecesora será exactamente igual al tiempo de procesado de dicho trabajo en la máquina *i*.

### 2.1.3 Problemas con esperas permitidas pero limitadas

Este tercer tipo de problema usa la restricción menos común de las usadas hasta ahora, por lo que se le debe otorgar una notación para poder incluirla en el campo β de la denominación del problema. La restricción que limita el tiempo máximo que pueden esperar los trabajos entre máquina y máquina se denota como *'max_t'*. Así, un campo β que incluya la restricción max_20 hace que los trabajos puedan esperar hasta un máximo de 20 unidades temporales entre que terminan su actividad en una máquina y entran en la siguiente, y se denotaría *Fm|prmu,max20|Cmax.*

MIN $C\_max$

s.a:

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

$$C_{i-1,j} + e_{ij} + p_{ij} = C_{ij} \quad \forall i \in M - \{1\}, \ \forall j \in N$$

$$e_{ij} \leq e\_max \quad \forall i \in M, \ \forall j \in N$$

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

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

$$C\_max \geq C_{mj} \quad \forall j \in N$$

En este modelo, el parámetro *e_max* presente en el tercer conjunto de restricciones es igual a la 't' que acompaña a 'max' en la restricción correspondiente del campo β de la notación del mismo.

## *2.2 Problemas donde se minimiza el tiempo total de finalización*

Se han visto los modelos de los problemas a tratar en este proyecto atendiendo a los distintos tipos de restricciones que se aplican, pero sólo para uno de los dos objetivos considerados, el *makespan*. Sin embargo, considerar el otro objetivo (minimización de la suma de los tiempos de finalización de procesado de todos los trabajos) sólo conlleva un cambio en la función objetivo de los modelos, pues las restricciones no se ven alteradas por ello, por lo que el modelado de estos problemas es idéntico a los anteriores salvo por el cambio en el objetivo.

### 2.2.1 Problemas con esperas ilimitadas

A continuación se presenta el modelo de programación lineal entera para el problema $Fm|prmu|\sum C_j$:

MIN $\sum_{j=1}^{n} C_{mj}$

s.a:

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

$$C_{ij} - C_{i-1,j} \geq p_{ij} \quad \forall i \in M - \{1\}, \ \forall j \in N$$

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

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

$$C\_max \geq C_{mj} \quad \forall j \in N$$

### 2.1.2 Problemas con esperas no admisibles

A continuación se presenta el modelo de programación lineal entera para el problema *Fm|prmu,nwt|*$\sum C_j$:

MIN $\sum_{j=1}^{n} C_{mj}$

s.a:

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

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

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

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

$$C\_max \geq C_{mj} \quad \forall j \in N$$

### 2.1.3 Problemas con esperas permitidas pero limitadas

A continuación se presenta el modelo de programación lineal entera para el problema *Fm|prmu,max_t|*$\sum C_j$:

MIN $\sum_{j=1}^{n} C_{mj}$

s.a:

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

$$C_{i-1,j} + e_{ij} + p_{ij} = C_{ij} \quad \forall i \in M - \{1\}, \ \forall j \in N$$

$$e_{ij} \leq e\_max \quad \forall i \in M, \ \forall j \in N$$

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

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

$$C\_max \geq C_{mj} \quad \forall j \in N$$

# 3 METODOLOGÍA

## 3.1 Problemas y análisis

Este proyecto estudia el efecto de limitaciones en el tiempo de espera de los trabajos entre máquinas en un taller de flujo regular, y para ello se resuelven los modelos propuestos en el capítulo 2, y se comparan los resultados que se obtienen de cada uno de ellos.

Como se ha descrito en el capítulo 2, los problemas con los que se trabaja son:

- *Fm|prmu|Cmax* ($\sum C_j$) : taller de flujo regular con restricción de permutación, PFSP.
- *Fm|prmu,nwt|Cmax* ($\sum C_j$) : PFSP sin esperas permitidas entre máquinas.
- *Fm|prmu,max_t|Cmax* ($\sum C_j$) : PFSP con esperas limitadas entre máquinas según el valor que tome el parámetro *t*, que serán varios a lo largo de los análisis.

El primer análisis que se realizará será la comparación de los resultados óptimos de los problemas con restricciones en los tiempos de espera entre máquinas, para cada uno de los entornos contemplados, respecto al problema con esperas indefinidas permitidas. Debido a que un problema con un tiempo de espera permitido reducido es siempre más restrictivo que un PFSP, el valor del objetivo a minimizar aumenta, pero se analizará en qué cantidad lo hace para cada tipo de restricción y entorno.

Posteriormente, se analizará la dificultad de los problemas de formas diferentes pero relacionadas. Calculando la diferencia media entre el resultado que se obtienen con una secuencia óptima y el que da la óptima, y hallando también la distribución de todas las secuencias según los resultados que se obtengan de las mismas.

Por último, se estudiará el comportamiento de la secuencia óptima del problema con esperas permitidas indefinidamente (PFSP) en los otros cinco problemas, con esperas limitadas o incluso prohibidas. Asimismo, también se aplicará la óptima de cada uno de esos cinco problemas en el PFSP con el fin de comparar sus resultados con los que se obtienen con la óptima del propio PFSP.

## *3.2 Indicadores y Enumeración Completa*

### 3.2.1 Indicadores

Además del propio valor de la FO que se use en cada caso, se introduce un indicador para medir la dificultad de cada problema, que expresa el porcentaje de desviación respecto al óptimo.

El RPD (*Relative Percentage Deviation*) indica el incremento porcentual relativo del valor de la FO para una secuencia cualquiera respecto al valor mínimo de dicha FO, es decir, el valor que toma la FO cuando se sigue la secuencia óptima para el problema en cuestión.

$$RPD = \frac{FO(\pi_i) - FO(\pi_{opt})}{FO(\pi_{opt})} \, x \, 100$$

Dependiendo de si se está calculando el RPD para el objetivo del *makespan* o del tiempo total de finalización, FO será *Cmax* o $\sum C_j$.

Se denomina RPD cuando se calcula para una secuencia concreta. Pero lo que resulta de interés es el aumento medio del valor de la FO para todas las posibles secuencias del problema, pues eso caracteriza al problema mucho mejor que el RPD de una secuencia cualquiera. En ese caso el indicador recibe el nombre de ARPD (*Average Relative Percentage Deviation*).

Para hallar el ARPD, se debe promediar el RPD de cada una secuencias posibles, que serán *n!*.

$$ARPD = \frac{\sum_{sec=1}^{n!} RPD_{sec}}{n!}$$

Así, si se tuviese un problema con un ARPD del 20%, eso significaría que el valor de la FO al seguir una secuencia cualquiera sería, de media, un 20% mayor que el mínimo valor de la misma, obtenido con la secuencia óptima. Es decir, que si se siguiese una secuencia de trabajos aleatoria, el valor de la FO que se obtendría sería un 20% superior al mejor posible para ese problema.

Calcular el RPD para una secuencia concreta, aunque no sirva para conocer la dificultad de un problema, sí es de utilidad cuando se quiere comparar la secuencia óptima de un problema en otro. Como se trabaja con seis conjuntos de restricciones diferentes pero relacionados entre sí (seis valores distintos para el tiempo de espera máximo de los trabajos), resulta de interés analizar el resultado que se obtiene al usar la secuencia óptima de un problema en otro.

### 3.2.2 Enumeración completa

Para proceder con los análisis planeados y obtener los resultados deseados, es necesario estudiar todas las secuencias posibles de cada problema y en cada instancia, para lo cual se usa la enumeración completa, que se aplica dos veces en cada una de las instancias.

## 3. Metodología

Para comprender mejor el proceso, se explica cómo se procedería para un problema simplificado, con apenas dos máquinas y tres trabajos, sin esperas permitidas entre máquinas, y con los siguientes tiempos de proceso.

| $p_{ij}$ | | trabajo (j) | | |
|---|---|---|---|---|
| | | ***1*** | ***2*** | ***3*** |
| máquina (i) | ***1*** | 6 | 8 | 3 |
| | ***2*** | 3 | 7 | 6 |

**Tabla 3.1** – Tiempos de proceso para un problema con dos máquinas y tres trabajos (Elaboración propia)

La primera aplicación de la enumeración completa solamente sirve para saber cuáles son los valores mínimo y máximo que toma la FO en la instancia en cuestión, así como la secuencia óptima, que es con la que se obtiene el mínimo de la FO. Esa información se recoge en la tabla 3.2, y es que se deben conocer esos valores de antemano para poder obtener los resultados que definen en profundidad las características de los problemas.

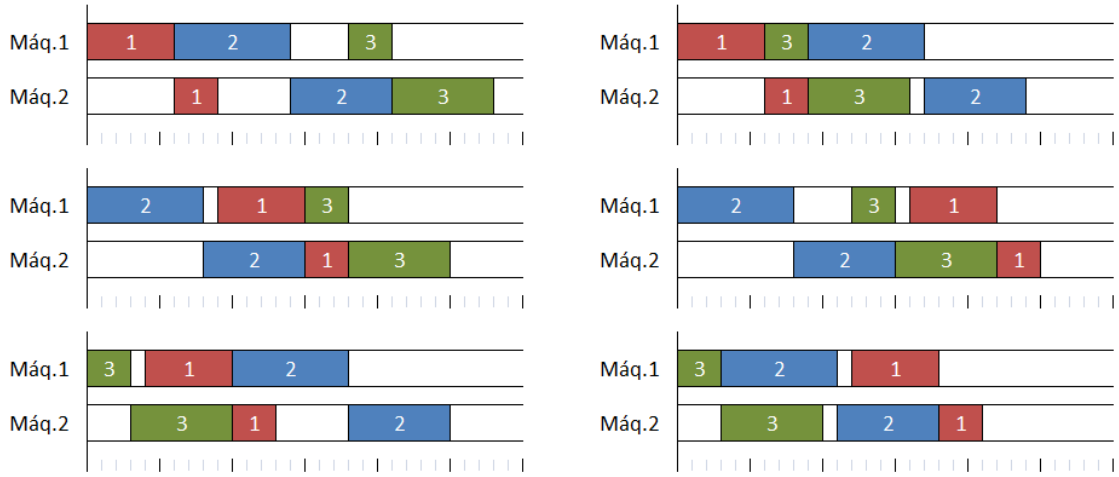


**Figura 3.1** – Diagramas de Gantt para todas las posibles secuencias de un problema de taller de flujo regular sin esperas permitidas con 3 trabajos, con los tiempos de proceso de la tabla 3.1 (Elaboración propia)

| Secuencia | FO: Cmax |
|---|---|
| 1-2-3 | 28 |
| 1-3-2 | 24 |
| 2-1-3 | 25 |
| 2-3-1 | 25 |
| 3-1-2 | 25 |
| 3-2-1 | 21 |
| MIN 21 *(3-2-1)* | |
| MAX 28 | |

**Tabla 3.2** – Valores del *makespan* para cada una de las secuencias de un problema con tres trabajos (Elaboración propia)

En la segunda aplicación de la enumeración completa, al conocer ya el valor de la FO para la secuencia óptima, se puede hallar en cuánto aumenta el valor de la FO para cada una de las otras secuencias posibles, pudiendo hallar de esta forma el ARPD para la instancia en cuestión.

| Secuencia | FO: Cmax | RPD |
|---|---|---|
| 1-2-3 | 28 | 33,33 |
| 1-3-2 | 24 | 14,29 |
| 2-1-3 | 25 | 19,05 |
| 2-3-1 | 25 | 19,05 |
| 3-1-2 | 25 | 19,05 |
| 3-2-1 | 21 | 0,00 |
| | ARPD | *17,46* |

**Tabla 3.3** – Cálculo del RPD para cada una de las secuencias de un problema, y de su ARPD (Elaboración propia)

Para calcular el ARPD basta con conocer el valor mínimo de la FO en la primera aplicación, pero se ha visto que no es lo único que se halla en ella. El valor máximo, por ejemplo, sirve para acotar los valores extremos que la FO puede tomar según la secuencia seguida.

Siguiendo con este problema modelo, el intervalo de posibles valores comprendería aquéllos entre 21 y 28. En los problemas reales estudiados, que se explicarán en el capítulo siguiente, se dividirá dicho intervalo en cien tramos debido a la mayor complejidad de los problemas objeto de estudio en este proyecto. Sin embargo, para analizar el problema modelo en el que se centra este capítulo, se dividirá en sólo diez tramos, pues se tienen muy pocas secuencias.

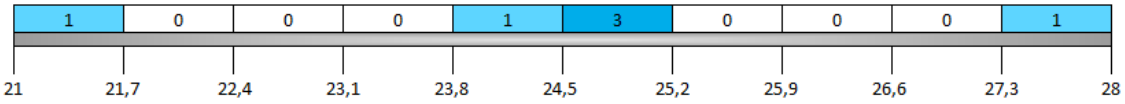


**Figura 3.2** – Densidad de probabilidad de las secuencias del problema *Fm|prmu,nwt|Cmax* con los tiempos de proceso de la tabla 3.1, según el valor que toma la FO (Elaboración propia)

En este caso, por ejemplo, 3 de las 6 secuencias se encuentran en el tramo seis, por lo que existe un 50% de probabilidad de que, al elegir aleatoriamente una secuencia para este problema, ésta dé un valor de la FO entre 24,5 y 25,2. Evidentemente, hay al menos una secuencia en el primer tramo y otra en el último, algo que tendrá cierta importancia a la hora de analizar los resultados de los problemas objeto de estudio en este proyecto, en el capítulo 5.

El tercer dato que se obtiene a partir de la primera aplicación de la enumeración completa es la secuencia óptima, que es necesario conocer a la hora de estudiar cómo se comportan las secuencias óptimas de unos problemas en otros con diferentes restricciones. Por ejemplo, si se quisiera estudiar la aplicación de la secuencia óptima del problema sin esperas, mostrada en la tabla 3.2, en el problema sin limitación en los tiempos de espera, habría que hallar el mejor resultado del PFSP, y posteriormente comparar dicho valor con el obtenido al usar la secuencia de *no-wait*.

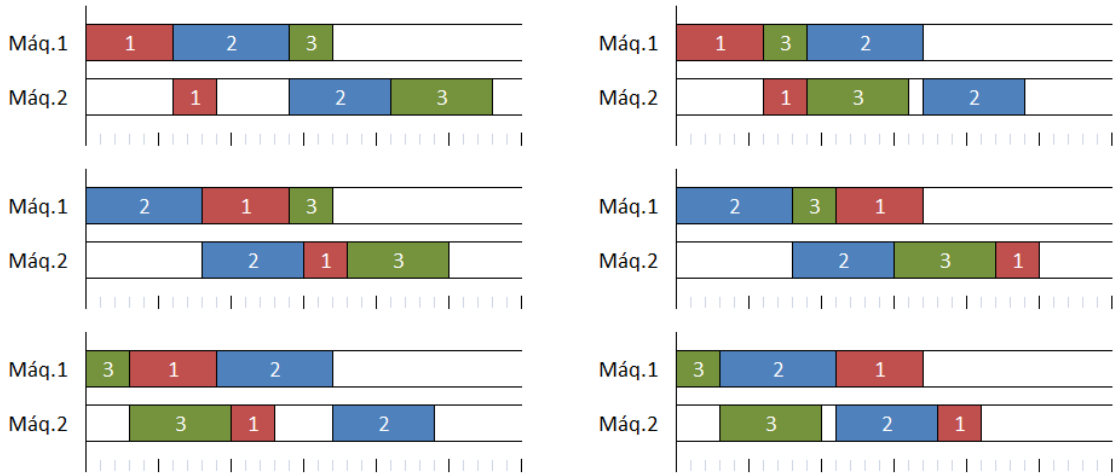


**Figura 3.3** – Diagramas de Gantt para todas las posibles secuencias de un problema de taller de flujo regular con esperas ilimitadas y 3 trabajos, con los tiempos de proceso de la tabla 3.1 (Elaboración propia)

La secuencia óptima del problema *no-wait* era 3-2-1, que coincide con la del PFSP, por lo que se podría afirmar que para este ejemplo concreto se puede usar la secuencia óptima del primer problema en el segundo y seguir obteniendo el óptimo del PFSP. Evidentemente, en entornos tan simples, las secuencias de los distintos problemas suelen coincidir, pues las restricciones no cambian enormemente el comportamiento de los mismos. Sin embargo, cuando el número de trabajos y máquinas crece, las diferencias en las limitaciones de los tiempos de espera, aún siendo pequeñas, hacen que la secuencia óptima de un problema pueda dar soluciones muy alejadas del óptimo de otros.

# 4 DISEÑO DE EXPERIMENTOS

Para aplicar la metodología expuesta en el capítulo anterior de forma que se obtengan resultados de los que se puedan extraer conclusiones, se deben considerar tamaños de problema adecuados, para así analizar el efecto de los distintos componentes del entorno en las soluciones del problema.

Además, para cada uno de los problemas con los que se trabaja, se generan 10 instancias, en cada cual se obtiene el valor de las dos FO consideradas (*Cmax* , $\sum C_j$ ) para todas sus posibles secuencias.

Los elementos que caracterizan cada uno de los entornos de trabajo son, además del número de trabajos (*n*) y máquinas (*m*), y los tiempos de proceso de los trabajos en cada una de las máquinas ($p_{ij}$), así como la restricción aplicada, que es el tiempo máximo permitido de espera de los trabajos entre máquinas.

Los números de trabajos que se contemplan, para poder realizar la enumeración completa en tiempos de computación razonables, son 5, 6, 7 y 8. De esta forma, se pueden llevar a cabo las dos aplicaciones de la enumeración completa sin emplear un tiempo excesivo. (Por ejemplo, en el caso de *n=9* ya se habrían de analizar *9!=362.880* secuencias, multiplicado por dos veces, por cada combinación de los elementos restantes, y de las FO).

El número de máquinas del entorno no es tan crítico como el de trabajos, pues como éste no afecta a la cantidad de posibles secuencias del problema, y no influye tan drásticamente en la complejidad al resolverlo. Se consideran talleres con 5, 10, 15 y 20 máquinas.

Los tiempos de proceso de los *n* trabajos en cada una de las *m* máquinas se generan de forma aleatoria en cada instancia, tomando valores enteros entre 1 y 99.

Por último, los tiempos máximos de espera permitidos entre máquinas comprenderán desde el caso donde éstas se contemplen y permitan indefinidamente hasta aquél en el que no estén permitidas, pasando por situaciones intermedias donde las esperas se permitan, pero estén limitadas a una cierta cantidad de unidades temporales, que serán 10, 20, 30 y 40.

Así pues, al tener 16 combinaciones diferentes de *n x m* y 6 valores posibles del tiempo de espera permitida, se tienen 96 problemas posibles para cada función objetivo, para cada una de las cuales se generan 10 instancias, que vienen definidas por el valor que toman los tiempos de proceso ($p_{ij}$) generados aleatoriamente.

Para que este procedimiento tenga validez, en las instancias de los distintos problemas se deben tener los mismos tiempos de proceso, de forma que se puedan comparar los resultados. Así, la cuarta instancia del problema *F15|prmu|Cmax* tendrá los mismos valores de los tiempos de proceso $p_{ij}$ que la del problema *F15|prmu,max20|*$\sum C_j$ con igual número de trabajos.

# 5 IMPLEMENTACIÓN

En este trabajo se analiza la influencia de posibles limitaciones en los tiempos máximos de espera de los trabajos entre máquinas, para lo que se deben resolverse los problemas descritos en el capítulo 2.

## *5.1 Adquisición de resultados*

Para implementar dichos modelos se ha usado el lenguaje C, y el entorno de desarrollo integrado libre *Code::Blocks* para generar aplicaciones. También se ha hecho uso de las librerías aportadas por el personal docente de la asignatura "Programación de Operaciones" para el uso de funciones como imprimir por pantalla las componentes de un vector.

A continuación, se procede a explicar brevemente cómo se ha realizado el código (sólo para el *makespan* como objetivo, por simplicidad), para qué sirve cada una de las partes del mismo, y qué resultados se obtienen de él.

El código íntegro se adjuntará en el anexo, así como una explicación más detallada de cómo funciona. En este capítulo se omiten las líneas de código meramente estéticas (saltos de línea, tabulaciones…) que se han usado para lograr una mejor comprensión del mismo.

Para empezar, se incluyen las librerías que se van a usar, y se definen las funciones necesarias, que serán, además de las tres que calculan el *makespan* para los problemas con cada uno de los tipos de restricciones existentes, las que calculan el mínimo y el máximo de dos números enteros dados, la que calcula el factorial de un número entero, y las dos aplicaciones de la enumeración completa. El funcionamiento de todas estas funciones se explica en el anexo.

También se declaran las variables con las que se calcularán los diferentes indicadores explicados en el capítulo 3, como el vector que almacena la secuencia con la que se esté trabajando en cada momento, o la matriz con los tiempos de proceso para la instancia en cuestión.

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>

int max(int a, int b);
int min(int a, int b);
int factorial (int a);
int Cmax(int m, int n, MAT_INT P, VECTOR_INT sec);
int CmaxNOWAIT(int m, int n, MAT_INT P, VECTOR_INT sec);
int CmaxMAXWAIT(int m, int n, MAT_INT P, VECTOR_INT sec, int waitmax);
void enumeracionCompleta1(VECTOR_INT sec, int n, int par_length, int m, MAT_INT P);
void enumeracionCompleta2(VECTOR_INT sec, int n, int par_length, int m, MAT_INT P);
int x1,x2,x3,x4,x5,x6;
int x1max,x2max,x3max,x4max,x5max,x6max;
int x1min,x2min,x3min,x4min,x5min,x6min;
int x1tot,x2tot,x3tot,x4tot,x5tot,x6tot;
float ARPD1,ARPD2,ARPD3,ARPD4,ARPD5,ARPD6;
VECTOR_INT sec;
```

```
MAT_INT secMIN;
VECTOR_INT prob1;
VECTOR_INT prob2;
VECTOR_INT prob3;
VECTOR_INT prob4;
VECTOR_INT prob5;
VECTOR_INT prob6;
MAT_INT P;
```

A continuación, se definen el número de trabajos y máquinas que se tendrán en el entorno en el que se esté trabajando, además de dimensionar algunos de los vectores y matrices que se declararon anteriormente y que serán utilizados posteriormente.

```
void main()
{
int it,i,j;
int n=5;
int m=20;
float fact=factorial(n);
sec=DIM_VECTOR_INT(n+1);
secMIN=DIM_MAT_INT(6,n+1);
prob1=DIM_VECTOR_INT(100);
prob2=DIM_VECTOR_INT(100);
prob3=DIM_VECTOR_INT(100);
prob4=DIM_VECTOR_INT(100);
prob5=DIM_VECTOR_INT(100);
prob6=DIM_VECTOR_INT(100);
P=DIM_MAT_INT(m,n);
```

Se crean las 10 iteraciones, y para cada una de ellas se genera aleatoriamente la matriz de tiempos de proceso. Como la semilla no cambia, sus valores serán idénticos para los problemas con diferentes restricciones pero igual entorno, que es el objetivo.

```
for(it=1;it<=10;it++)
{
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            P[i][j]=rand()%99+1;
        }
    }
```

Para efectuar la primera aplicación de la enumeración completa, se deben inicializar los valores de los mínimos y máximos de los problemas adecuadamente, y definir una secuencia inicial.

El código se ha diseñado para secuencias que empiecen con el valor 0, aunque éste no corresponda a ningún trabajo. Es decir, que para un entorno con 6 trabajos, el vector *sec* tendrá 7 componentes, el primero de los cuales será un 0.

Entonces, se aplica la enumeración completa, y se obtienen los valores mínimos y máximos de la FO en cada uno de los seis problemas. También se halla la secuencia óptima de cada problema, pero no se muestra por pantalla pues no es un dato que se vaya a usar, aunque sí se guarda en memoria de cara a la segunda aplicación de la enumeración completa.

```
printf("ITERACION %d",it);

x1max=x2max=x3max=x4max=x5max=x6max=1;
x1min=x2min=x3min=x4min=x5min=x6min=100000;

for(i=0;i<=n;i++)
    {
    sec[i]=i;
    }

enumeracionCompleta1(sec,n,n+1,m,P);

printf("makespan PFSP: MIN = %d , MAX = %d",x1min,x1max);
printf("makespan max40: MIN = %d , MAX = %d",x2min,x2max);
printf("makespan max30: MIN = %d , MAX = %d",x3min,x3max);
printf("makespan max20: MIN = %d , MAX = %d",x4min,x4max);
printf("makespan max10: MIN = %d , MAX = %d",x5min,x5max);
printf("makespan nwt: MIN = %d , MAX = %d",x6min,x6max);
```

Para aplicar la enumeración completa por segunda vez, hay que inicializar más variables, como por ejemplo los vectores donde se recogerá la función de densidad de probabilidad de cada problema.

```
    x1tot=x2tot=x3tot=x4tot=x5tot=x6tot=0;

    for(i=0;i<100;i++)
        {
        prob1[i]=0;
        prob2[i]=0;
        prob3[i]=0;
        prob4[i]=0;
        prob5[i]=0;
        prob6[i]=0;
        }

    for(i=0;i<=n;i++)
        {
        sec[i]=i;
        }

    enumeracionCompleta2(sec,n,n+1,m,P);
```

5. Implementación

Ahora ya se puede conocer el ARPD de cada problema, así como mostrar por pantalla las funciones de densidad de probabilidad, y el resultado de aplicar la secuencia óptima del PFSP en los otros cinco problemas, así como las óptimas de dichos problemas en el PFSP.

```
ARPD1=((float)x1tot/(float)x1min)*(100/fact)-100;
printf("ARPD PFSP = %f",ARPD1);
printf("Densidad de probabilidad PFSP:");
print_int_vector(prob1,100);
printf("Optima del PFSP en el no-wait = %d",CmaxNOWAIT(m,n,P,secMIN[0]));
printf("Optima del PFSP en el max10 = %d",CmaxMAXWAIT(m,n,P,secMIN[0],10));
printf("Optima del PFSP en el max20 = %d",CmaxMAXWAIT(m,n,P,secMIN[0],20));
printf("Optima del PFSP en el max30 = %d",CmaxMAXWAIT(m,n,P,secMIN[0],30));
printf("Optima del PFSP en el max40 = %d",CmaxMAXWAIT(m,n,P,secMIN[0],40));

ARPD2=((float)x2tot/(float)x2min)*(100/fact)-100;
printf("ARPD max40 = %f",ARPD2);
printf("Densidad de probabilidad max40:");
print_int_vector(prob2,100);
printf("Optima del max40 en el PFSP = %d",Cmax(m,n,P,secMIN[1]));

ARPD3=((float)x3tot/(float)x3min)*(100/fact)-100;
printf("ARPD max30 = %f",ARPD3);
printf("Densidad de probabilidad max30:");
print_int_vector(prob3,100);
printf("Optima del max30 en el PFSP = %d",Cmax(m,n,P,secMIN[2]));

ARPD4=((float)x4tot/(float)x4min)*(100/fact)-100;
printf("ARPD max20 = %f",ARPD4);
printf("Densidad de probabilidad max20:");
print_int_vector(prob4,100);
printf("Optima del max20 en el PFSP = %d",Cmax(m,n,P,secMIN[3]));

ARPD5=((float)x5tot/(float)x5min)*(100/fact)-100;
printf("ARPD max10 = %f",ARPD5);
printf("Densidad de probabilidad max10:");
print_int_vector(prob5,100);
printf("Optima del max10 en el PFSP = %d",Cmax(m,n,P,secMIN[4]));

ARPD6=((float)x6tot/(float)x6min)*(100/fact)-100;
printf("ARPD no-wait = %f",ARPD6);
printf("Densidad de probabilidad no-wait:");
print_int_vector(prob6,100);
printf("Optima del no-wait en el PFSP = %d",Cmax(m,n,P,secMIN[5]));

}
}
```

Tras aplicar el código anterior, se muestran por pantalla los resultados obtenidos para cada una de las iteraciones del problema con el entorno y objetivo correspondientes:

- Valores mínimo y máximo de la FO para cada uno de los problemas, según su restricción.
- ARPD para cada uno de los problemas.
- Función de densidad de probabilidad de las secuencias de cada problema, expresada como un vector de 100 componentes.
- Resultados obtenidos al aplicar la secuencia óptima del PFSP en cada uno de los otros cinco problemas.
- Resultados obtenidos al aplicar las secuencias óptimas de los cinco problemas con esperas limitadas en el PFSP.

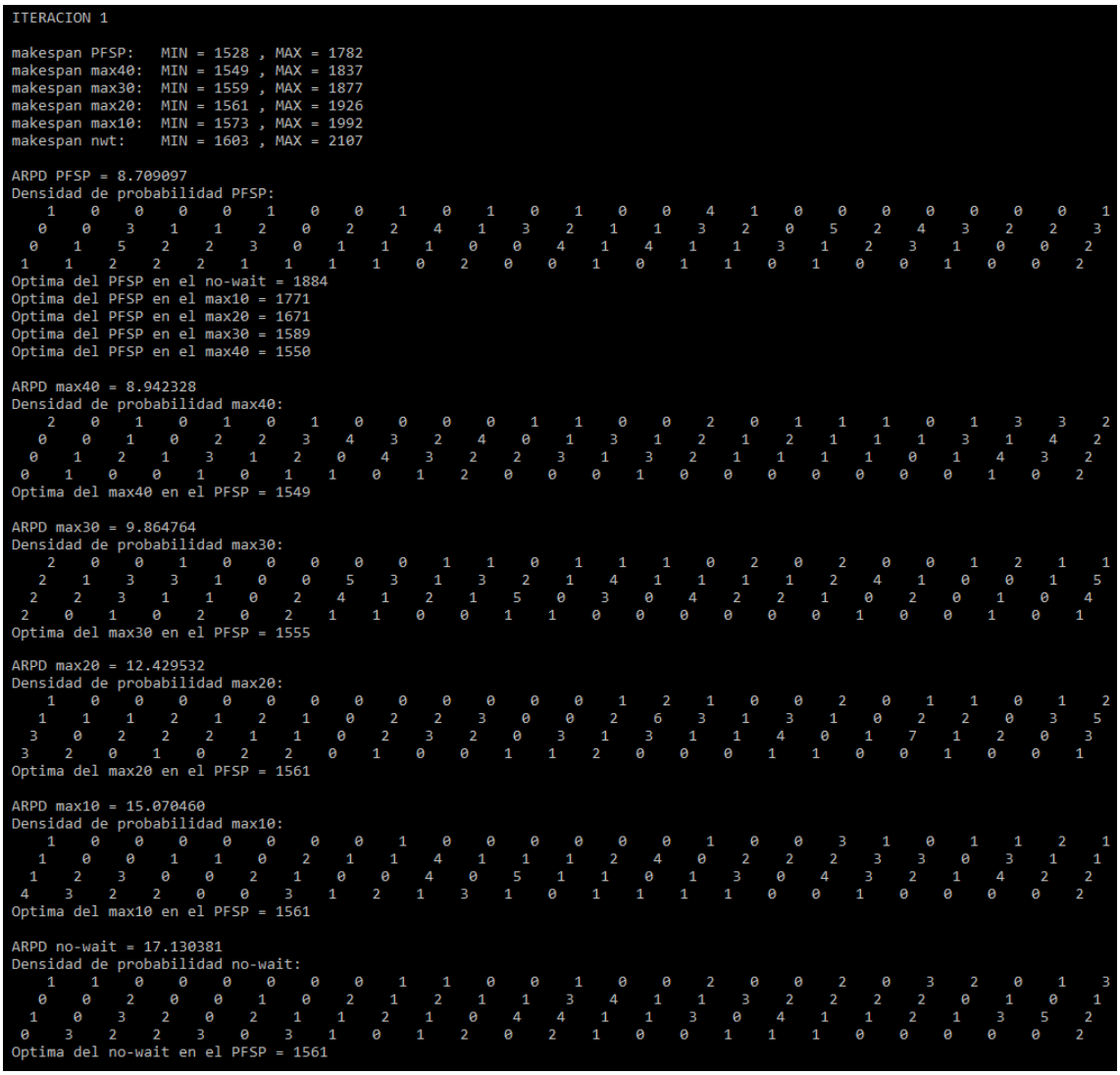


**Figura 5.1** – Resultado de aplicar dos veces la enumeración completa a una instancia usando el código propuesto (Elaboración propia)

Los resultados que se observan en la figura 5.1 corresponden a una de las diez iteraciones que se realizan para el entorno con 5 trabajos y 20 máquinas, que es solamente uno de los dieciséis entornos que se contemplan en el trabajo, como se explicó en el capítulo 4.

Es decir, que se deben recoger 160 conjuntos de resultados tales como el que describe la figura 5.1, para cada uno de los dos objetivos, pues los problemas con el *makespan* como objetivo se estudian de forma independiente a los que minimizan el tiempo total de finalización, aunque luego comparen algunos de sus resultados.

## 5.2 Organización de la información

Con el fin de manejar tal volumen de información, se usa el software Microsoft Excel, ya que permite agrupar los resultados según las restricciones aplicadas, el entorno… así como representarlos gráficamente de forma relativamente sencilla.

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 8 | 16,09 | 916 | 1180 | 1037 | 13,21 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| | | | 9 | 18,83 | 906 | 1232 | 1058 | 16,78 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 10 | 14,09 | 958 | 1226 | 1063 | 10,96 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | | | ***18,17*** | ***890,30*** | | | ***12,51*** | **13** | **2** | **1** | **3** | **4** | **3** | **7** | **2** | **4** |
| | n = 5 | | | ARPD | MIN | MAX | Cmax(PFSP) | inc.% PFSP | 13 | 15 | 16 | 19 | 23 | 26 | 33 | 35 | 39 |
| | | | 1 | 12,88 | 1314 | 1642 | 1417 | 7,84 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| | | | 2 | 17,48 | 1120 | 1440 | 1268 | 13,21 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | | 3 | 15,95 | 1210 | 1612 | 1406 | 16,20 | 1 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |
| | | | 4 | 16,02 | 1185 | 1515 | 1212 | 2,28 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | | 5 | 18,42 | 1191 | 1606 | 1328 | 11,50 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | m = 15 | 6 | 16,20 | 1204 | 1612 | 1469 | 22,01 | 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| | | | 7 | 14,63 | 1355 | 1720 | 1547 | 14,17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | | 8 | 16,14 | 1087 | 1422 | 1142 | 5,06 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | | 9 | 21,22 | 1199 | 1664 | 1271 | 6,01 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 10 | 21,64 | 1107 | 1583 | 1127 | 1,81 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| | | | | ***17,06*** | ***1197,20*** | | | ***10,01*** | **11** | **3** | **2** | **3** | **3** | **2** | **2** | **2** | **4** |
| | | | | ARPD | MIN | MAX | Cmax(PFSP) | inc.% PFSP | 11 | 14 | 16 | 19 | 22 | 24 | 26 | 28 | 32 |
| | | | 1 | 17,13 | 1603 | 2107 | 1884 | 17,53 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | | 2 | 18,55 | 1522 | 2007 | 1709 | 12,29 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | | 3 | 17,80 | 1357 | 1781 | 1357 | 0,00 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | | 4 | 18,80 | 1501 | 2019 | 1842 | 22,72 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | | 5 | 12,81 | 1668 | 2061 | 1808 | 8,39 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | m = 20 | 6 | 12,96 | 1497 | 1858 | 1632 | 9,02 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| | | | 7 | 21,79 | 1416 | 1985 | 1707 | 20,55 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 8 | 18,88 | 1487 | 2044 | 1834 | 23,34 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 0 |

PFSP | no wait | max 10 | max20 | max30 | max40 | COMPARACION | Hoja1

**Figura 5.2** – Extracto del libro de Microsoft Excel donde se recoge la información obtenida a partir del programa (Elaboración propia)

Como se puede apreciar en la figura 5.2, los valores obtenidos en las diez iteraciones de un problema *no-wait* para el mínimo de la FO en cada una de ellas, su ARPD, y el incremento relativo de la FO al usar la secuencia óptima del PFSP se promedian para así hallar los valores característicos de dicho entorno, menos afectados por la aleatoriedad de los tiempos de proceso generados.

Los vectores de densidad de probabilidad se disponen de forma que se sumen, componente a componente, los de las diez iteraciones, para así tener la función de densidad de probabilidad de todas ellas, y a partir de ella calcular la de distribución del problema con el entorno y la restricción correspondiente. Se observa, por ejemplo, que los valores de la iteración 1 del

entorno con 5 trabajos y 20 máquinas son los que aparecen en la figura 5.1, cuando se ejecutó el programa.

El volumen de datos es extremadamente alto como para considerar introducir toda la información a mano, por lo que se hace uso de ficheros. Así, al aplicar el programa, la información que interese extraer aparece en un archivo con formato *.txt*, que se podrá abrir desde un libro de Excel para ya disponer de los datos separados en celdas pero organizados según correspondan en filas y columnas, de forma que puedan manipularse con facilidad y minimizando la probabilidad de equivocación al copiarlos al libro donde se almacenan los resultados.

Para ello, se define y crea el fichero donde se exportarán los datos mediante la siguiente línea de código, que se debe incluir antes de entrar en el bucle de las 10 iteraciones:

```
FILE * fichero=fopen("salida.txt","a");
```

La "a" final hace que, para cada iteración, se añadan los resultados al final del fichero, es decir, después de las iteraciones ejecutadas previamente, para que no se sobrescriban los resultados de iteraciones previas cada vez que se ejecute una nueva.

Por ejemplo, si se quisiera exportar a un fichero los vectores función de densidad de probabilidad del problema *no-wait* para cada una de las diez iteraciones del entorno que se esté estudiando, se añadiría al final de la función principal las siguientes líneas:

```
for(i=0;i<99;i++)
    {
    fprintf(fichero,"%d, ",prob6[i]);
    }
fprintf(fichero,"%d;\n",prob6[i]);
```

De esta forma, cada uno de los 100 componentes del vector estará separado por una coma, mientras que tras el último valor habrá un punto y coma seguido de un salto de línea, que marca el paso a la siguiente iteración. Con esos signos de puntuación, al abrir el fichero exportado desde un libro Excel, aparecerán dispuestos los vectores en filas y columnas perfectos para ser copiados al libro representado en la figura 5.2.

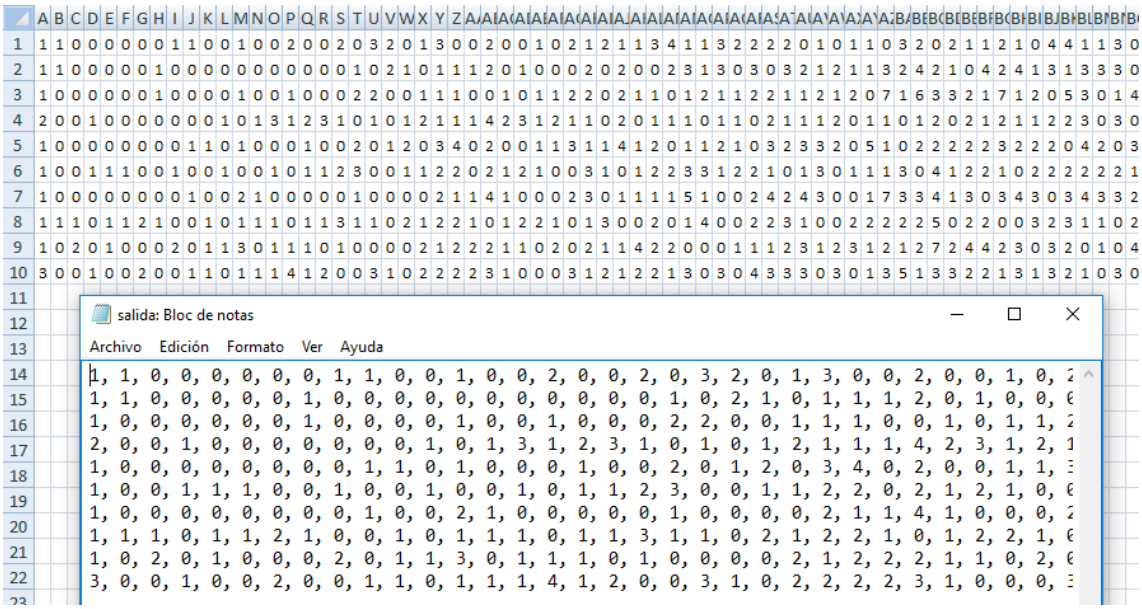


**Figura 5.3** – Ejemplo del resultado de exportar resultados del programa a un fichero, y a un libro Excel (Elaboración propia)

Cuando el objetivo a minimizar es $\sum C_j$ el código es idéntico, con la diferencia evidente de que las funciones no serán *Cmax, CmaxNOWAIT, CmaxMAXWAIT*. Para ello, también se construyen las funciones *sumaCj, sumaCjNOWAIT, sumaCjMAXWAIT*. Estas funciones son exactamente iguales a las anteriores, pues el método para calcular el tiempo de finalización de cada trabajo en cada máquina no cambia, pero lo que devuelven no es el tiempo en el que acaba de procesarse el último trabajo, sino la suma de los tiempos de finalización de todos ellos.

En el anexo se adjuntan todas las funciones usadas en los distintos códigos:

- Primera aplicación de la enumeración completa
- Segunda aplicación de la enumeración completa
- Máximo de dos números enteros
- Mínimo de dos números enteros
- Factorial de un número entero
- *Cmax* obtenido para una secuencia dada aplicada a un PFSP
- *Cmax* obtenido para una secuencia dada aplicada a un problema *no-wait*
- *Cmax* obtenido para una secuencia dada aplicada a un problema con esperas limitadas
- $\sum C_j$ obtenido para una secuencia dada aplicada a un PFSP
- $\sum C_j$ obtenido para una secuencia dada aplicada a un problema *no-wait*
- $\sum C_j$ obtenido para una secuencia dada aplicada a un problema con esperas limitadas

# 6 EVALUACIÓN COMPUTACIONAL

Los códigos implementados aportan una gran cantidad de resultados, que han de ser manejados adecuadamente para su correcta comprensión. Como se comentó en el capítulo 4, se han usado cuatro valores diferentes para el número de trabajos (5, 6, 7, 8) y otros tantos para el número de máquinas (5, 10, 15, 20), teniendo hasta 16 tamaños distintos para cada tipo de problema. Asimismo, se han considerado seis restricciones: desde el caso extremo donde las esperas se permiten sin importar su duración hasta aquél en el que los trabajos que terminan de procesarse en una máquina deben comenzar a tratarse en la siguiente inmediatamente, pasando por cuatro situaciones intermedias, donde se contemplan esperas siempre y cuando no superen una duración máxima preestablecida (10, 20, 30, 40).

Para cada combinación posible de número de trabajos y máquinas se han ejecutado 10 instancias, cada cual compuesta por *n x m* tiempos de proceso de trabajos en las máquinas, y en las que se han evaluado los diferentes problemas usando las seis distintas restricciones, obteniendo resultados de interés como la secuencia que aporta el óptimo en cada una de ellas y el valor de la función de objetivo para ella, por ejemplo.

Posteriormente, se analizará el efecto de las distintas restricciones relacionadas con el tiempo de espera máximo permitido de los trabajos entre máquinas consecutivas. Se compararán los problemas con las distintas restricciones de tiempo de espera, es decir, los problemas *Fm|prmu|Cmax*, *Fm|prmu,max10|Cmax*, … , *Fm|prmu,nwt|Cmax*, así como los problemas que tengan como función objetivo el tiempo total de finalización, $\sum C_j$.

## 6.1 Resultados óptimos de los problemas

Se comienza analizando el valor mínimo de las distintas funciones objetivo, es decir, el caso más beneficioso posible atendiendo al criterio escogido. Para cada entorno y restricción, se ha realizado la media del óptimo de sus 10 iteraciones.

En la tabla 6.1 se puede observar el valor medio del óptimo (en primer lugar teniendo como objetivo *Cmax*) para cada una de las 16 posibles combinaciones de número de máquinas y trabajos, y también para cada uno de los seis tiempos de espera máximos considerados en el proyecto.

| FO: C max | | PFSP | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 507,5 | 510,4 | 515,4 | 518,9 | 524,0 | 535,6 |
| | m = 10 | 841,2 | 850,3 | 855,5 | 862,1 | 875,0 | 890,3 |
| | m = 15 | 1135,6 | 1146,8 | 1154,0 | 1164,3 | 1182,6 | 1197,2 |
| | m = 20 | 1459,0 | 1478,1 | 1484,8 | 1489,9 | 1498,5 | 1521,9 |
| n = 6 | m = 5 | 550,8 | 560,1 | 564,1 | 573,6 | 580,3 | 588,5 |
| | m = 10 | 871,8 | 878,4 | 882,4 | 892,2 | 909,2 | 932,7 |
| | m = 15 | 1142,5 | 1155,8 | 1166,6 | 1181,8 | 1207,5 | 1236,3 |
| | m = 20 | 1440,6 | 1471,1 | 1488,8 | 1503,0 | 1526,9 | 1551,9 |
| n = 7 | m = 5 | 577,9 | 585,4 | 591,4 | 600,2 | 617,8 | 641,8 |
| | m = 10 | 911,5 | 927,9 | 935,9 | 946,8 | 969,5 | 999,4 |
| | m = 15 | 1230,9 | 1257,3 | 1268,2 | 1279,8 | 1307,0 | 1342,6 |
| | m = 20 | 1529,4 | 1546,2 | 1557,7 | 1583,7 | 1626,9 | 1678,3 |
| n = 8 | m = 5 | 643,5 | 658,1 | 666,1 | 678,4 | 696,1 | 716,9 |
| | m = 10 | 955,6 | 971,6 | 981,9 | 1003,7 | 1035,1 | 1076,0 |
| | m = 15 | 1278,5 | 1305,7 | 1325,6 | 1357,0 | 1395,4 | 1440,2 |
| | m = 20 | 1586,0 | 1626,6 | 1650,1 | 1681,7 | 1723,7 | 1777,9 |

**Tabla 6.1** – Valores medios del óptimo para los problemas con FO: *Cmax* (Elaboración propia)

Aunque se trata de conclusiones tan evidentes que no hacía falta generar instancias aleatorias para deducirlas, se comprueba que obviamente el *makespan* es función del número de trabajos a procesar y del número de máquinas por los que estos deben pasar. También se aprecia que, a medida que se reduce el tiempo máximo de espera permitido de los trabajos entre máquina y máquina, el proceso se demora y el valor del *makespan* aumenta.

Para extraer conclusiones más interesantes sobre este problema, se adjunta la tabla 6.2, donde se indica el aumento relativo (en tanto por ciento) del *makespan* de la secuencia óptima de cada problema respecto al valor del mismo en el PFSP de igual número de trabajos y máquinas.

| FO: C max | | PFSP | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 0,0 | 0,6 | 1,6 | 2,2 | 3,3 | 5,5 |
| | m = 10 | 0,0 | 1,1 | 1,7 | 2,5 | 4,0 | 5,8 |
| | m = 15 | 0,0 | 1,0 | 1,6 | 2,5 | 4,1 | 5,4 |
| | m = 20 | 0,0 | 1,3 | 1,8 | 2,1 | 2,7 | 4,3 |
| n = 6 | m = 5 | 0,0 | 1,7 | 2,4 | 4,1 | 5,4 | 6,8 |
| | m = 10 | 0,0 | 0,8 | 1,2 | 2,3 | 4,3 | 7,0 |
| | m = 15 | 0,0 | 1,2 | 2,1 | 3,4 | 5,7 | 8,2 |
| | m = 20 | 0,0 | 2,1 | 3,3 | 4,3 | 6,0 | 7,7 |
| n = 7 | m = 5 | 0,0 | 1,3 | 2,3 | 3,9 | 6,9 | 11,1 |
| | m = 10 | 0,0 | 1,8 | 2,7 | 3,9 | 6,4 | 9,6 |
| | m = 15 | 0,0 | 2,1 | 3,0 | 4,0 | 6,2 | 9,1 |
| | m = 20 | 0,0 | 1,1 | 1,9 | 3,6 | 6,4 | 9,7 |
| n = 8 | m = 5 | 0,0 | 2,3 | 3,5 | 5,4 | 8,2 | 11,4 |
| | m = 10 | 0,0 | 1,7 | 2,8 | 5,0 | 8,3 | 12,6 |
| | m = 15 | 0,0 | 2,1 | 3,7 | 6,1 | 9,1 | 12,6 |
| | m = 20 | 0,0 | 2,6 | 4,0 | 6,0 | 8,7 | 12,1 |

**Tabla 6.2** – Incremento relativo de *Cmax* al aplicar cada una de las restricciones de tiempo máximo de espera (Elaboración propia)

Como ya se podía apreciar en la tabla 6.1, se comprueba que el *makespan* aumenta a medida que se reduce el tiempo máximo de espera. En primer lugar, se ve que no sólo los valores del *makespan* de la secuencia óptima son mayores en los casos donde el número de trabajos es mayor, sino que dichos problemas parece que son los más afectados por la aplicación de restricciones que limitan los tiempos de espera, porque por ejemplo, un PFSP con 5 trabajos y 20 máquinas al que se le aplique una restricción *no-wait* verá empeorado su objetivo en apenas un 4%, mientras que un PFSP con igual número de máquinas pero con 8 trabajos, sufrirá un aumento del valor del *makespan* del 12%. Sin embargo, parece difícil afirmar que pueda existir una relación entre la influencia de la aplicación de las restricciones y el número de máquinas que conforman el taller, pues dependiendo de la restricción y el número de trabajos, hay correlaciones positivas, negativas… pero con márgenes tan pequeños que seguramente dichas variaciones se deban más a la aleatoriedad de las instancias generadas que a que realmente exista una relación entre ambos factores.

Se repite el proceso, volviendo a comparar los problemas aplicando las distintas restricciones, sólo que ahora teniendo como función objetivo el tiempo total de finalización, $\sum C_j$.

| FO: ∑Cj | | PFSP | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 1828,7 | 1832,5 | 1837,6 | 1845,6 | 1856,4 | 1879,0 |
| | m = 10 | 3254,1 | 3258,7 | 3263,2 | 3271,0 | 3295,5 | 3330,0 |
| | m = 15 | 4562,1 | 4571,0 | 4579,9 | 4594,6 | 4621,3 | 4652,3 |
| | m = 20 | 5966,4 | 5994,3 | 6006,8 | 6026,5 | 6062,3 | 6112,6 |
| n = 6 | m = 5 | 2345,7 | 2348,2 | 2357,6 | 2379,3 | 2418,7 | 2467,9 |
| | m = 10 | 4083,5 | 4098,2 | 4112,2 | 4141,3 | 4198,6 | 4258,0 |
| | m = 15 | 5613,8 | 5643,0 | 5670,7 | 5711,6 | 5758,5 | 5816,3 |
| | m = 20 | 7392,8 | 7426,9 | 7453,8 | 7498,1 | 7554,6 | 7616,2 |
| n = 7 | m = 5 | 2861,7 | 2877,8 | 2897,3 | 2924,0 | 2959,8 | 3028,6 |
| | m = 10 | 4900,9 | 4935,0 | 4962,7 | 5016,6 | 5078,1 | 5162,0 |
| | m = 15 | 6835,4 | 6883,3 | 6915,5 | 6972,3 | 7032,3 | 7128,8 |
| | m = 20 | 8709,9 | 8751,7 | 8785,2 | 8841,0 | 8922,6 | 9013,7 |
| n = 8 | m = 5 | 3563,8 | 3590,5 | 3608,8 | 3630,1 | 3680,8 | 3758,4 |
| | m = 10 | 5772,1 | 5806,8 | 5838,0 | 5893,9 | 5984,5 | 6103,4 |
| | m = 15 | 8167,3 | 8223,7 | 8274,1 | 8352,8 | 8488,6 | 8635,6 |
| | m = 20 | 10235,7 | 10335,3 | 10414,7 | 10521,1 | 10635,3 | 10830,2 |

**Tabla 6.3** – Valores medios del óptimo para los problemas con FO: $\sum C_j$ (Elaboración propia)

| FO: ∑Cj | | PFSP | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 0,0 | 0,2 | 0,5 | 0,9 | 1,5 | 2,8 |
| | m = 10 | 0,0 | 0,1 | 0,3 | 0,5 | 1,3 | 2,3 |
| | m = 15 | 0,0 | 0,2 | 0,4 | 0,7 | 1,3 | 2,0 |
| | m = 20 | 0,0 | 0,5 | 0,7 | 1,0 | 1,6 | 2,5 |
| n = 6 | m = 5 | 0,0 | 0,1 | 0,5 | 1,4 | 3,1 | 5,2 |
| | m = 10 | 0,0 | 0,4 | 0,7 | 1,4 | 2,8 | 4,3 |
| | m = 15 | 0,0 | 0,5 | 1,0 | 1,7 | 2,6 | 3,6 |
| | m = 20 | 0,0 | 0,5 | 0,8 | 1,4 | 2,2 | 3,0 |
| n = 7 | m = 5 | 0,0 | 0,6 | 1,2 | 2,2 | 3,4 | 5,8 |
| | m = 10 | 0,0 | 0,7 | 1,3 | 2,4 | 3,6 | 5,3 |
| | m = 15 | 0,0 | 0,7 | 1,2 | 2,0 | 2,9 | 4,3 |
| | m = 20 | 0,0 | 0,5 | 0,9 | 1,5 | 2,4 | 3,5 |
| n = 8 | m = 5 | 0,0 | 0,7 | 1,3 | 1,9 | 3,3 | 5,5 |
| | m = 10 | 0,0 | 0,6 | 1,1 | 2,1 | 3,7 | 5,7 |
| | m = 15 | 0,0 | 0,7 | 1,3 | 2,3 | 3,9 | 5,7 |
| | m = 20 | 0,0 | 1,0 | 1,7 | 2,8 | 3,9 | 5,8 |

**Tabla 6.4** – Incremento relativo de $\sum C_j$ al aplicar cada una de las restricciones de tiempo máximo de espera (Elaboración propia)

Aparte del hecho de que los valores de $\sum C_j$ son evidentemente mucho mayores que los que tomaba *Cmax*, pues ahora se tiene en cuenta el instante de finalización del procesado de todos los trabajos y no sólo del último, parece repetirse la tendencia en las dos tablas anteriores, pues todo lo se podía observar en los problemas con el *makespan* como objetivo parece cumplirse también en los problemas con el otro objetivo. No obstante, para una mejor comprensión de los problemas, se ha analizado el incremento relativo del valor de cada una de las funciones objetivo cuando se aplica una restricción *no-wait*, al ser éste el caso más extremo respecto al PFSP, obteniéndose el siguiente resultado:

| | Cmax | ∑Cj |
|---|---|---|
| n = 5 | 5,3 | 2,4 |
| n = 6 | 7,4 | 4,0 |
| n = 7 | 9,9 | 4,7 |
| n = 8 | 12,2 | 5,7 |
| m = 5 | 8,7 | 4,8 |
| m = 10 | 8,8 | 4,4 |
| m = 15 | 8,8 | 3,9 |
| m = 20 | 8,5 | 3,7 |

**Tabla 6.5** – Incremento relativo de cada FO al aplicar la restricción *no-wait* a un PFSP (Elaboración propia)

Se han promediado los valores de los incrementos correspondientes a cada entorno que aparecen en las tablas 6.2 y 6.4, y la primera conclusión que se observa en la tabla 6.5 es que, como ya se pudo ver anteriormente, el incremento de la FO en cuestión es independiente del número de máquinas, pero no así del número de trabajos.

Estas dos últimas deducciones se pueden comprobar gráficamente en las siguientes figuras:

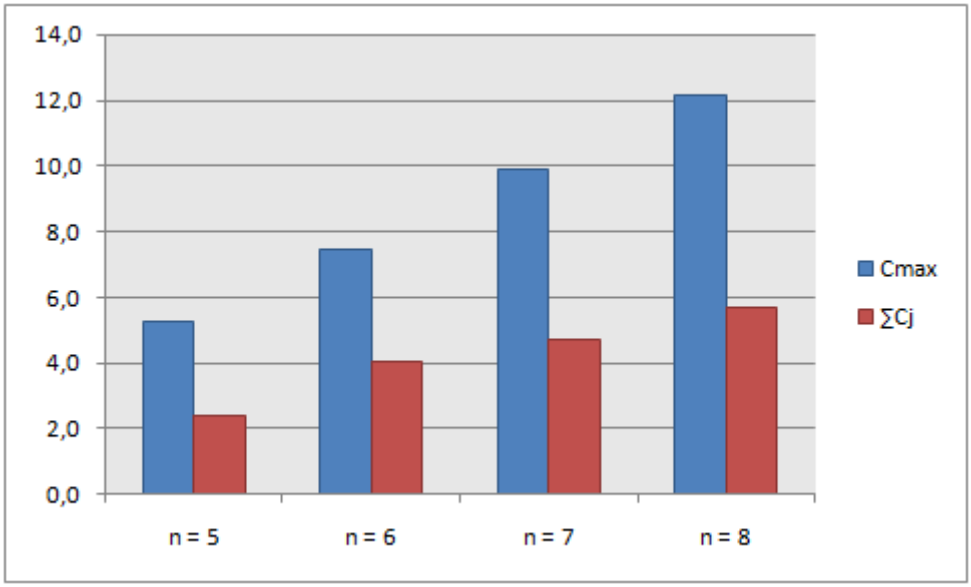


**Figura 6.1** – Incremento relativo de cada FO al aplicar la restricción *no-wait* a un PFSP, dependiendo del número de trabajos del problema (Elaboración propia)

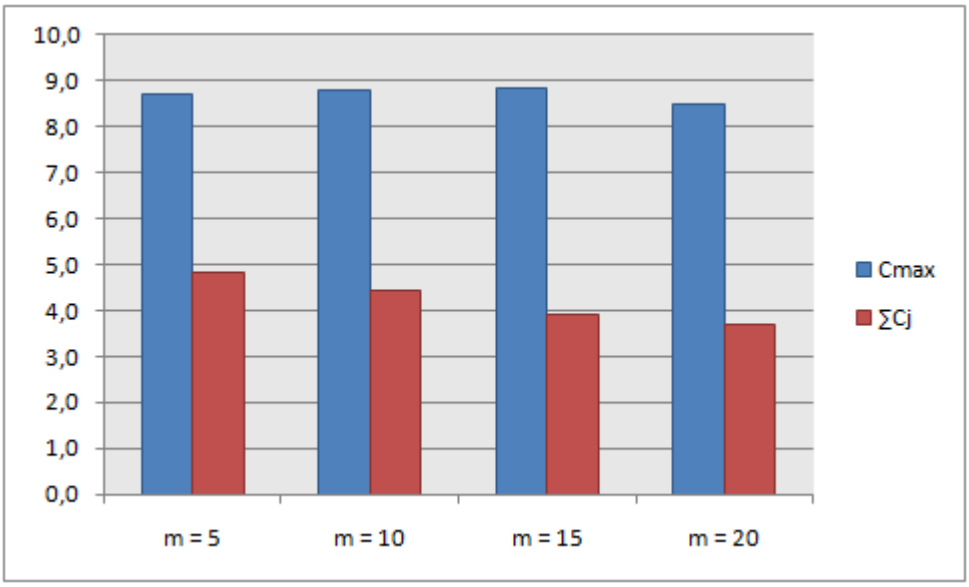


**Figura 6.2** – Incremento relativo de cada FO al aplicar la restricción *no-wait* a un PFSP, dependiendo del número de máquinas del problema (Elaboración propia)

Otra información a extraer de la tabla 6.5 es que los problemas donde el objetivo consiste en minimizar el *makespan* son mucho más sensibles a restricciones que limiten el tiempo de espera, pues se aprecia como el incremento relativo de la FO en ellos es aproximadamente el doble que el que se produce en aquéllos donde el objetivo es minimizar el tiempo total de finalización, para cualquier entorno de trabajo y tiempo máximo de espera permitido. Aunque podría no ser del todo acertado comparar el incremento de los valores de las dos FO, pues una solamente depende del tiempo de finalización de un trabajo, mientras que la otra lo hace de todos los trabajos, tomando así valores considerablemente mayores.

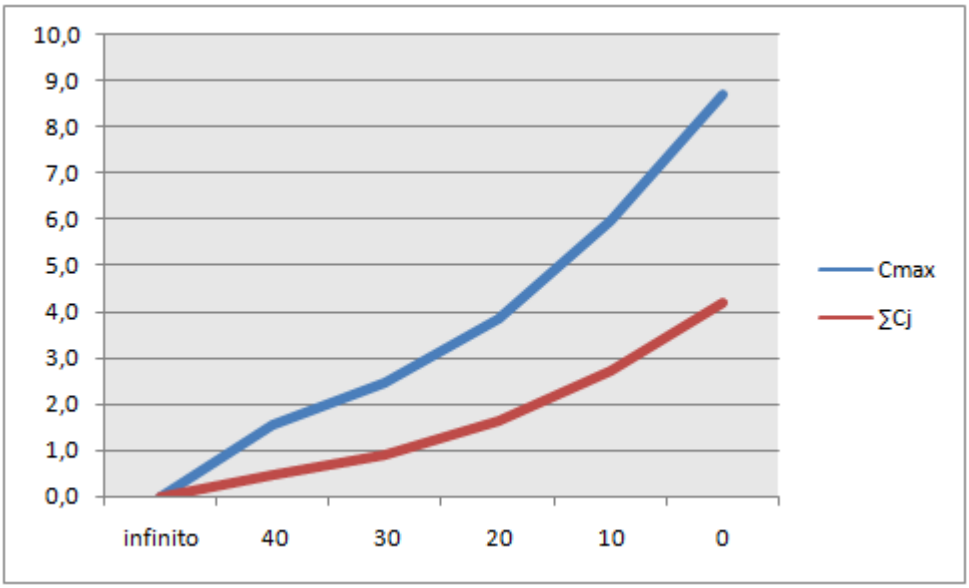


**Figura 6.3** – Incremento relativo medio de cada FO según el tiempo de espera máximo permitido para los trabajos entre dos máquinas consecutivas (Elaboración propia)

## 6.2 Dificultad de los problemas

Si bien en el apartado anterior sólo se tenía en cuenta la mejor secuencia de trabajos para cada instancia, pues sólo eran los valores óptimos de las FO en cada una de ellas lo que se estudiaba, en éste se analiza la dispersión de los valores que toman las FO para todas las secuencias en cada uno de los problemas, según la restricción aplicada y el entorno de trabajo.

La información que se presenta a continuación ayuda a identificar en qué situaciones hallar la secuencia óptima (o una que dé un resultado próximo a ella) es fundamental, o si por el contrario hay una alta probabilidad de obtener una solución lo suficientemente buena sin la necesidad de tener que hallar dicha secuencia, algo que para un número de trabajos y/o máquinas lo suficientemente elevado puede resultar muy complicado.

### 6.2.1 FO: *Cmax*

Para estudiar la distribución de los resultados, usaremos el concepto del ARPD, que como ya se explicó en el capítulo 3, indica el incremento relativo medio de la FO para todas las posibles secuencias de un problema respecto al valor que toma la FO con la secuencia óptima en cada instancia. En otras palabras, muestra cuán peor sería, de media, el valor de la FO que daría una secuencia aleatoria en comparación al que se obtendría con la óptima.

| FO: C max | | PFSP | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 17,4 | 19,0 | 19,2 | 20,4 | 22,0 | 22,8 |
| | m = 10 | 11,8 | 12,4 | 13,2 | 14,5 | 16,0 | 18,2 |
| | m = 15 | 10,4 | 11,2 | 12,0 | 13,1 | 14,4 | 17,1 |
| | m = 20 | 8,2 | 8,7 | 9,8 | 11,7 | 14,2 | 16,6 |
| n = 6 | m = 5 | 19,2 | 21,0 | 22,2 | 23,1 | 25,3 | 28,4 |
| | m = 10 | 13,9 | 15,8 | 17,1 | 18,4 | 19,9 | 21,6 |
| | m = 15 | 14,6 | 15,7 | 16,3 | 17,3 | 18,3 | 20,3 |
| | m = 20 | 12,4 | 12,2 | 12,3 | 13,4 | 14,6 | 16,7 |
| n = 7 | m = 5 | 23,1 | 25,1 | 26,1 | 27,2 | 27,5 | 27,4 |
| | m = 10 | 16,6 | 18,9 | 20,6 | 23,0 | 25,3 | 27,8 |
| | m = 15 | 13,8 | 15,5 | 17,2 | 19,9 | 22,4 | 25,4 |
| | m = 20 | 12,5 | 15,0 | 16,6 | 18,2 | 19,9 | 22,3 |
| n = 8 | m = 5 | 22,9 | 25,9 | 27,2 | 28,6 | 30,1 | 32,1 |
| | m = 10 | 19,7 | 23,3 | 25,2 | 26,6 | 28,3 | 30,0 |
| | m = 15 | 16,3 | 18,9 | 20,1 | 21,5 | 23,3 | 25,7 |
| | m = 20 | 13,9 | 16,2 | 17,6 | 19,6 | 21,8 | 24,3 |

**Tabla 6.6** – ARPD de los problemas con FO: *Cmax* para los distintos números de trabajos y máquinas, y restricciones de tiempos máximos de espera (Elaboración propia)

La tabla 6.6 muestra claramente que el ARPD:

- Es mayor cuando crece el número de trabajos a procesar.
- Es menor cuando aumenta el número de máquinas que componen el taller de flujo.
- Se incrementa cuando las limitaciones de los tiempos de espera son más estrictas.

Una interesante interpretación de la tabla es que si el taller está formado por una cantidad considerablemente alta de máquinas, hallar la secuencia óptima no tiene una importancia tan capital. Por ejemplo, suponiendo el caso de un taller con 20 máquinas que tenga que procesar 8 trabajos sin restricciones de espera, una secuencia cualquiera de entre las 40.320 posibles daría una solución solamente un 13,9% peor que la óptima, mientras que si apenas fueran 5 máquinas y las esperas dejaran de estar permitidas, ese valor se dispararía hasta el 32,1%.

Para estudiar el efecto del número de trabajos y máquinas, se promedian los valores correspondientes de la tabla 6.6, obteniéndose las siguientes gráficas:

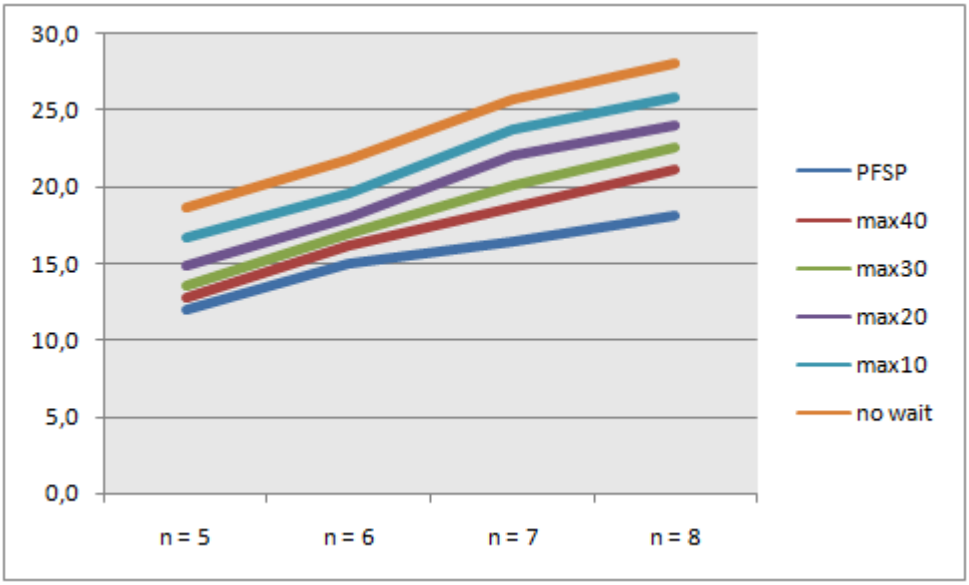


**Figura 6.4** – ARPD de los problemas con FO: *Cmax*, según el número de trabajos (Elaboración propia)

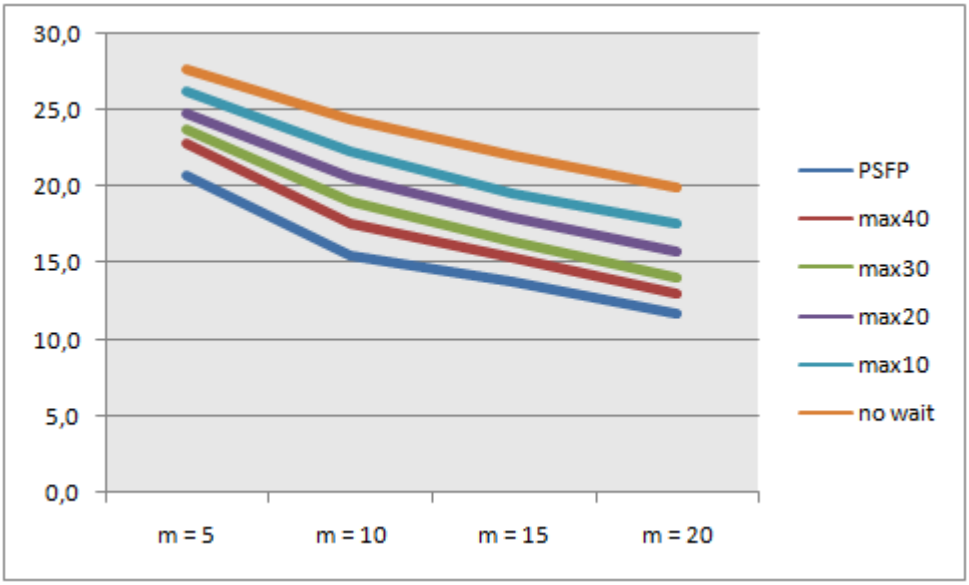


**Figura 6.5** – ARPD de los problemas con FO: *Cmax*, según el número de máquinas (Elaboración propia)

Se comprueba que el problema con restricción *no-wait* es el que tiene un mayor ARPD, mientras que el problema que únicamente tiene la de permutación (PFSP) es el que presenta menores desviaciones en su FO. Los problemas con esperas permitidas pero limitadas están en una situación intermedia, ordenados según dicho tiempo máximo de espera permitido. También se aprecian las relaciones ya mencionadas entre el ARPD y el número de trabajos y máquinas, respectivamente, que ya se podían anticipar tras obtener la tabla 6.6, pero las gráficas parecen indicar además que se dichas relaciones tienen un comportamiento lineal.

Otra forma de estudiar los distintos valores que toman las FO dependiendo de la secuencia de trabajos, sin usar el indicador ARPD, es usando distribuciones de densidad y probabilidad. Para ello, para cada instancia, se hallan las secuencias extremas (la mejor y la peor atendiendo a la FO considerada), y los valores que ésta toma para dichas secuencias marcan el inicio y el final de un intervalo que se divide en 100 tramos iguales, y en el que se encuentran los valores de la FO para todas las demás posibles secuencias de la instancia correspondiente.

Se presentan a continuación dos funciones de densidad de probabilidad para dos problemas claramente diferenciados, de forma que los efectos que las esperas y el número de trabajos o máquinas tienen sobre las soluciones, que ya se pueden deducir a partir de toda la información anterior, se reflejen en ellas.

El primero de los dos problemas a estudiar será el que permite esperas indefinidas, con el mayor tamaño del taller contemplado (20 máquinas), un número reducido de trabajos (6), y que trata de minimizar el *makespan*. Se construye la función de densidad de probabilidad de sus *6!=720* secuencias posibles tal como se explicó en el capítulo 3, es decir, acotando los valores que toma el objetivo para la mejor y peor secuencia, y distribuyendo los que se obtienen siguiendo el resto de secuencias en los cien tramos equivalentes en los que se divide dicho intervalo.

El segundo de los problemas analizados será considerablemente diferente, para así apreciar las diferencias en el comportamiento de la distribución de las soluciones. Se tratará de un problema donde se permiten esperas pero de hasta un máximo de 20 unidades temporales, con un taller con la mitad de máquinas (10), más trabajos a procesar (8), pero el mismo objetivo: minimizar el *makespan*. Para este problema se tienen *8!=40.320* secuencias diferentes, que son 56 veces más que las que se tenían en el primer problema. Por ello, para poder comparar los resultados, no se representará la cantidad de secuencias que hay en cada intervalo, sino la proporción respecto al número total de secuencias posibles que tiene el problema.

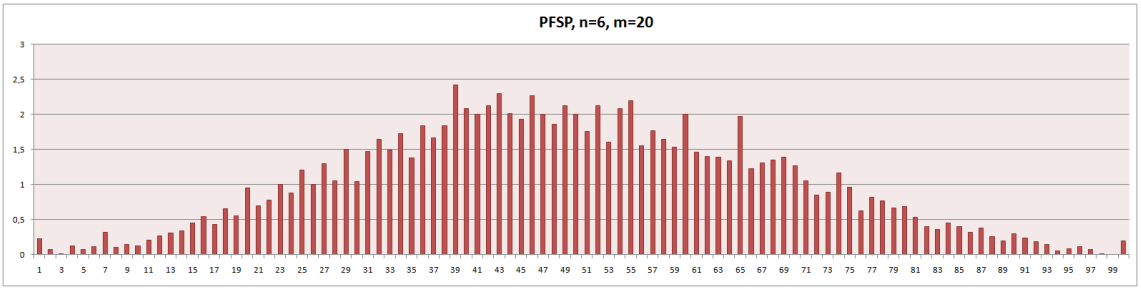


**Figura 6.6** – Función de densidad de probabilidad de las secuencias para un problema *F20|prmu|Cmax* con 6 trabajos (Elaboración propia)

El hecho de que el problema estudiado en la figura 6.6 sea uno con pocos trabajos (y por consiguiente con no muchas posibles secuencias), conlleva que, aunque se pueda intuir la distribución normal de probabilidades, la forma de la Campana de Gauss no sea tan evidente como la que se puede apreciar en la figura 6.7, pues la aleatoriedad de los tiempos de proceso generados hace que se produzcan ciertos picos a lo largo del intervalo.

La figura 6.6 tiene una peculiaridad que no ocurre en la 6.7, y es que las secuencias que se encuentran en el primer y último tramo son bastantes más que en sus tramos más inmediatamente cercanos. Esto es así porque, para cada instancia, hay al menos una secuencia (la mejor) que se encuentra en el primer tramo, y otra (la peor) que está en el tramo 100. Dado que al tener apenas 6 trabajos existen 720 secuencias posibles, se sabe que al menos una de esas 720 estará en cada uno de los extremos, que es una probabilidad considerable para tratarse de un intervalo dividido en cien tramos. En la figura 6.7, al tratarse de un problema con 8 trabajos, ya no se ven valores tan poco intuitivos en los tramos extremos. Aunque es posible que en el primer tramo haya un mayor número de secuencias que en el segundo debido al motivo que se explica anteriormente, 1 secuencia entre 40.320 posibles se trata de una probabilidad enormemente inferior en comparación a la que existe en los tramos del intervalo más alejados de los extremos.

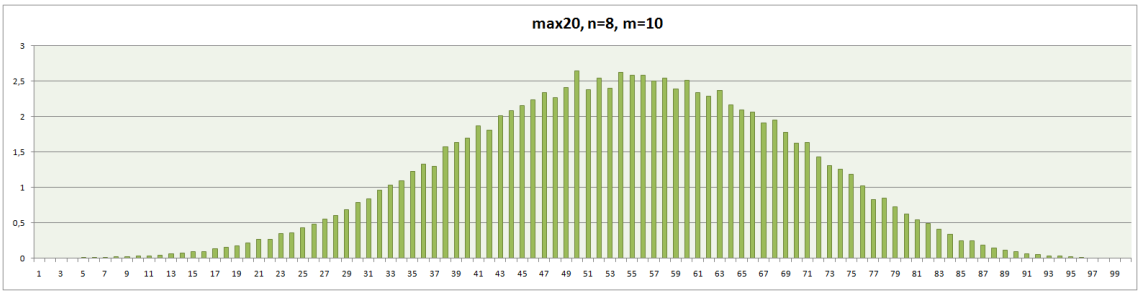


**Figura 6.7** – Función de densidad de probabilidad de las secuencias para un problema *F10|prmu,max20|Cmax* con 8 trabajos (Elaboración propia)

Asumiendo que las dos distribuciones anteriores se comportasen como distribuciones normales, se puede ver que la media de la normal de la figura 6.7 se encuentra claramente a la derecha de la de la figura 6.6, algo que no es sorprendente, pues el problema max20 con 8 trabajos y 10 máquinas tenía un ARPD claramente mayor al del PFSP con 6 trabajos y 20 máquinas. Ese ARPD superior indicaba mayor lejanía respecto al óptimo, y es precisamente lo que indica esa función de densidad desplazada hacia la derecha.

Otra forma de representar la distribución de las secuencias de los problemas es usando la probabilidad acumulada, de manera que en lugar de obtener la probabilidad de que una secuencia cualquiera se encuentre en uno de los cien tramos del intervalo, lo que se obtiene es la de que esté en un tramo que parta del óptimo, es decir, del inicio del eje x.

Sumando las probabilidades de cada uno de los tramos de la distribución de la figura 6.7, se obtiene la siguiente función de distribución de probabilidad acumulada.

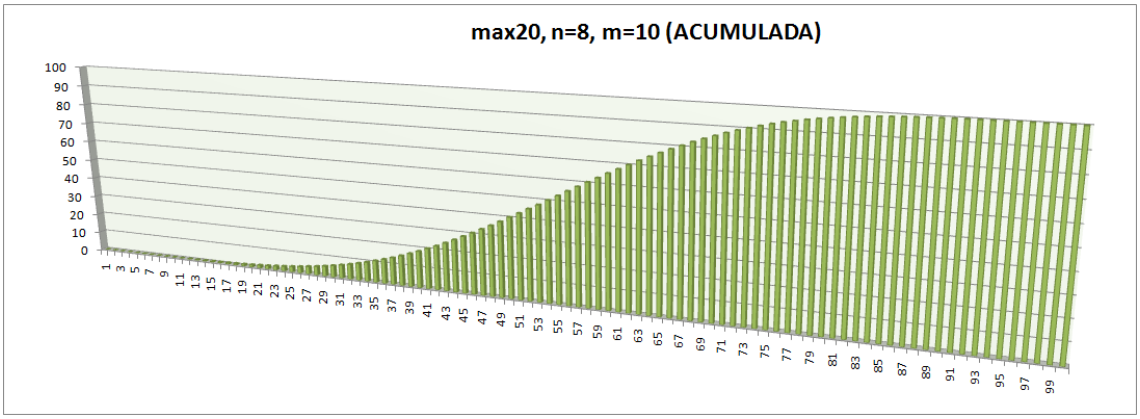


**Figura 6.8** – Función de distribución de probabilidad de las secuencias para un problema F10|*prmu*,*max20*|*Cmax* con 8 trabajos (Elaboración propia)

Se puede ver en la figura 6.8 que sólo el 40% de todas las secuencias posibles se encuentran en los 50 primeros tramos de los 100 de los que se compone el intervalo, es decir, que si se escogiese una secuencia aleatoria de 8 trabajos para este problema concreto, sería más probable que el resultado obtenido estuviera más cerca del de la peor secuencia posible que del *makespan* que diera la secuencia óptima. Conviene recordar que estos 100 tramos simplemente indican la distancia relativa entre los valores de la FO para la mejor y la peor secuencia, pero no reflejan el incremento porcentual del valor de la misma.

Como se quiere estudiar el comportamiento general de los problemas para cada tipo de restricción, es necesario hallar las funciones de distribución de probabilidad para cada combinación de trabajos y máquinas, aplicando a cada una de ellas las distintas limitaciones de tiempos de espera. Dichas distribuciones se deben normalizar para que indiquen probabilidad y no el número de secuencias que se encuentran en cada uno de los cien tramos (como ya se ha hecho en las figuras 6.6 y 6.7), para que así sea indiferente el número de posibles secuencias que haya en cada problema, algo que como ya se ha visto se incrementa enormemente a medida que se aumenta el número de trabajos a procesar.

Una vez normalizadas todas las distribuciones, y combinadas las de todos los problemas con idéntica limitación de tiempos de espera, se obtiene la representación la distribución acumulada de probabilidad para los problemas con cada una de las restricciones, como se muestra en la figura 6.9.

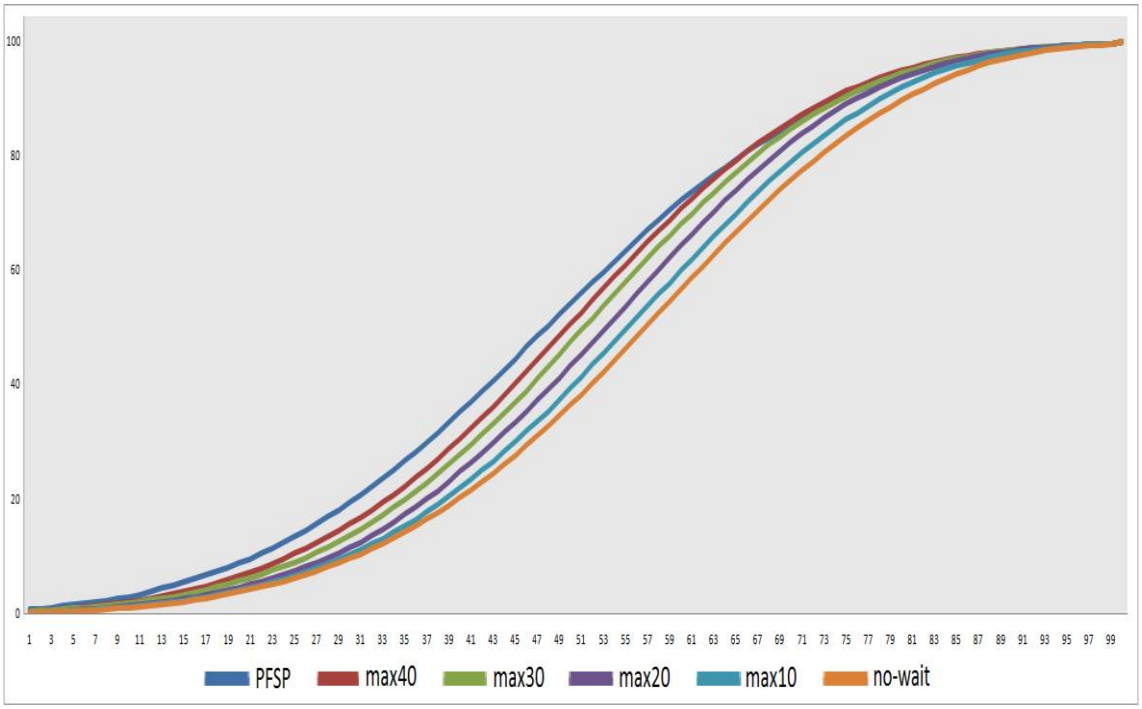


**Figura 6.9** – Funciones de distribución de probabilidad de las secuencias para los problemas con cada una de las restricciones de tiempos máximos de espera, con FO: *Cmax* (Elaboración propia)

Evidentemente, cada curva tiene una forma similar a la de la figura 6.8, pues se trata del mismo tipo de distribución de probabilidad acumulada. La utilidad de esta gráfica es que se puede comprobar, una vez más, que en los problemas con una limitación estricta del tiempo de espera entre máquinas es más difícil aproximarse al óptimo que en un PFSP.

### 6.2.2 FO: $\sum C_j$

Obtenemos, al igual que se hizo para el objetivo del *makespan*, el ARPD para cada uno de los problemas estudiados, para ver lo alejadas del óptimo que están las secuencias para cada uno de los problemas.

| FO: ∑Cj | | PFSP | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 19,8 | 20,5 | 20,9 | 21,4 | 22,1 | 22,4 |
| | m = 10 | 13,9 | 14,5 | 15,0 | 15,8 | 16,5 | 17,5 |
| | m = 15 | 12,1 | 12,7 | 13,1 | 13,7 | 14,5 | 15,8 |
| | m = 20 | 10,6 | 10,9 | 11,3 | 11,9 | 12,8 | 13,9 |
| n = 6 | m = 5 | 20,0 | 21,7 | 22,2 | 22,5 | 22,5 | 22,8 |
| | m = 10 | 13,0 | 13,7 | 14,1 | 14,6 | 14,9 | 15,9 |
| | m = 15 | 12,6 | 13,0 | 13,2 | 13,5 | 14,3 | 15,6 |
| | m = 20 | 8,6 | 8,9 | 9,2 | 9,5 | 10,1 | 11,2 |
| n = 7 | m = 5 | 21,4 | 22,4 | 22,7 | 23,2 | 24,0 | 24,2 |
| | m = 10 | 15,1 | 16,2 | 17,0 | 17,7 | 19,1 | 20,8 |
| | m = 15 | 13,7 | 14,7 | 15,4 | 16,3 | 18,0 | 19,9 |
| | m = 20 | 12,6 | 13,7 | 14,4 | 15,3 | 16,8 | 18,9 |
| n = 8 | m = 5 | 21,0 | 23,0 | 24,1 | 25,6 | 27,0 | 28,2 |
| | m = 10 | 17,7 | 19,5 | 20,5 | 21,6 | 22,9 | 24,4 |
| | m = 15 | 13,2 | 14,6 | 15,4 | 16,3 | 17,3 | 18,9 |
| | m = 20 | 13,2 | 14,3 | 14,8 | 15,8 | 17,2 | 18,6 |

**Tabla 6.7** – ARPD de los problemas con FO: $\sum C_j$ para los distintos números de trabajos y máquinas, y restricciones de tiempos máximos de espera (Elaboración propia)

Se puede apreciar, al igual que ocurría con la otra FO, que el valor del ARPD es mayor para un taller pequeño, y que aumenta cuando también lo hace el número de trabajos. Sin embargo, para *n=6* los problemas presentan un ARDP algo menor que para *n=5*, algo que se acentúa para los entornos con 10 máquinas, y especialmente cuando hay 20 máquinas.

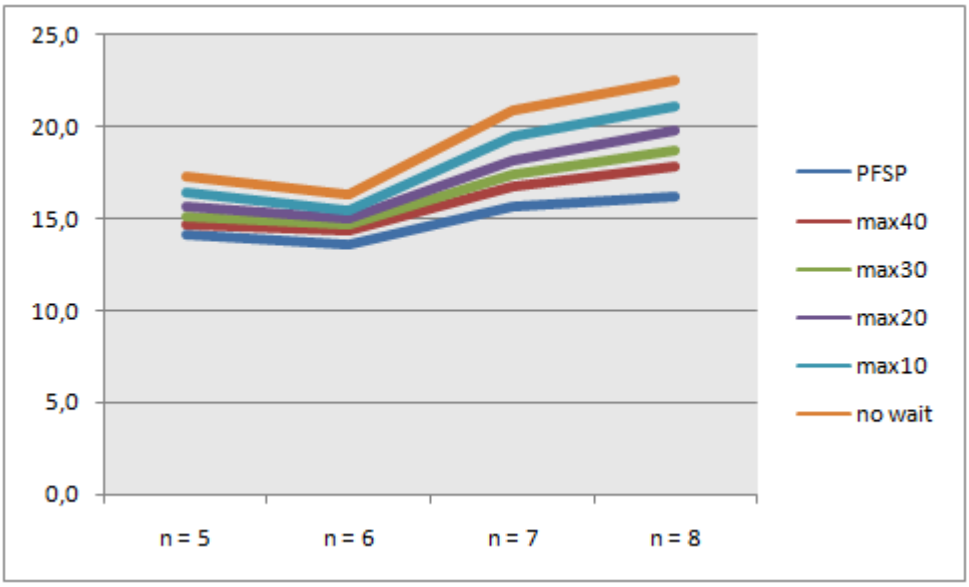


**Figura 6.11** – ARPD de los problemas con FO: $\sum C_j$, dependiendo del número de trabajos a procesar (Elaboración propia)

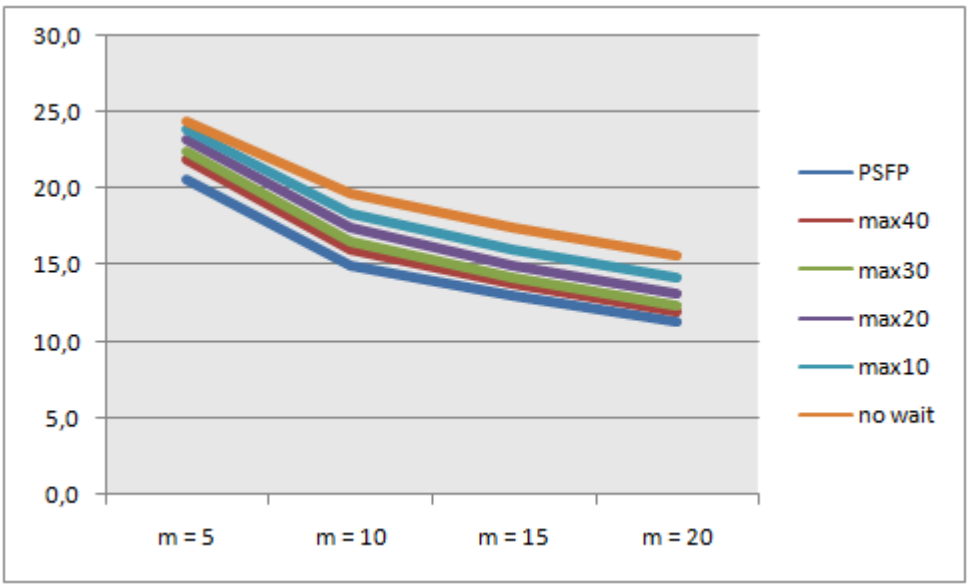


**Figura 6.12** – ARPD de los problemas con FO: $\sum C_j$, dependiendo del número de máquinas que formen el taller de flujo (Elaboración propia)

Al igual que ocurría con el *makespan*, es más difícil acercarse al óptimo a medida que las restricciones que limitan el tiempo máximo de espera entre máquinas se hacen más estrictas, teniendo lugar una mayor diferencia entre los problemas PFSP y *no-wait* cuando el número de trabajos crece. Dicha diferencia, por otra parte, parece independiente del número de máquinas.

Se analiza la distribución de las secuencias según el valor de $\sum C_j$ para los mismos dos problemas analizados en el punto anterior, y como se puede observar, las distribuciones tienen un aspecto muy similar a las que se construyeron para el caso del *Cmax*.

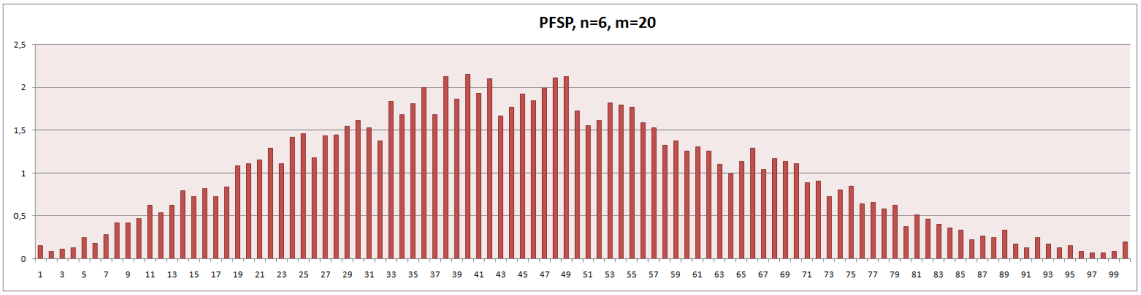


**Figura 6.13** – Función de densidad de probabilidad de las secuencias para un problema F20|prmu|$\sum C_j$ con 6 trabajos (Elaboración propia)

Una pequeña diferencia es que la función representada en la figura 6.14 es algo más asimétrica que la que muestra la figura 6.7, pues la media de la Campana de Gauss a la que se asemeja la función de densidad de probabilidad está situada algo más a la derecha, lo que indicaría que para el tiempo total de finalización las soluciones están, por lo general, más alejadas del óptimo que cuando se usaba el *makespan*.

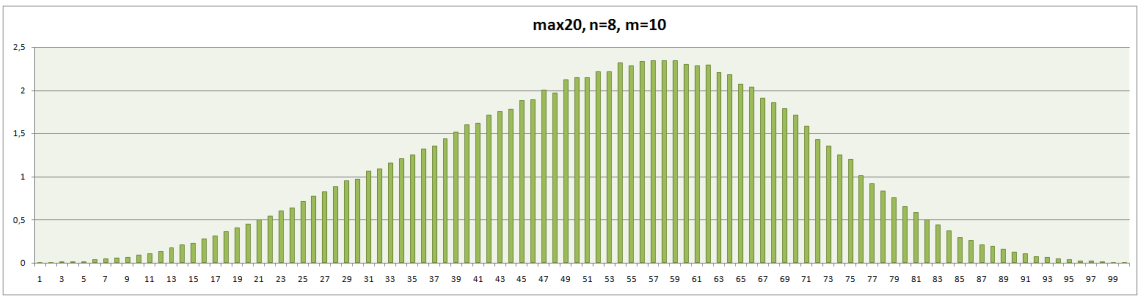


**Figura 6.14** – Función de densidad de probabilidad de las secuencias para un problema F10|prmu, max20|$\sum C_j$ con 8 trabajos (Elaboración propia)

En esta ocasión, se muestra en la figura 6.15 la función de distribución acumulada de probabilidad para el problema PFSP con 6 trabajos y 20 máquinas, pero tiene una forma similar a la que se construyó para el otro problema con la otra FO, representada en la figura 6.8.

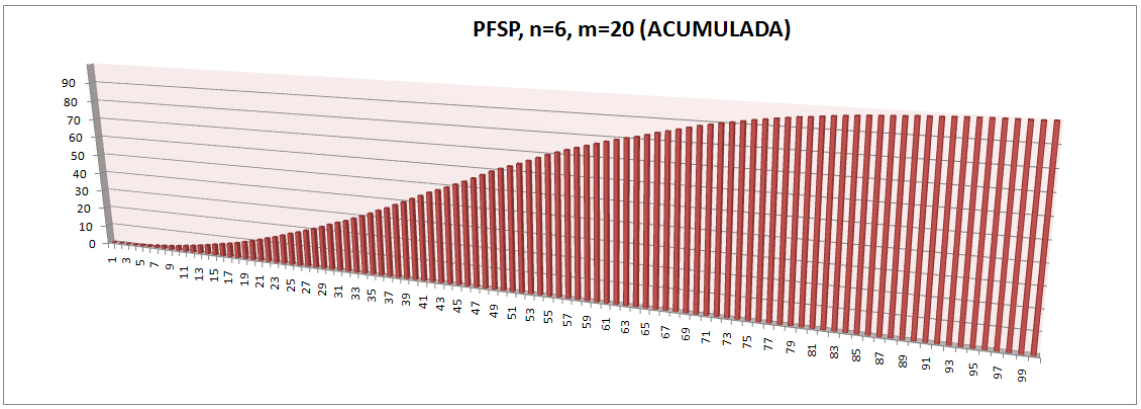


**Figura 6.15** – Función de distribución de probabilidad de las secuencias para un problema *F20|prmu|Cmax* con 6 trabajos (Elaboración propia)

La diferencia más clara e importante es que en la figura 6.15 se empiezan a alcanzar valores altos bastante antes que en la 6.8, pues se comprobó que en el problema representado en la primera las soluciones estaban más próximas al óptimo, ya que el valor de su ARPD era menor. No obstante, la forma de estas distribuciones de probabilidad acumuladas son bastante similares para todos los problemas con este objetivo, pues se vio que las diferencias entre sus ARPD no eran especialmente grandes. Así que, de nuevo, se superponen las distribuciones acumuladas para todas las restricciones estudiadas en una sola figura.

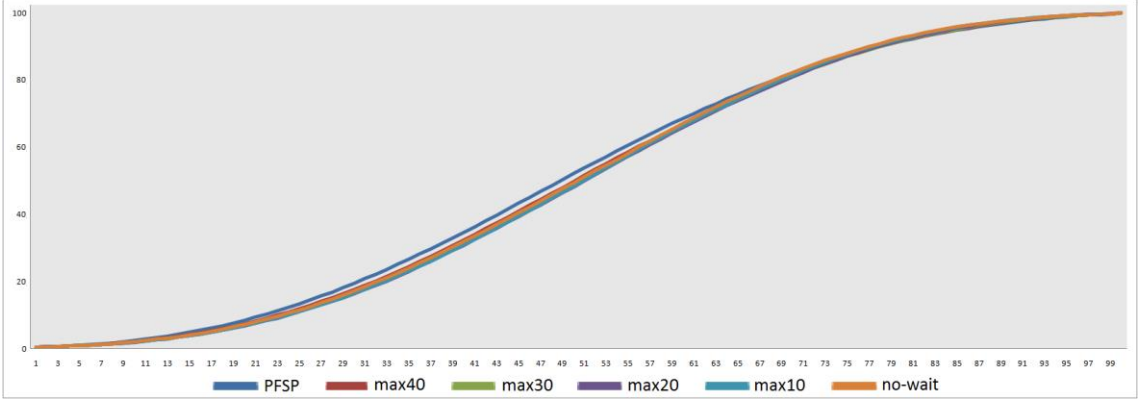


**Figura 6.16** – Funciones de distribución de probabilidad de las secuencias para los problemas con cada una de las restricciones de tiempos máximos de espera, con FO: $\sum C_j$ (Elaboración propia)

De la figura 6.16 se puede extraer con certeza que el PFSP es el problema menos difícil de los seis, pues sus secuencias están ligeramente más próximas a la óptima que en los otros. Sin embargo, entre dichos problemas no existe una gran diferencia, lo que significa que la limitación de los tiempos de espera no parece hacer una gran diferencia en este sentido para problemas con $\sum C_j$ como objetivo, y la probabilidad de encontrar soluciones próximas al óptima no dependerá tanto de la ella.

### 6.2.3 Comparación entre problemas con ambos objetivos

Si bien en los puntos 6.2.1 y 6.2.2 se ha estudiado la influencia del número de trabajos, máquinas y el tipo de restricción en los problemas con cada uno de los objetivos, en este punto se analiza la diferencia entre los problemas con idénticos tamaños de problema y restricciones, pero distinto objetivo.

En primer lugar, se comparan los valores del ARPD obtenidos para cada una de las seis posibles restricciones. Para ello, se promedian los 16 valores del ARPD que se hallaron para cada uno de los 16 tamaños de problema estudiados en las tablas 6.6 y 6.7, todo ello para los seis tipos de problemas distintos que se contemplan según la limitación de los tiempos de espera de los trabajos.

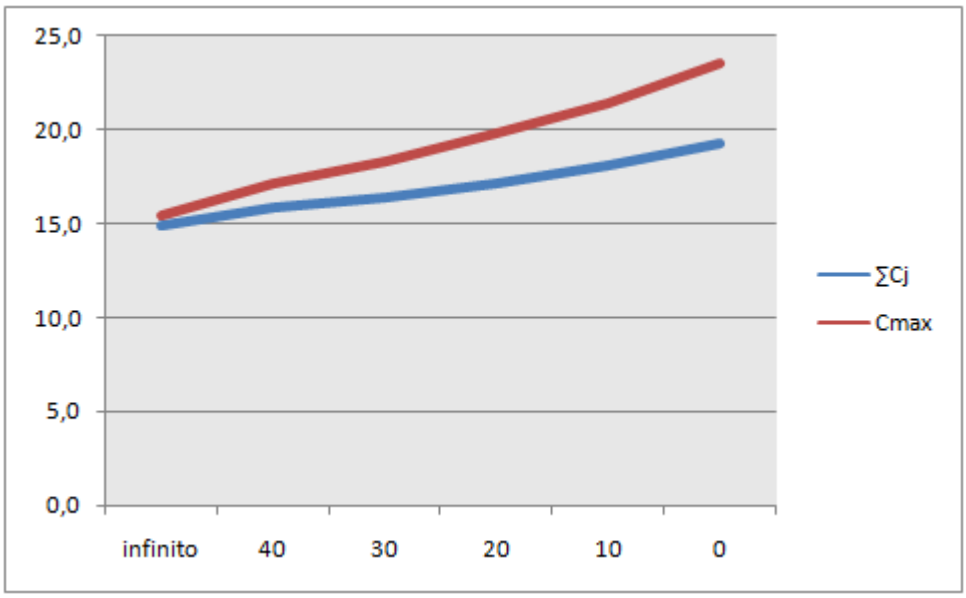


**Figura 6.17** – ARPD de los problemas (eje y) con cada uno de los dos objetivos, para cada uno de los valores contemplados del tiempo de espera permitido máximo (eje x) (Elaboración propia)

Como ya se observó en los puntos anteriores, limitar las esperas hace que no sólo aumente el *makespan* y el tiempo total de finalización, sino que también hace que las secuencias estén más alejadas del óptimo. Además, la figura 6.17 muestra cómo el *makespan* es más sensible que el tiempo total de finalización ante restricciones estrictas sobre los tiempos de espera, pues los problemas donde se intenta minimizar *Cmax* tienen un ARPD mayor que los que tienen como objetivo $\sum C_j$, especialmente cuando las esperas permitidas son breves o nulas.

Para seguir comparando ambas FO, también se estudiarán las funciones de distribución acumuladas de probabilidad de los problemas con las restricciones extremas (esperas indefinidas permitidas y esperas no admitidas) para cada uno de los objetivos.

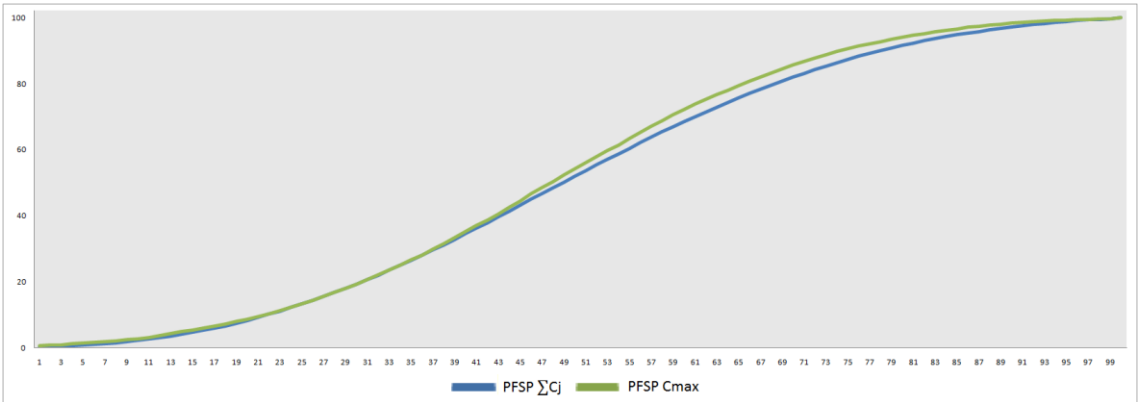


**Figura 6.18** – Funciones de distribución de probabilidad de las secuencias para los problemas con esperas permitidas ilimitadas, para *Cmax* y $\sum C_j$ (Elaboración propia)

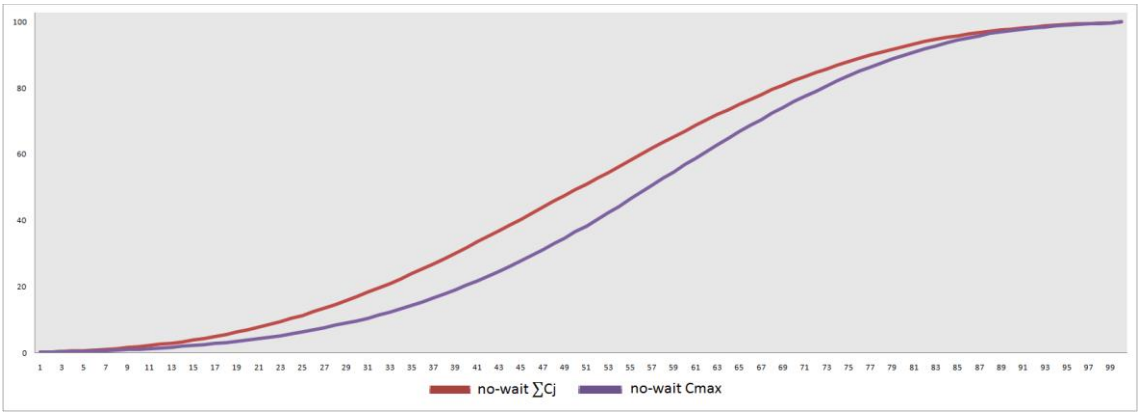


**Figura 6.19** – Funciones de distribución de probabilidad de las secuencias para los problemas con esperas no permitidas, para *Cmax* y $\sum C_j$ (Elaboración propia)

En la figura 6.18 se puede apreciar que las distribuciones de las secuencias de los dos problemas representados son bastante similares, algo que cuadra con la figura 6.17, donde se veía que cuando no se limitaba el tiempo máximo de espera, el ARPD era muy parecido también para los dos objetivos. Por otra parte, la figura 6.17 indicaba que cuando se tenía una restricción *no-wait*, el problema con el *makespan* como objetivo tenía un ARPD considerablemente mayor, por lo que las distintas secuencias dan soluciones más alejadas del óptimo. Y eso mismo se observa en la figura 6.19, donde la probabilidad de obtener valores próximos al óptimo siempre es mayor para el problema con $\sum C_j$ como objetivo.

### 6.2.4 Análisis de los resultados mediante distribuciones estadísticas

Una vez se ha analizado cómo afectan de forma general las restricciones, el objetivo, y el tamaño de los problemas a la dificultad de los mismos, se procede a estudiar cuán simple es hallar secuencias que aporten resultados muy próximos al óptimo en cada problema.

Para ello, y en aras de lograr una simple y clara interpretación de los resultados obtenidos, se trabaja solamente con los problemas con las restricciones extremas (PFSP y *no-wait*), pues se ha demostrado que el resto de ellos presentan comportamientos intermedios, y para aquéllos en los que se busca minimizar el *makespan*. Para empezar, se deben construir las funciones de distribución de probabilidad para cada uno de los problemas a estudiar. La figura 6.20 muestra dicha función para los problemas con 5 trabajos. Como ya se explicó en el capítulo 3, los valores del eje x no indican el incremento (ni absoluto ni relativo) de la FO, sino que indican el valor del objetivo que se obtiene con cada secuencia respecto a los valores mínimo y máximo del mismo, que son los extremos del intervalo dividido en 100 tramos que representa dicho eje.

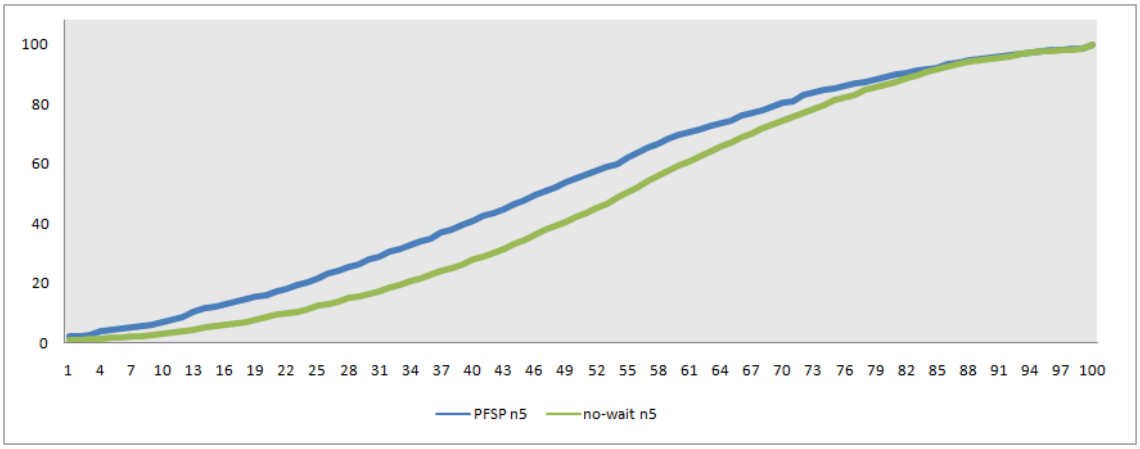


**Figura 6.20** – Funciones de distribución de probabilidad de las secuencias para los problemas PFSP y *no-wait* con 5 trabajos (Elaboración propia)

En el anexo 9.2 se muestran también las distribuciones de probabilidad para los problemas con el resto de número de trabajos considerados (6, 7, 8).

Con el fin de estudiar los resultados próximos al óptimo, el análisis debe centrarse en la parte izquierda de la gráfica, que es donde se encuentran las secuencias con las que se obtienen dichos valores. La figura 6.21 muestra detalladamente dicha sección de la gráfica para los problemas con 5 trabajos.

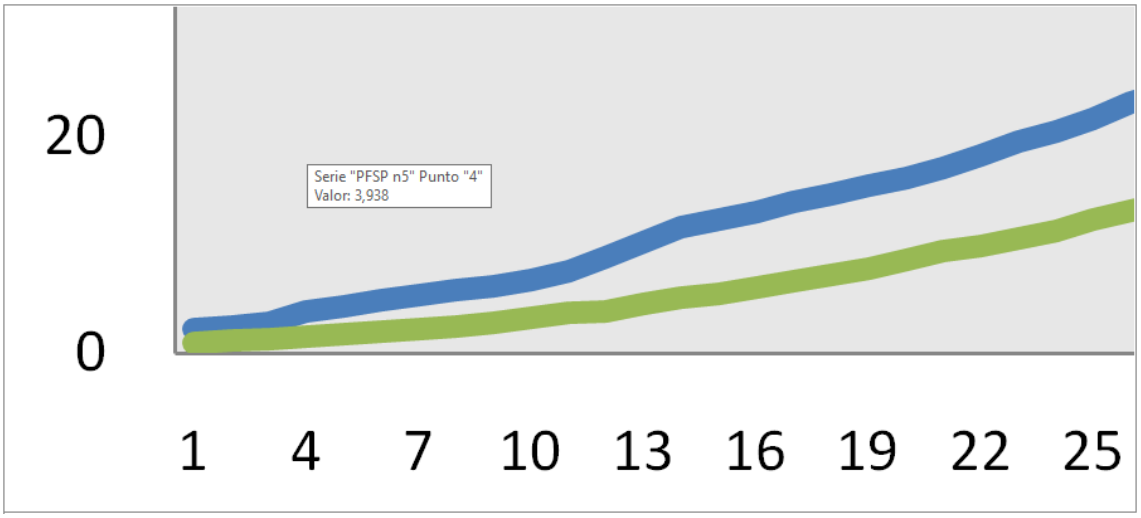


**Figura 6.21** – Funciones de distribución de probabilidad de las secuencias próximas al óptimo para los problemas PFSP y *no-wait* con 5 trabajos (Elaboración propia)

Como se observa en la figura 6.21, existe una probabilidad del 3,938% de que una secuencia generada aleatoriamente se encuentre en el 4% menor de los valores del *Cmax* para el problema sin limitaciones en los tiempos de espera. Dado que cada secuencia generada es independiente de las generadas previamente, y que la probabilidad de obtener buenos resultados es invariable, cada generación se podría describir como un ensayo de Bernoulli, ya que sólo se consideran dos situaciones: éxito (si se obtiene un resultado lo suficiente próximo al óptimo) o fracaso.

Para realizar este análisis, se definirían las siguientes variables aleatorias:

$X_k$: Variable aleatoria que toma el valor 1 si con una secuencia generada aleatoriamente se obtiene un resultado que se encuentre entre los *k* primeros tramos iguales del eje x.

$Y_k$: Variable aleatoria que indica, para *n* secuencias generadas aleatoriamente, con cuántas de ellas se obtienen resultados que se encuentren entre los *k* primeros tramos iguales del eje x.

$X_k$ se distribuye mediante una Bernoulli de parámetro *p* igual a la probabilidad de que una secuencia aleatoria se encuentre en los *k* primeros tramos del eje x.

$$X_k \sim Be\ (p)$$

$Y_k$, por su parte, se caracteriza mediante una Binomial con un parámetro *p* idéntico al de la $X_k$ con igual *k*, y una *n* igual al número de secuencias aleatorias generadas.

$$Y_k \sim B\ (p\ , n)$$

Para el problema representado en las figuras 6.20 y 6.21, cuando se buscan resultados entre los 4 primeros tramos del eje x la probabilidad de que un ensayo sea un éxito es de un 3,938%, y las variables aleatorias para ese caso se representan de la siguiente forma:

$$X_4 \sim Be\ (0{,}03938)$$

$$Y_4 \sim B\ (0{,}03938\ , n)$$

Una vez conocida la probabilidad de éxito de cada ensayo, lo que se busca es el número de ensayos que hay que realizar para que se produzca, al menos, un éxito. Es decir, cuántas secuencias se deben generar para que al menos una de ellas se encuentre lo suficientemente próxima a la óptima como para considerar que el resultado es satisfactorio. No obstante, nunca se puede asegurar al 100% que se obtenga al menos un éxito, aunque se generase una enorme cantidad de secuencias. Por ello, se calcula el número de secuencias generadas con el que se pueda afirmar con un 99% de confianza que con al menos una de ellas se obtiene un resultado próximo al óptimo.

Debido al hecho de que el número de ensayos a realizar es a priori desconocido (pues es lo que se quiere calcular), no es posible calcular directamente la probabilidad de que haya al menos una secuencia entre las *n* generadas que aporte resultados satisfactorios.

$$P\ (Y_k \geq 1) = P\ (Y_k = 1) + P\ (Y_k = 2) + \ \ldots + P\ (Y_k = n)$$

Sin embargo, basándose en la complementariedad, se puede calcular la probabilidad de obtener al menos un éxito a partir de la probabilidad de que las *n* secuencias generadas sean fracasos.

$$P\ (Y_k \geq 1) = 1 - P\ (Y_k < 1) = 1 - P\ (Y_k = 0)$$

Esta probabilidad, por definición, se ha definido del 99%, que es el nivel de confianza con el que se asegura que generar las *n* secuencias aportará al menos una con un buen resultado. Por tanto, sólo quedaría calcular la probabilidad de no obtener ningún éxito al generar *n* secuencias, y resolver la siguiente ecuación:

$$P\ (Y_k \geq 1) = 0{,}99 = 1 - (1 - p)^n$$

$$n = \log_{1-p} 0{,}01$$

Volviendo al ejemplo concreto representado en las figuras 6.20 y 6.21, en el que la probabilidad de que un ensayo sea exitoso es del 3,938%, las secuencias que se deben generar para que exista un 99% de probabilidad de que con al menos una de ellas se consiga un *makespan* en los primeros 4 de los 100 tramos de valores es:

$$\mathrm{n} = \log_{0{,}96062} 0{,}01 = 114{,}62$$

El resultado ha de redondearse siempre al alza, pues en caso contrario la seguridad con la que se realizaría la afirmación sería inferior al 99%, por lo que en este caso habrían de generarse 115 secuencias.

Este mismo procedimiento se repite para el resto de problemas contemplados, y se halla la proporción de secuencias que se encuentran en cada uno de los 10 primeros tramos del eje x de la función de distribución de las secuencias, o lo que es igual, la probabilidad de que con una secuencia aleatoria se obtenga un resultado que se encuentre en los primeros *k* tramos de valores del *makespan*.

| | *n* = 5 | | *n* = 6 | | *n* = 7 | | *n* = 8 | |
|---|---|---|---|---|---|---|---|---|
| ***k*** | PFSP | no wait | PFSP | no wait | PFSP | no wait | PFSP | no wait |
| *1* | 2,313 | 1,000 | 0,646 | 0,177 | 0,069 | 0,030 | 0,076 | 0,004 |
| *2* | 2,521 | 1,188 | 0,792 | 0,222 | 0,103 | 0,042 | 0,099 | 0,007 |
| *3* | 2,875 | 1,375 | 0,927 | 0,271 | 0,153 | 0,060 | 0,123 | 0,010 |
| *4* | 3,938 | 1,583 | 1,174 | 0,330 | 0,220 | 0,085 | 0,161 | 0,015 |
| *5* | 4,375 | 1,854 | 1,382 | 0,399 | 0,299 | 0,116 | 0,217 | 0,020 |
| *6* | 5,000 | 2,000 | 1,639 | 0,472 | 0,431 | 0,161 | 0,261 | 0,029 |
| *7* | 5,500 | 2,333 | 1,885 | 0,549 | 0,541 | 0,215 | 0,321 | 0,043 |
| *8* | 5,896 | 2,500 | 2,219 | 0,681 | 0,686 | 0,269 | 0,418 | 0,058 |
| *9* | 6,313 | 2,917 | 2,691 | 0,809 | 0,879 | 0,341 | 0,539 | 0,080 |
| *10* | 6,896 | 3,333 | 2,951 | 0,948 | 1,133 | 0,436 | 0,668 | 0,110 |

**Tabla 6.8** – Probabilidad, para cada problema, de que una secuencia generada aleatoriamente aporte una solución en los primeros *k* tramos de valores de *Cmax* (Elaboración propia)

Como ya se ha visto en puntos anteriores, y como se vuelve a comprobar en la tabla 6.8, a medida que aumenta la dificultad del problema (mayor número de trabajos, restricciones más estrictas) es considerablemente menos probable encontrar secuencias con las que se obtengan buenos resultados. Por otra parte, buscar resultados más próximos al óptimo reduce la probabilidad de que las secuencias generadas sean consideradas satisfactorias.

Aunque la tabla 6.8 sirve para confirmar las conclusiones que se pudieron obtener a partir del estudio del ARPD de cada problema, pues se sigue apreciando qué aspectos incrementan su dificultad, el interés de este análisis reside en usar las probabilidades que aparecen en dicha tabla para construir la tabla 6.9.

| | *n* = 5<br>120 secuencias | | *n* = 6<br>720 secuencias | | *n* = 7<br>5.040 secuencias | | *n* = 8<br>40.320 secuencias | |
|---|---|---|---|---|---|---|---|---|
| ***k*** | PFSP | no wait | PFSP | no wait | PFSP | no wait | PFSP | no wait |
| *1* | 197 | 459 | 711 | 2.599 | 6.677 | 15.218 | 6.041 | 109.222 |
| *2* | 181 | 386 | 580 | 2.071 | 4.462 | 10.921 | 4.669 | 66.910 |
| *3* | 158 | 333 | 495 | 1.699 | 3.012 | 7.735 | 3.746 | 46.710 |
| *4* | 115 | 289 | 391 | 1.394 | 2.094 | 5.396 | 2.859 | 31.469 |
| *5* | 103 | 247 | 331 | 1.151 | 1.538 | 3.983 | 2.123 | 22.642 |
| *6* | 90 | 228 | 279 | 973 | 1.067 | 2.855 | 1.765 | 15.767 |
| *7* | 82 | 196 | 242 | 838 | 850 | 2.142 | 1.434 | 10.778 |
| *8* | 76 | 182 | 206 | 675 | 669 | 1.708 | 1.101 | 7.933 |
| *9* | 71 | 156 | 169 | 567 | 522 | 1.348 | 852 | 5.729 |
| *10* | 65 | 136 | 154 | 484 | 404 | 1.056 | 688 | 4.185 |

**Tabla 6.9** – Probabilidad, para cada problema, de que una secuencia generada aleatoriamente aporte una solución en los primeros *k* tramos de valores de *Cmax* (Elaboración propia)

En la tabla 6.9 se descubre que un problema *no-wait* es mucho más difícil que un PFSP de lo que parece en la figura 6.20, por ejemplo, pues observando dicha gráfica se podría pensar que las distribuciones de las secuencias de ambos problemas son muy parecidas y la diferencia no sería considerable a efectos prácticos. Sin embargo, observando esta tabla, se aprecia que se deben generar muchísimas más secuencias cuando se tiene un problema *no-wait* si se quieren obtener resultados similares que en un PFSP, creciendo dicha diferencia enormemente cuando se procesa una elevada cantidad de trabajos. Se aprecia, también, que si se necesitan obtener resultados muy próximos al óptimo (especialmente en problemas *no-wait*), la probabilidad de hallar las secuencias que los proporcionan es tan remota que puede darse el caso de que se acaben teniendo que generar más secuencias aleatorias que las totales del problema. En esos casos, realizar la enumeración completa requeriría generar una menor cantidad de secuencias, además de contar con la seguridad de que así se hallaría el óptimo.

## 6.3 Comportamiento de secuencias óptimas de problemas con esperas limitadas en un PFSP

Resulta de interés estudiar si la secuencia óptima de un problema puede usarse en otros y seguir obteniendo buenos resultados, para así no tener que realizar en múltiples ocasiones la enumeración completa en caso de tener que resolver problemas con idéntico entorno pero con distintas limitaciones del tiempo de espera.

### 6.3.1 FO: *Cmax*

Al aplicar por primera vez la enumeración completa, se guardan no sólo los resultados óptimos de los objetivos, sino también las secuencias que proporcionan dichos valores en cada uno de los problemas estudiados.

Lo que se analiza en la tabla 6.10 es el aumento porcentual del valor del *makespan* en un PFSP, para cada uno de los tamaños de problema contemplados, cuando se usa la secuencia óptima de cada uno de los diferentes problemas con esperas limitadas, en comparación al *Cmax* obtenido con la secuencia óptima del problema de permutación. Se aprecia que, a medida que disminuye el tiempo máximo de espera permitido es menos aconsejable usar el óptimo del problema con esperas limitadas en el PFSP, pues evidentemente los problemas estarán más diferenciados. Para problemas con esperas lo suficientemente altas, su secuencia óptima podría incluso coincidir con la del PFSP, dependiendo del número de trabajos y máquinas, y de los tiempos de proceso de cada trabajo en cada una de las máquinas.

| *FO: Cmax* | | max40 | max30 | max20 | max10 | no wait | ARPD PFSP |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 0,4 | 0,8 | 2,0 | 2,0 | 2,7 | 17,4 |
| | m = 10 | 0,8 | 1,3 | 1,9 | 2,6 | 4,6 | 11,8 |
| | m = 15 | 0,3 | 1,4 | 2,0 | 2,5 | 3,5 | 10,4 |
| | m = 20 | 0,7 | 1,6 | 2,0 | 2,1 | 3,0 | 8,2 |
| n = 6 | m = 5 | 1,1 | 1,4 | 2,8 | 4,1 | 4,1 | 19,2 |
| | m = 10 | 0,7 | 0,9 | 1,5 | 2,3 | 3,1 | 13,9 |
| | m = 15 | 0,5 | 0,7 | 2,2 | 2,8 | 5,0 | 14,6 |
| | m = 20 | 1,1 | 1,7 | 3,0 | 4,2 | 5,7 | 12,4 |
| n = 7 | m = 5 | 1,0 | 1,3 | 1,5 | 2,8 | 3,5 | 23,1 |
| | m = 10 | 1,1 | 1,9 | 2,6 | 3,4 | 5,1 | 16,6 |
| | m = 15 | 0,9 | 2,9 | 3,3 | 3,7 | 5,5 | 13,8 |
| | m = 20 | 0,9 | 1,4 | 1,9 | 3,4 | 5,9 | 12,5 |
| n = 8 | m = 5 | 1,6 | 2,5 | 3,9 | 5,5 | 7,6 | 22,9 |
| | m = 10 | 1,0 | 1,3 | 2,0 | 4,2 | 5,8 | 19,7 |
| | m = 15 | 1,6 | 2,4 | 3,0 | 5,7 | 5,8 | 16,3 |
| | m = 20 | 1,8 | 2,7 | 4,1 | 5,4 | 6,6 | 13,9 |

**Tabla 6.10** – Incremento relativo de *Cmax* al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, y valores del ARPD para el PFSP (Elaboración propia)

Se ha añadido en la tabla 6.10 los valores del ARPD para el problema de permutación que aparecían en la tabla 6.6, y se comparan dichos valores con el incremento del *Cmax* medio para cada uno de los problemas. Que el valor del ARPD sea claramente mayor en cualquier caso quiere decir que, en caso de tener que usar una secuencia sin conocer el óptimo del PFSP, es mucho más conveniente usar, si se conoce, la óptima de otro de los problemas (incluso del *no-wait*, que es el más diferente) antes que una aleatoria.

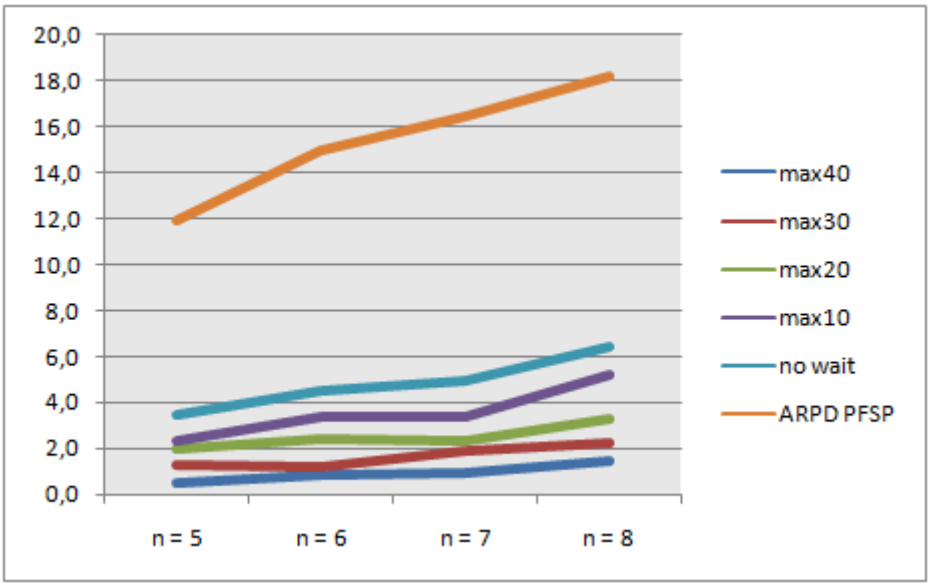


**Figura 6.22** – Incremento relativo de *Cmax* al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, y valores del ARPD para el PFSP, según el número de trabajos (Elaboración propia)

La figura 6.22 muestra que usar la óptima de un problema con esperas limitadas en el PFSP da soluciones más alejadas del óptimo a medida que el número de trabajos a procesar crece, así como también es más difícil acercarse al óptimo usando secuencias aleatorias (aumenta el ARPD, como indicaba la tabla 6.6).

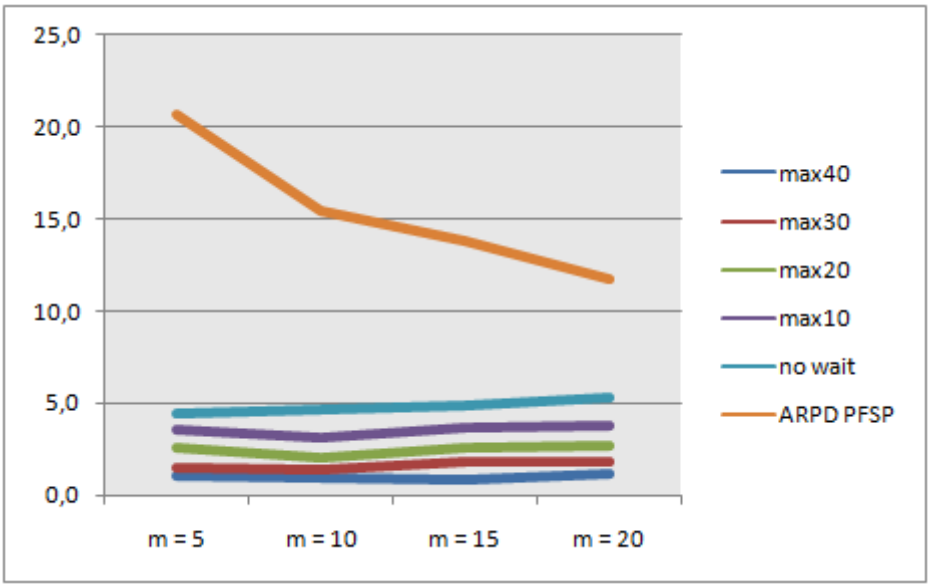


**Figura 6.23** – Incremento relativo de *Cmax* al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, y valores del ARPD para el PFSP, según el número de máquinas (Elaboración propia)

Sin embargo, la figura 6.23 indica que el número de máquinas parece no afectar al resultado de usar las secuencias óptimas de otros problemas en el PFSP, aunque sí que hace disminuir considerable el ARPD del mismo. Esto significa que para un PFSP con un número de máquinas lo suficientemente alto y con pocos trabajos a procesar, podría haber un momento en el que fuera más favorable incluso usar una secuencia generada aleatoriamente que la óptima de uno de los problemas con esperas limitadas.

### 6.3.2 FO: $\sum C_j$

De igual forma, el análisis del punto previo se realiza también para los problemas que tienen como objetivo minimizar el tiempo total de finalización de los trabajos. Se construye una tabla como la 6.10, y se aprecia que para este objetivo usar las secuencias óptimas de los problemas con limitación de esperas en el PFSP da soluciones muy próximas a la que se obtiene con la óptima del propio PFSP. De hecho, para entornos con sólo 5 trabajos, las secuencias óptimas de los problemas donde las esperas se limitan en 40 unidades temporales dan los mismos resultados que los problemas que sólo tienen la restricción de permutación, siempre y cuando el número de máquinas no sea lo suficientemente alto.

| FO: ∑Cj | | max40 | max30 | max20 | max10 | no wait | ARPD PFSP |
|---|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 0,0 | 0,1 | 0,1 | 0,2 | 0,3 | 19,8 |
| | m = 10 | 0,0 | 0,1 | 0,1 | 0,1 | 0,3 | 13,9 |
| | m = 15 | 0,0 | 0,0 | 0,3 | 0,4 | 0,4 | 12,1 |
| | m = 20 | 0,2 | 0,2 | 0,5 | 0,7 | 0,9 | 10,6 |
| n = 6 | m = 5 | 0,0 | 0,1 | 0,1 | 1,1 | 1,7 | 20,0 |
| | m = 10 | 0,2 | 0,4 | 0,6 | 0,8 | 1,3 | 13,0 |
| | m = 15 | 0,2 | 0,2 | 0,9 | 1,2 | 1,4 | 12,6 |
| | m = 20 | 0,1 | 0,4 | 0,4 | 0,7 | 1,3 | 8,6 |
| n = 7 | m = 5 | 0,1 | 0,2 | 0,6 | 1,1 | 2,1 | 21,4 |
| | m = 10 | 0,2 | 0,3 | 1,0 | 1,6 | 2,1 | 15,1 |
| | m = 15 | 0,3 | 0,5 | 1,1 | 1,1 | 1,9 | 13,7 |
| | m = 20 | 0,3 | 0,5 | 0,6 | 0,7 | 1,5 | 12,6 |
| n = 8 | m = 5 | 0,4 | 0,6 | 1,0 | 1,4 | 2,2 | 21,0 |
| | m = 10 | 0,3 | 0,4 | 1,2 | 1,6 | 2,0 | 17,7 |
| | m = 15 | 0,3 | 0,4 | 0,8 | 1,8 | 2,2 | 13,2 |
| | m = 20 | 0,4 | 0,7 | 1,1 | 1,1 | 2,0 | 13,2 |

**Tabla 6.11** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, y valores del ARPD para el PFSP (Elaboración propia)

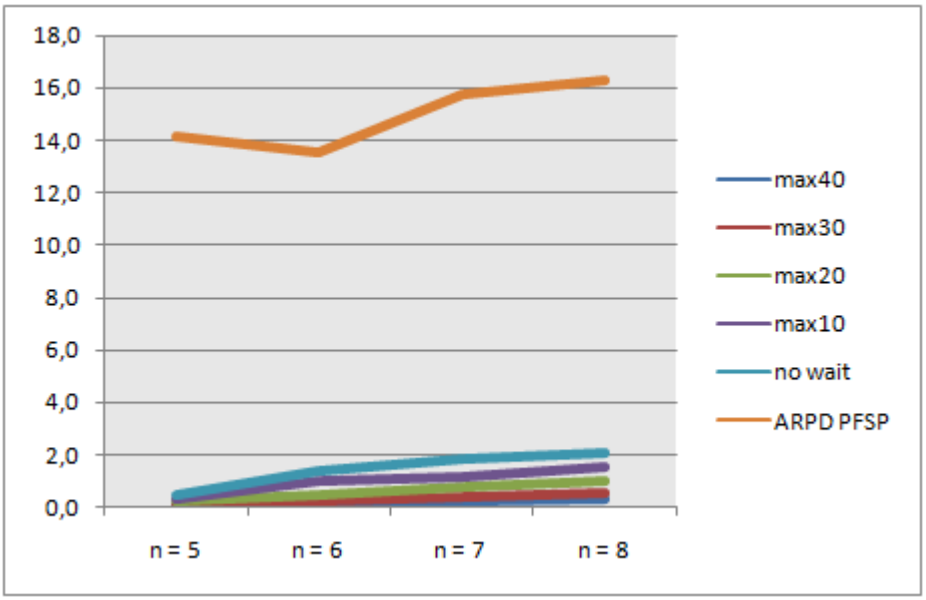


**Figura 6.24** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, y valores del ARPD para el PFSP, según el número de trabajos (Elaboración propia)

Al igual que ocurría con el otro objetivo, usar la solución óptima de un problema en otro tendrá mejores resultados en problemas con restricciones similares, como se puede apreciar en las figuras 6.24 y 6.25. Asimismo, un gran número de trabajos a procesar hace que la secuencia óptima de otro problema usada en el PFSP no esté tan cerca del óptimo. El número de máquinas, por otra parte, vuelve a no tener ninguna consecuencia en ese aspecto.

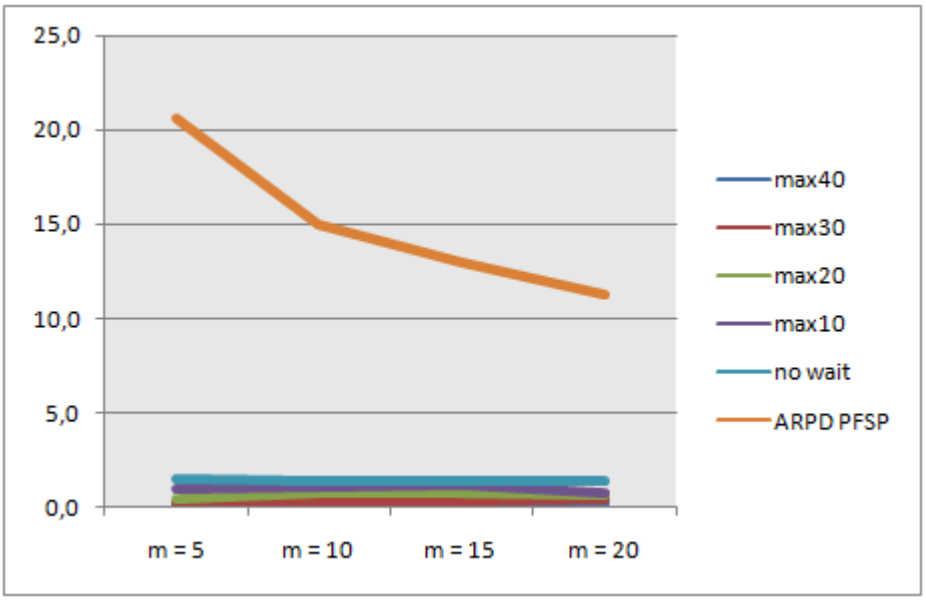


**Figura 6.25** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, y valores del ARPD para el PFSP, según el número de máquinas (Elaboración propia)

### 6.3.3 Comparación entre problemas con ambos objetivos

Promediando los valores para cada uno de los problemas contemplados, según el número de trabajos y máquinas, que aparecen en las tablas 6.10 y 6.11, se construye la siguiente tabla, donde se resume lo expuesto en los últimos puntos, donde se analizaba cada uno de los objetivos independientemente.

| *FO: Cmax* | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|
| n = 5 | 0,5 | 1,3 | 2,0 | 2,3 | 3,5 |
| n = 6 | 0,9 | 1,2 | 2,4 | 3,4 | 4,5 |
| n = 7 | 1,0 | 1,9 | 2,3 | 3,4 | 5,0 |
| n = 8 | 1,5 | 2,2 | 3,2 | 5,2 | 6,5 |
| m = 5 | 1,0 | 1,5 | 2,6 | 3,6 | 4,5 |
| m = 10 | 0,9 | 1,4 | 2,0 | 3,1 | 4,7 |
| m = 15 | 0,8 | 1,8 | 2,6 | 3,7 | 4,9 |
| m = 20 | 1,1 | 1,8 | 2,7 | 3,8 | 5,3 |

| *FO: ∑Cj* | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|
| n = 5 | 0,1 | 0,1 | 0,2 | 0,3 | 0,5 |
| n = 6 | 0,1 | 0,3 | 0,5 | 1,0 | 1,4 |
| n = 7 | 0,2 | 0,4 | 0,8 | 1,1 | 1,9 |
| n = 8 | 0,4 | 0,5 | 1,0 | 1,5 | 2,1 |
| m = 5 | 0,1 | 0,2 | 0,4 | 0,9 | 1,5 |
| m = 10 | 0,2 | 0,3 | 0,7 | 1,0 | 1,4 |
| m = 15 | 0,2 | 0,3 | 0,8 | 1,1 | 1,5 |
| m = 20 | 0,3 | 0,4 | 0,7 | 0,8 | 1,4 |

**Tabla 6.12** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP (Elaboración propia)

Para comparar ambos objetivos en este contexto, se analiza el caso donde se usa la secuencia óptima del problema *no-wait* en el PFSP.

| | *C*max | ∑*C*j |
|---|---|---|
| *n = 5* | 3,5 | 0,5 |
| *n = 6* | 4,5 | 1,4 |
| *n = 7* | 5,0 | 1,9 |
| *n = 8* | 6,5 | 2,1 |
| *m = 5* | 4,5 | 1,5 |
| *m = 10* | 4,7 | 1,4 |
| *m = 15* | 4,9 | 1,5 |
| *m = 20* | 5,3 | 1,4 |

**Tabla 6.13** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del problema sin esperas permitidas en el PFSP (Elaboración propia)

Se confirma que, para cualquiera de los tamaños de problema, cuando el objetivo es minimizar $\sum C_j$ se obtienen resultados mucho más próximos al óptimo de PFSP al usar la óptima de otros problemas que cuando se intenta minimizar *Cmax*.

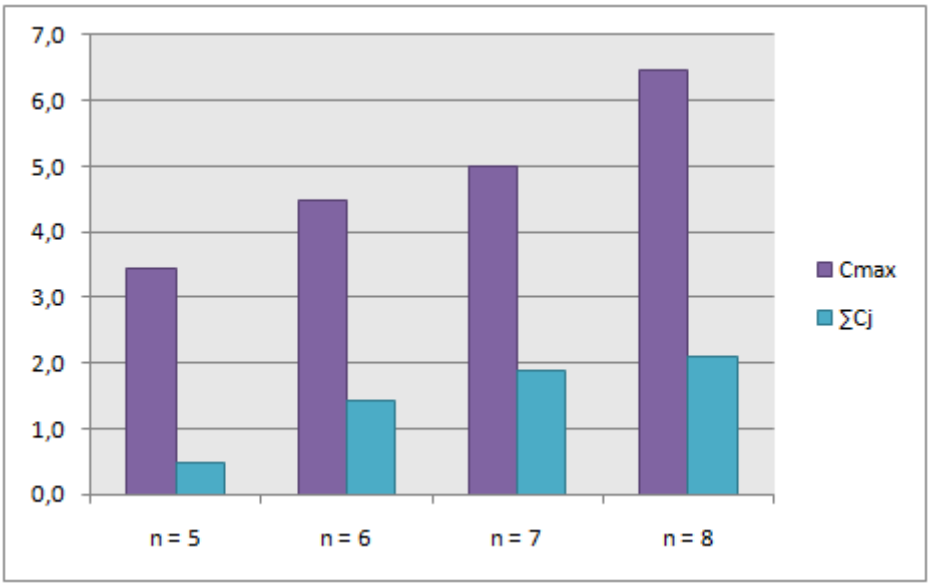


**Figura 6.26** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del problema sin esperas permitidas en el PFSP, según el número de trabajos (Elaboración propia)

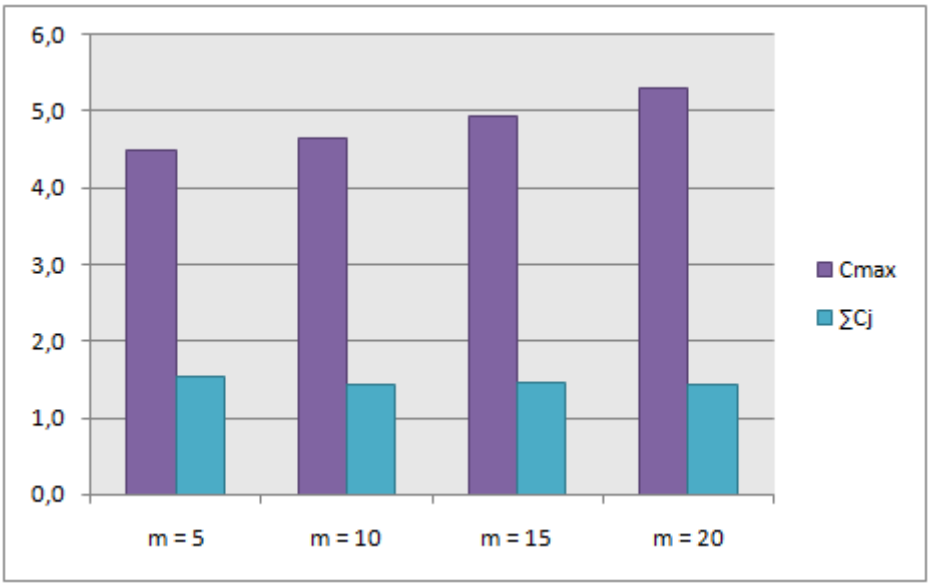


**Figura 6.27** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del problema sin esperas permitidas en el PFSP, según el número de máquinas (Elaboración propia)

## 6.4 Comportamiento de la secuencia óptima de un PFSP en problemas con esperas limitadas

Estudiar los resultados que se obtienen en problemas con esperas limitadas usando la secuencia óptima del PFSP tiene incluso mayor utilidad que el caso contrario, pues como se vio en puntos anteriores, éstos son problemas más difíciles, con sus secuencias más alejadas de la óptima que en el problema el PFSP.

Así que si hallando la secuencia óptima del PFSP, que es el problema más sencillo de los estudiados, se pudiera usar en los otros sin obtener resultados muy alejados del óptimo, podría ser de gran utilidad en casos donde sea extremadamente difícil hallar la óptima de un problema con limitaciones en los tiempos de espera de los trabajos entre máquinas.

### 6.4.1 FO: *Cmax*

De forma similar a como se procedió en el punto anterior, se usa la secuencia óptima del PFSP en los otros problemas, y se calcula en cuánto aumenta el valor del *makespan* respecto al que éste adquiere en el problema con esperas indefinidas permitidas.

| FO: Cmax | | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 1,3 | 1,7 | 3,5 | 5,5 | 7,8 |
| | m = 10 | 1,2 | 2,7 | 5,3 | 8,7 | 12,5 |
| | m = 15 | 1,4 | 2,7 | 4,8 | 6,7 | 10,0 |
| | m = 20 | 1,8 | 3,0 | 5,3 | 9,2 | 13,9 |
| n = 6 | m = 5 | 2,4 | 4,8 | 8,2 | 12,5 | 17,4 |
| | m = 10 | 2,2 | 4,2 | 6,9 | 9,7 | 13,2 |
| | m = 15 | 1,8 | 2,6 | 4,0 | 6,0 | 9,4 |
| | m = 20 | 1,9 | 2,8 | 4,8 | 7,2 | 10,5 |
| n = 7 | m = 5 | 2,8 | 5,0 | 7,6 | 10,0 | 12,2 |
| | m = 10 | 2,6 | 5,4 | 9,0 | 12,3 | 16,4 |
| | m = 15 | 4,2 | 6,1 | 8,4 | 11,2 | 15,0 |
| | m = 20 | 3,0 | 4,5 | 6,4 | 8,9 | 12,8 |
| n = 8 | m = 5 | 4,3 | 7,4 | 10,5 | 13,5 | 16,4 |
| | m = 10 | 4,8 | 7,4 | 9,9 | 12,8 | 15,8 |
| | m = 15 | 4,6 | 6,4 | 9,0 | 12,6 | 16,6 |
| | m = 20 | 4,1 | 6,8 | 10,3 | 14,2 | 18,4 |

**Tabla 6.14** – Incremento relativo de *Cmax* al usar la secuencia óptima del PFSP en problemas con esperas limitadas (Elaboración propia)

La diferencia más evidente frente a lo estudiado en el punto anterior es que, para entornos complejos, usar la óptima de un PFSP en un problema *no-wait* puede hacer que se obtengan valores de este objetivo hasta un 20% peores, mientras que en el caso inverso dicho valor apenas llegaba al 7%. Esto tiene su explicación en el hecho de que los problemas son considerablemente

más sencillos si sus esperas permitidas son mayores, como se comprobó en el capítulo 6.2. Por ello, como la secuencia óptima del PFSP se halla sin preocuparse de los tiempos de espera, usarla en problemas que las limiten puede tener grandes consecuencias sobre el resultado, algo que no ocurre para el caso contrario.

Por otro lado, en la figura 6.28 se observa que un mayor número de trabajos empeora el resultado obtenido en un problema con esperas limitadas al usar la secuencia óptima del PFSP, algo que en el capítulo 6.3 se comprobó que también ocurría en el caso contrario.

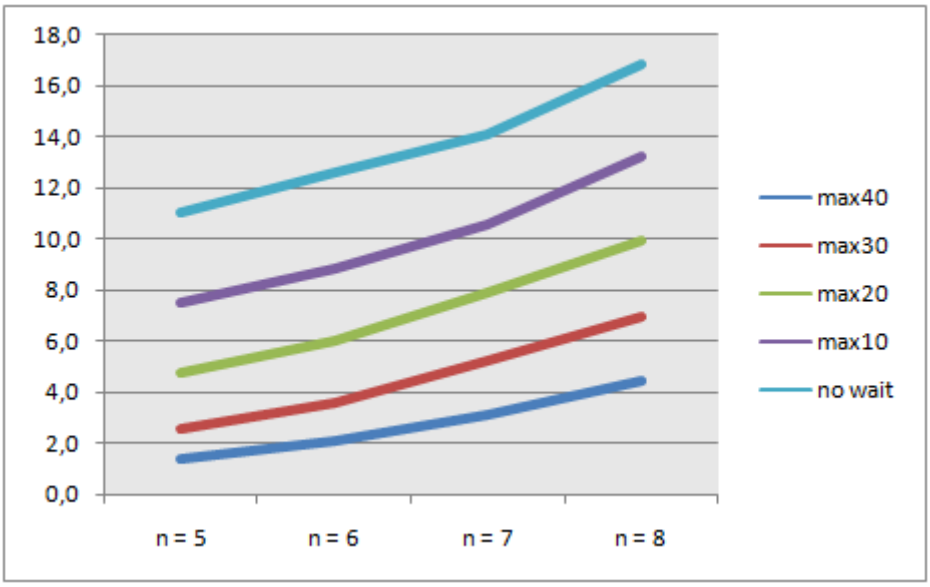


**Figura 6.28** – Incremento relativo de *Cmax* al usar la secuencia óptima del PFSP en problemas con esperas limitadas, según el número de trabajos (Elaboración propia)

El número de máquinas, por su parte, vuelve a ser indiferente para lo que se está estudiando en este punto, como se aprecia en la figura 6.29

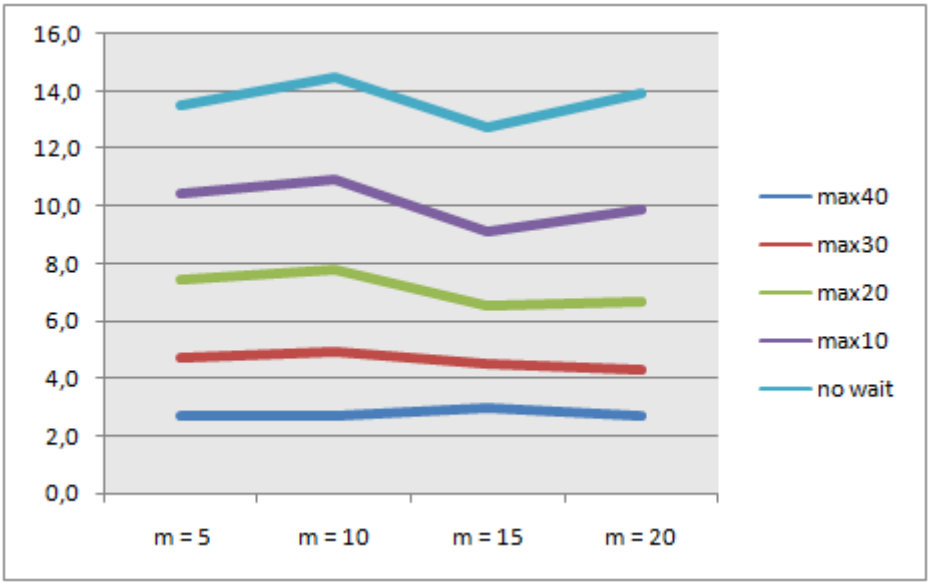


**Figura 6.29** – Incremento relativo de *Cmax* al usar la secuencia óptima del PFSP en problemas con esperas limitadas, según el número de máquinas (Elaboración propia)

De nuevo, nos centramos en los problemas con las restricciones extremas. En este caso, se usa la secuencia óptima del PFSP en el problema *no-wait* sin esperas permitidas, y se calcula su RPD (el incremento relativo en comparación al resultado obtenido con la óptima del *no-wait*). Dichos valores ya aparecían en la tabla 6.14, pero en la 6.15 se comparan con el ARPD del problema *no-wait*. De esta forma, el aumento del *makespan* se contextualiza para cada tamaño de problema, pues se compara cuán malo es el resultado obtenido al seguir la secuencia óptima del PFSP con el que se obtendría, de media, usando el resto de secuencias.

| FO: C max | | RPD ópt PFSP en nwt | ARPD nwt |
|---|---|---|---|
| n = 5 | m = 5 | 7,8 | 22,8 |
| | m = 10 | 12,5 | 18,2 |
| | m = 15 | 10,0 | 17,1 |
| | m = 20 | 13,9 | 16,6 |
| n = 6 | m = 5 | 17,4 | 28,4 |
| | m = 10 | 13,2 | 21,6 |
| | m = 15 | 9,4 | 20,3 |
| | m = 20 | 10,5 | 16,7 |
| n = 7 | m = 5 | 12,2 | 27,4 |
| | m = 10 | 16,4 | 27,8 |
| | m = 15 | 15,0 | 25,4 |
| | m = 20 | 12,8 | 22,3 |
| n = 8 | m = 5 | 16,4 | 32,1 |
| | m = 10 | 15,8 | 30,0 |
| | m = 15 | 16,6 | 25,7 |
| | m = 20 | 18,4 | 24,3 |

**Tabla 6.15** – Incremento relativo de *Cmax* al usar la secuencia óptima del PFSP en el problema *no-wait*, y ARPD del problema sin esperas permitidas, para cada tamaño de problema (Elaboración propia)

Se observa que, a medida que aumenta el número de máquinas que forman el taller, la diferencia entre los dos valores se hace cada vez menor, siendo especialmente pequeña en los problemas con 5 y 8 trabajos. Precisamente se analiza este último caso, pues es el más cercano entre los estudiados a un caso real, en la figura 6.28, y de ella se puede deducir que cuando el número de máquinas alcanza un valor lo suficientemente alto, hay una mayor probabilidad de obtener un mejor resultado en el problema *no-wait* utilizando una secuencia aleatoria que con la óptima del PFSP.

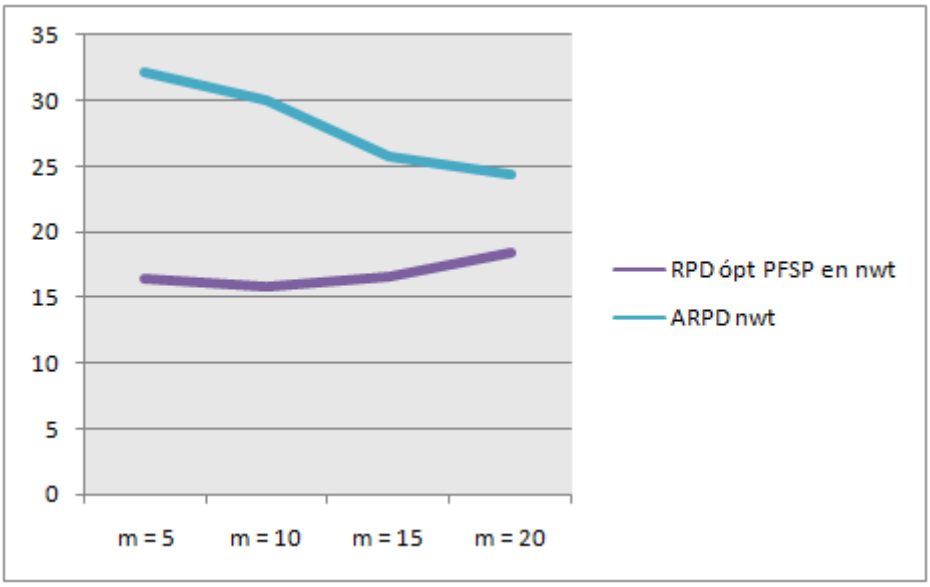


**Figura 6.30** – Incremento relativo de *Cmax* al usar la secuencia óptima del PFSP en el problema *no-wait*, y ARPD del problema sin esperas permitidas, para problemas de 8 trabajos (Elaboración propia)

### 6.4.2 FO: $\sum C_j$

Al igual que cuando el *makespan* era el objetivo a minimizar, se usa la secuencia óptima del PFSP en el resto de problemas, y $\sum C_j$ sufre un incremento que se refleja en la siguiente tabla:

| FO: ∑Cj | | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|---|
| n = 5 | m = 5 | 0,0 | 0,0 | 0,1 | 0,4 | 0,9 |
| | m = 10 | 0,2 | 0,3 | 0,5 | 0,6 | 1,0 |
| | m = 15 | 0,1 | 0,1 | 0,2 | 0,5 | 1,0 |
| | m = 20 | 0,3 | 0,5 | 0,7 | 1,1 | 1,6 |
| n = 6 | m = 5 | 0,0 | 0,1 | 0,2 | 0,7 | 1,7 |
| | m = 10 | 0,2 | 0,6 | 1,2 | 1,7 | 2,8 |
| | m = 15 | 0,5 | 0,7 | 1,0 | 1,6 | 2,6 |
| | m = 20 | 0,2 | 0,4 | 0,7 | 1,2 | 2,1 |
| n = 7 | m = 5 | 0,5 | 0,9 | 1,5 | 2,9 | 3,9 |
| | m = 10 | 0,8 | 1,3 | 1,9 | 2,9 | 4,2 |
| | m = 15 | 0,4 | 0,9 | 1,5 | 2,7 | 4,0 |
| | m = 20 | 0,4 | 0,6 | 0,9 | 1,5 | 2,4 |
| n = 8 | m = 5 | 0,5 | 1,1 | 2,0 | 2,8 | 4,0 |
| | m = 10 | 0,8 | 1,3 | 2,1 | 3,5 | 5,4 |
| | m = 15 | 0,3 | 0,6 | 1,1 | 1,4 | 2,2 |
| | m = 20 | 0,3 | 0,8 | 1,4 | 2,7 | 4,2 |

**Tabla 6.16** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima del PFSP en problemas con esperas limitadas (Elaboración propia)

Como con la otra función objetivo, el incremento es mayor cuando crece el número de trabajos, y es independiente al de máquinas, como se observa en las dos siguientes figuras:

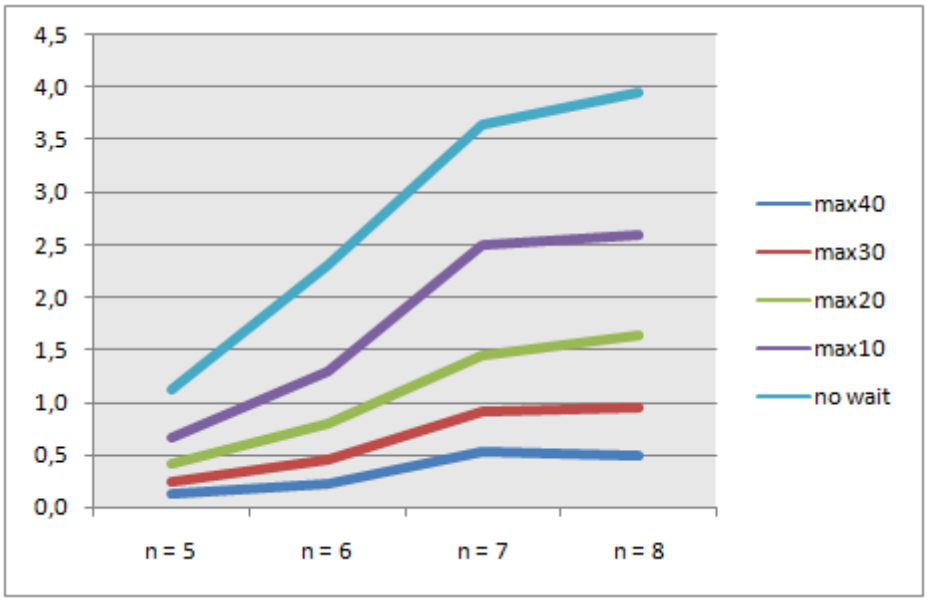


**Figura 6.31** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima del PFSP en problemas con esperas limitadas, según el número de trabajos (Elaboración propia)

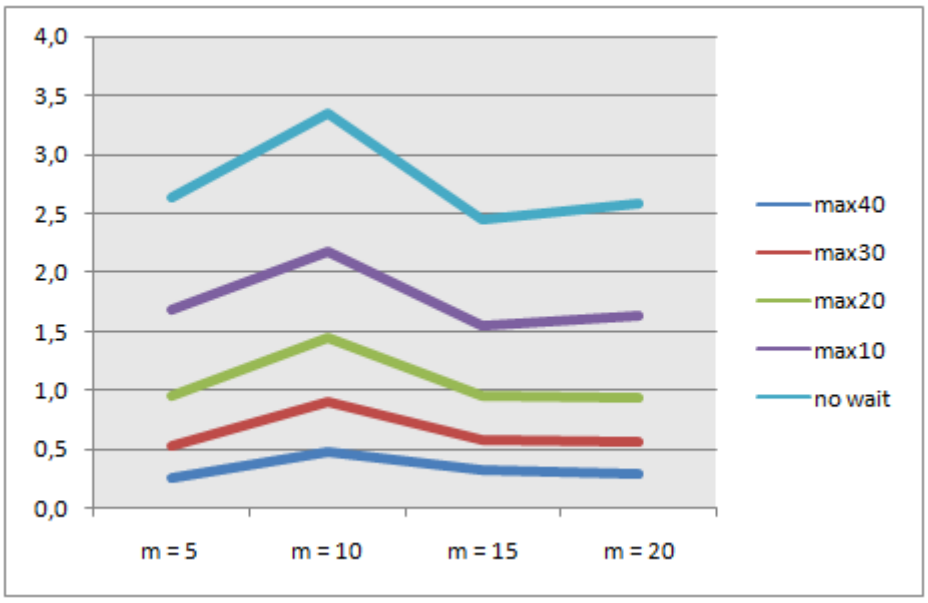


**Figura 6.32** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima del PFSP en problemas con esperas limitadas, según el número de máquinas (Elaboración propia)

Con el fin de contextualizar el incremento de $\sum C_j$, se compara con datos ya conocidos, como es el ARPD del problema *no-wait* para cada uno de los tamaños de problema (ver tabla 6.7).

| FO: ∑Cj | | RPD ópt PFSP en nwt | ARPD nwt |
|---|---|---|---|
| n = 5 | m = 5 | 0,9 | 22,4 |
| | m = 10 | 1,0 | 17,5 |
| | m = 15 | 1,0 | 15,8 |
| | m = 20 | 1,6 | 13,9 |
| n = 6 | m = 5 | 1,7 | 22,8 |
| | m = 10 | 2,8 | 15,9 |
| | m = 15 | 2,6 | 15,6 |
| | m = 20 | 2,1 | 11,2 |
| n = 7 | m = 5 | 3,9 | 24,2 |
| | m = 10 | 4,2 | 20,8 |
| | m = 15 | 4,0 | 19,9 |
| | m = 20 | 2,4 | 18,9 |
| n = 8 | m = 5 | 4,0 | 28,2 |
| | m = 10 | 5,4 | 24,4 |
| | m = 15 | 2,2 | 18,9 |
| | m = 20 | 4,2 | 18,6 |

**Tabla 6.17** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima del PFSP en el problema *no-wait*, y ARPD del problema sin esperas permitidas, para cada tamaño de problema (Elaboración propia)

A diferencia de lo que ocurría con el *makespan*, cuando el objetivo es el tiempo total de finalización sí que parece compensar usar la secuencia óptima del PFSP en cualquiera de los otros problemas, incluso en el problema *no-wait*, pues se obtendrá un resultado mucho más próximo al óptimo que el que se obtendría, de media, con el resto de posibles secuencias.

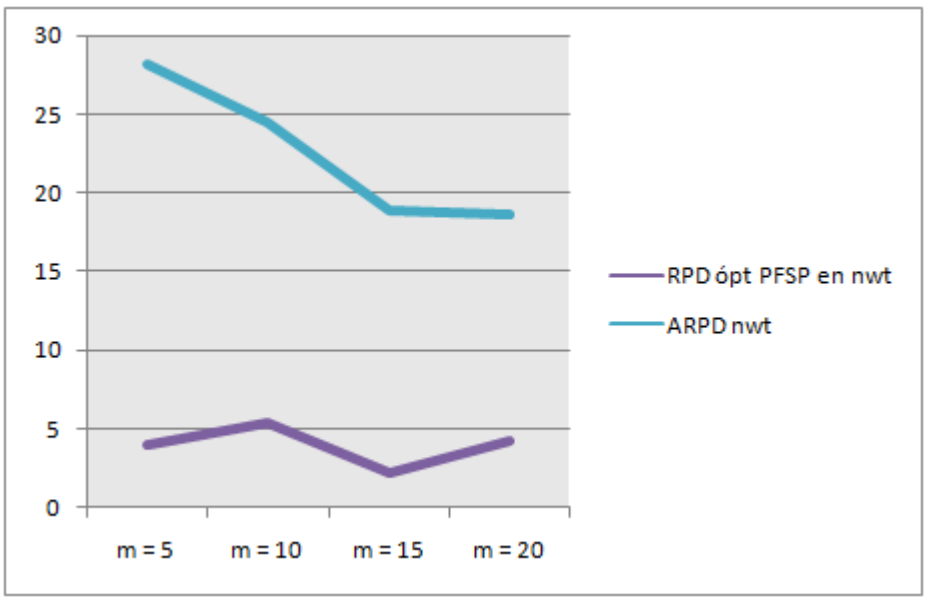


**Figura 6.33** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima del PFSP en el problema *no-wait*, y ARPD del problema sin esperas permitidas, para problemas de 8 trabajos (Elaboración propia)

### 6.4.3 Comparación entre problemas con ambos objetivos

Se calcula el promedio de los valores que forman las tablas 6.14 y 6.16 con el objeto de hallar el incremento relativo medio de los dos objetivos, para cada número de trabajos y máquinas, al usar la secuencia óptima del PFSP en cada uno de los problemas con restricciones en los tiempos de espera.

| FO: Cmax | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|
| n = 5 | 1,4 | 2,6 | 4,7 | 7,5 | 11,0 |
| n = 6 | 2,1 | 3,6 | 6,0 | 8,9 | 12,6 |
| n = 7 | 3,2 | 5,3 | 7,9 | 10,6 | 14,1 |
| n = 8 | 4,5 | 7,0 | 9,9 | 13,3 | 16,8 |
| m = 5 | 2,7 | 4,7 | 7,4 | 10,4 | 13,5 |
| m = 10 | 2,7 | 4,9 | 7,8 | 10,9 | 14,5 |
| m = 15 | 3,0 | 4,5 | 6,6 | 9,1 | 12,8 |
| m = 20 | 2,7 | 4,3 | 6,7 | 9,9 | 13,9 |

| FO: ∑Cj | max40 | max30 | max20 | max10 | no wait |
|---|---|---|---|---|---|
| n = 5 | 0,1 | 0,2 | 0,4 | 0,7 | 1,1 |
| n = 6 | 0,2 | 0,5 | 0,8 | 1,3 | 2,3 |
| n = 7 | 0,5 | 0,9 | 1,5 | 2,5 | 3,6 |
| n = 8 | 0,5 | 1,0 | 1,6 | 2,6 | 3,9 |
| m = 5 | 0,3 | 0,5 | 1,0 | 1,7 | 2,6 |
| m = 10 | 0,5 | 0,9 | 1,4 | 2,2 | 3,3 |
| m = 15 | 0,3 | 0,6 | 1,0 | 1,6 | 2,4 |
| m = 20 | 0,3 | 0,6 | 0,9 | 1,6 | 2,6 |

**Tabla 6.18** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del PFSP en los problemas con esperas limitadas (Elaboración propia)

Nos centraremos en los dos problemas extremos en la tabla 6.19, porque los problemas con esperas limitadas pero permitidas presentan situaciones intermedias, y así se pueden sacar conclusiones de forma más clara.

| | Cmax | ∑Cj |
|---|---|---|
| n = 5 | 11,0 | 1,1 |
| n = 6 | 12,6 | 2,3 |
| n = 7 | 14,1 | 3,6 |
| n = 8 | 16,8 | 3,9 |
| m = 5 | 13,5 | 2,6 |
| m = 10 | 14,5 | 3,3 |
| m = 15 | 12,8 | 2,4 |
| m = 20 | 13,9 | 2,6 |

**Tabla 6.19** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del PFSP en el problema sin esperas permitidas (Elaboración propia)

Como se ha visto en repetidas ocasiones, un gran número de trabajos a procesar hace que no se obtengan resultados tan próximos al óptimo al usar la secuencia óptima de un problema en otro, sean cuales sean éstos. El número de máquinas, por su parte, demuestra no tener incidencia alguna en este aspecto, para ninguno de los dos objetivos considerados.

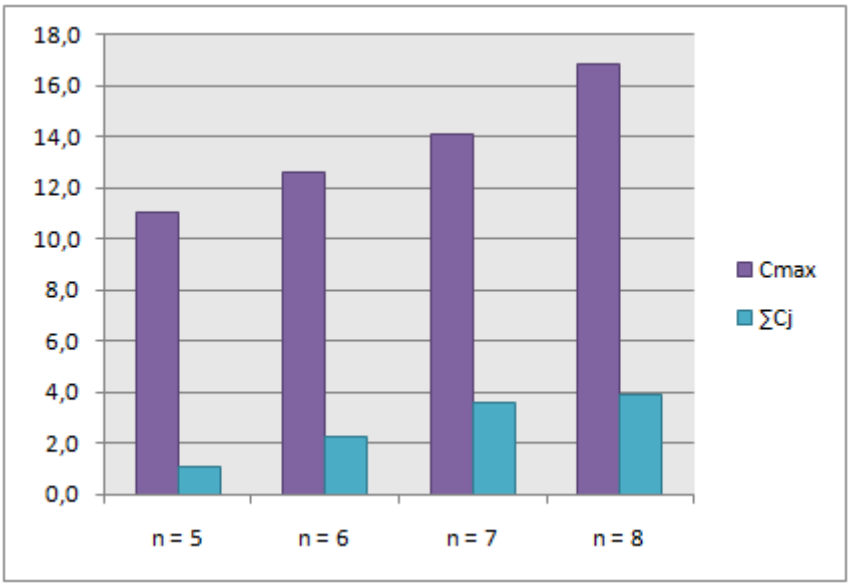


**Figura 6.34** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del PFSP en el problema sin esperas permitidas, según el número de trabajos (Elaboración propia)

Por otra parte, y como se puede observar en las figuras 6.34 y 6.35, existe una clara diferencia entre los problemas con distintos objetivos, pues cuando éste es el tiempo total de finalización, usar la secuencia óptima del PFSP en un problema considerablemente más complicado como es el *no-wait* sigue dando resultados como mucho un 4% peores, para los problemas que se han estudiado. Sin embargo, el *makespan* empeora, de media, un 10% en el caso analizado más sencillo, pudiendo llegar hasta el 17% el incremento relativo de su valor.

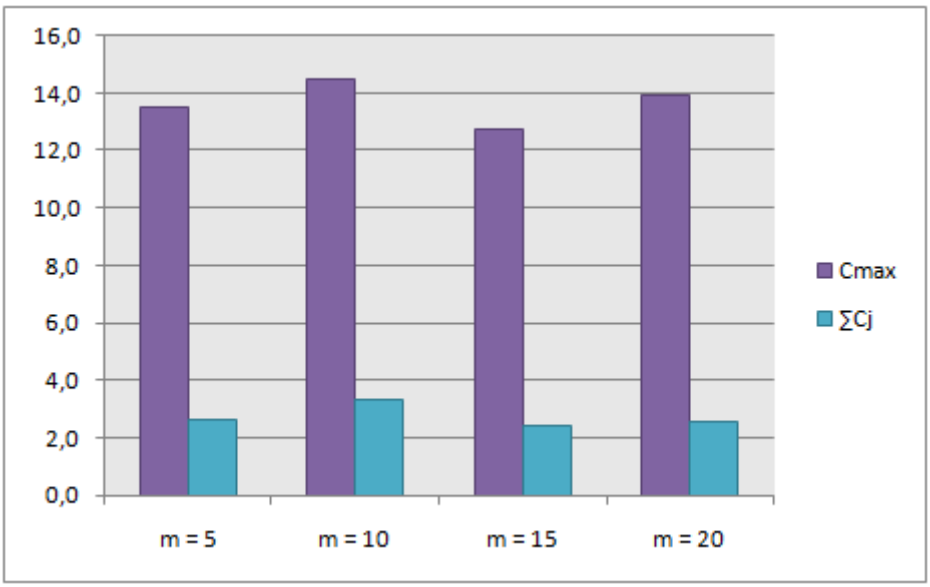


**Figura 6.35** – Incremento relativo de *Cmax* y $\sum C_j$ al usar la secuencia óptima del PFSP en el problema sin esperas permitidas, según el número de máquinas (Elaboración propia)

# 7 CONCLUSIONES

Los trabajos que se procesan en un taller de flujo regular, por lo general, suelen tener tiempos de espera entre dos máquinas consecutivas si al acabar de tratarse en una de ellas la siguiente aún está procesando el trabajo previo. Estas esperas pueden tolerarse en mayor o menor medida, o incluso no permitirse, según las características de dichos trabajos, y esas limitaciones afectan, de distinta forma, a la hora de intentar encontrar el óptimo del problema o soluciones próximas a él. Para analizar dicha influencia, se han estudiado 192 problemas diferentes (definidos por las restricciones aplicadas, el objetivo a minimizar, y el tamaño del problema), que han sido resueltos en 10 ocasiones, pues se generan múltiples iteraciones, cada una de ellas con distintos tiempos de proceso. Así, en cada una de esas 1.920 instancias, se han hallado los resultados que se obtienen siguiendo todas las posibles secuencias de cada problema.

La primera conclusión, y también la más evidente, es que la aplicación de estas restricciones hace que se incremente el valor de los objetivos a minimizar con los que se trabaja, que son el máximo tiempo de finalización (*makespan)* y el tiempo total de finalización ($\sum C_j$). Las restricciones más estrictas, es decir, las que permiten un menor tiempo máximo de espera de los trabajos entre máquinas, tienen un efecto mayor en el aumento del valor de la función objetivo. Se descubre que el tamaño del taller no afecta a dicho incremento, pues con un alto número de máquinas se tienen incrementos relativos del mismo orden que con una cantidad mucho más reducida. El número de trabajos, por su parte, sí afecta, obteniéndose resultados peores al aplicar restricciones a un problema con más trabajos que al aplicar las mismas en otros con menos trabajos. También se comprueba que el incremento relativo del valor del objetivo al limitar los tiempos de espera es considerablemente superior cuando éste es el *makespan* que cuando se intenta minimizar el tiempo total de finalización.

A continuación, se estudia la dificultad de los problemas según las restricciones aplicadas. Para ello, se estudian todas las secuencias de cada iteración generada y se hallan los resultados que aporta cada una de ellas con el fin de analizar la distribución de los resultados que se obtienen con la totalidad de las mismas. Se demuestra que, independiente de cuál de los dos objetivos se esté considerando, limitaciones estrictas aumentan la dificultad del problema, haciendo que sea más costoso encontrar resultados próximos al óptimo. Pero no sólo depende de las restricciones aplicadas la dificultad de los problemas, pues ésta se comprueba que tiene una correlación positiva con el número de trabajos que haya que procesar, y negativa con el número de máquinas que conformen el taller. El objetivo a minimizar no influye en la dificultad del problema siempre que en él se permita que los trabajos esperen indefinidamente entre máquinas. Si no fuera así, tener como objetivo el *makespan* dificulta el problema respecto a cuando se quiere minimizar el tiempo total de finalización.

Tras haber estudiado la dificultad de los problemas, y haber comprobado que hallar el óptimo puede llegar a resultar muy difícil para un número de trabajos a procesar lo suficientemente

alto, se analizan los resultados que se obtienen al seguir la secuencia de trabajos óptima de un problema en otros problemas con diferentes restricciones. En concreto, el análisis se centra en el problema que permite esperas indefinidas (PFSP), siguiéndose en él las secuencias óptimas de los problemas con esperas limitadas o prohibidas, pero también aplicando su óptima en dichos problemas.

Lo primero que se comprueba al respecto es que la secuencia óptima de un problema aporta buenos resultados en otro si sus restricciones son similares (en este caso, si el tiempo máximo de espera es similar), que un alto número de trabajos hace que siguiendo en un problema la secuencia óptima de otro se obtengan peores resultados que en problemas más pequeños, y que el número de máquinas que forman el taller es irrelevante en este contexto.

Analizando los resultados en profundidad, se demuestra que el incremento que sufre el *makespan* en un problema al seguir la óptima de otro es notablemente superior al del tiempo total de finalización, por lo que en problemas con este último valor como objetivo a minimizar es especialmente útil usar la secuencia óptima de un problema similar, pues se siguen obteniendo resultados bastante próximos al óptimo.

La última conclusión que se obtiene respecto al uso de secuencias óptimas de unos problemas en otros es que siguiendo la secuencia óptima de un problema en otro más simple se obtienen resultados mucho más cercanos al óptimo de éste último que en un caso inverso, donde, por ejemplo, se aplicara la secuencia óptima de un PFSP en un problema con la restricción *no-wait.*

Por lo general, los algoritmos que buscan resolver la secuenciación en un taller de flujo regular no consideran limitaciones en los tiempos de espera de los trabajos entre dos máquinas consecutivas, pero es algo que ocurre en diversos tipos de industrias. Mediante este análisis, se ha analizado el efecto de estos tiempos de espera tanto en los distintos objetivos considerados, relacionados con el tiempo de finalización de procesado de los trabajos, como en la dificultad de los problemas según la limitación aplicada.

# 8 BIBLIOGRAFÍA

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

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.*

Pérez, P., Fernández-Viagas, V., Framinan, J. M. (2016) "Métodos de programación de la producción: Exactos", *Apuntes de la asignatura Programación de Operaciones.*

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

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

Ye, H., Li, W., Abedin, A. (2017) "An improved heuristic for no-wait flow shop to minimize makespan", *Journal of Manufacturing System. Editorial Elsevier*

# 9 ANEXOS

## 9.1 Anexo – Funciones

En este anexo se incluyen las once funciones que se usan en el programa y que se nombraron en el capítulo 5, pero en él no se explicó la utilidad de las mismas, cómo estaban construidas, y tampoco la forma en la se que hallaban los resultados de interés.

- Máximo y mínimo de dos números enteros dados

```
int max(int a, int b)
{
     int x;
     if (a>=b)
     {
          x=a;
     }
     else{
          x=b;
     }
     return x;
     }
```

```
int min(int a, int b)
{
     int x;
     if (a>=b)
     {
          x=b;
     }
     else{
          x=a;
     }
     return x;
     }
```

- Factorial de un número entero dado

```
int factorial (int a)
{
     int i;
     int resul=1;
     for(i=a;i>=1;i--)
     {
          resul=resul*i;
     }
     return resul;
}
```

9. Anexos

- *Cmax* obtenido para una secuencia dada aplicada a un PFSP

```
int Cmax(int m, int n, MAT_INT P, VECTOR_INT sec)
{
    MAT_INT C=DIM_MAT_INT(m+1,n+1);
    int i,j;
    for(i=0;i<=m;i++)
    {
        for(j=0;j<=n;j++)
        {
            C[i][j]=0;
        }
    }

    for(i=1;i<=m;i++)
    {
        for(j=1;j<=n;j++)
        {
           C[i][sec[j]]=max(C[i-1][sec[j]],C[i][sec[j-1]])+P[i-1][sec[j]-
1];
        }
    }

    return C[m][sec[n]];
}
```

- $\sum C_j$ obtenido para una secuencia dada aplicada a un PFSP

```
int sumaCj(int m, int n, MAT_INT P, VECTOR_INT sec)
{
    MAT_INT C=DIM_MAT_INT(m+1,n+1);
    int i,j;
    for(i=0;i<=m;i++)
    {
        for(j=0;j<=n;j++)
        {
            C[i][j]=0;
        }
    }

    for(i=1;i<=m;i++)
    {
        for(j=1;j<=n;j++)
        {
           C[i][sec[j]]=max(C[i-1][sec[j]],C[i][sec[j-1]])+P[i-1][sec[j]-
1];
        }
    }


    int resul=0;
    for(j=1;j<=n;j++)
    {
        resul=resul+C[m][j];
    }
    return resul;
}
```

9. Anexos

- *Cmax* obtenido para una secuencia dada aplicada a un problema *no-wait*

```
int CmaxNOWAIT(int m, int n, MAT_INT P, VECTOR_INT sec)
{
    MAT_INT C=DIM_MAT_INT(m+1,n+1);
    int i,j,z,A,dif;
    for(i=0;i<=m;i++)
    {
        for(j=0;j<=n;j++)
        {
            C[i][j]=0;
        }
    }

    for(i=1;i<=m;i++)
    {
        C[i][sec[1]]=C[i-1][sec[1]]+P[i-1][sec[1]-1];
    }

    for(j=2;j<=n;j++)
    {
        for(i=1;i<m;i++)
        {
            C[i][sec[j]]=max(C[i][sec[j-1]],C[i-1][sec[j]])+P[i-
1][sec[j]-1];
            dif=C[i+1][sec[j-1]]-C[i][sec[j]];
            A=max(0,dif);
            for(z=i;z>=1;z--)
            {
             C[z][sec[j]]=C[z][sec[j]]+A;
            }
        }
        C[m][sec[j]]=C[m-1][sec[j]]+P[m-1][sec[j]-1];
    }

    return C[m][sec[n]];
}
```

- $\sum C_j$ obtenido para una secuencia dada aplicada a un problema *no-wait*

```
int CmaxNOWAIT(int m, int n, MAT_INT P, VECTOR_INT sec)
{
    MAT_INT C=DIM_MAT_INT(m+1,n+1);
    int i,j,z,A,dif;
    for(i=0;i<=m;i++)
    {
        for(j=0;j<=n;j++)
        {
            C[i][j]=0;
        }
    }

    for(i=1;i<=m;i++)
    {
        C[i][sec[1]]=C[i-1][sec[1]]+P[i-1][sec[1]-1];
    }

    for(j=2;j<=n;j++)
    {
        for(i=1;i<m;i++)
        {
            C[i][sec[j]]=max(C[i][sec[j-1]],C[i-1][sec[j]])+P[i-
1][sec[j]-1];
            dif=C[i+1][sec[j-1]]-C[i][sec[j]];
            A=max(0,dif);
            for(z=i;z>=1;z--)
            {
             C[z][sec[j]]=C[z][sec[j]]+A;
            }
        }
        C[m][sec[j]]=C[m-1][sec[j]]+P[m-1][sec[j]-1];
    }

    int resul=0;
    for(j=1;j<=n;j++)
    {
        resul=resul+C[m][j];
    }
    return resul;
}
```

- *Cmax* obtenido para una secuencia dada aplicada a un problema con esperas limitadas

```
int CmaxMAXWAIT(int m, int n, MAT_INT P, VECTOR_INT sec, int
waitmax)
{
    MAT_INT C=DIM_MAT_INT(m+1,n+1);
    int i,j,z,A,dif;
    for(i=0;i<=m;i++)
    {
        for(j=0;j<=n;j++)
        {
            C[i][j]=0;
        }
    }

    for(i=1;i<=m;i++)
    {
        C[i][sec[1]]=C[i-1][sec[1]]+P[i-1][sec[1]-1];
    }

    for(j=2;j<=n;j++)
    {
        for(i=1;i<m;i++)
        {
            C[i][sec[j]]=max(C[i][sec[j-1]],C[i-1][sec[j]])+P[i-
1][sec[j]-1];
            dif=C[i+1][sec[j-1]]-C[i][sec[j]]-waitmax;
            A=max(0,dif);
            for(z=i;z>=1;z--)
            {
             C[z][sec[j]]=C[z][sec[j]]+A;
            }
        }
        C[m][sec[j]]=max(C[m-1][sec[j]],C[m][sec[j-1]])+P[m-
1][sec[j]-1];
    }

    return C[m][sec[n]];
}
```

- $\sum C_j$ obtenido para una secuencia dada aplicada a un problema con esperas limitadas

```
int CmaxMAXWAIT(int m, int n, MAT_INT P, VECTOR_INT sec, int
waitmax)
{
    MAT_INT C=DIM_MAT_INT(m+1,n+1);
    int i,j,z,A,dif;
    for(i=0;i<=m;i++)
    {
        for(j=0;j<=n;j++)
        {
            C[i][j]=0;
        }
    }

    for(i=1;i<=m;i++)
    {
        C[i][sec[1]]=C[i-1][sec[1]]+P[i-1][sec[1]-1];
    }

    for(j=2;j<=n;j++)
    {
        for(i=1;i<m;i++)
        {
            C[i][sec[j]]=max(C[i][sec[j-1]],C[i-1][sec[j]])+P[i-
1][sec[j]-1];
            dif=C[i+1][sec[j-1]]-C[i][sec[j]]-waitmax;
            A=max(0,dif);
            for(z=i;z>=1;z--)
            {
             C[z][sec[j]]=C[z][sec[j]]+A;
            }
        }
        C[m][sec[j]]=max(C[m-1][sec[j]],C[m][sec[j-1]])+P[m-
1][sec[j]-1];
    }

    int resul=0;
    for(j=1;j<=n;j++)
    {
        resul=resul+C[m][j];
    }
    return resul;
}
```

9. Anexos

- Primera aplicación de la enumeración completa

```
void enumeracionCompleta1(VECTOR_INT sec, int n, int par_length,
int m, MAT_INT P)
{
    int i;
    if(par_length>1)
    {
        for(i=1;i<par_length;i++)
        {
            int aux=sec[i];
            sec[i]=sec[par_length-1];
            sec[par_length-1]=aux;
            enumeracionCompleta1(sec,n,par_length-1,m,P);
            aux=sec[i];
            sec[i]=sec[par_length-1];
            sec[par_length-1]=aux;
        }
    }
    else
    {
        x1=Cmax(m,n,P,sec);
        x1max=max(x1max,x1);
        x1min=min(x1min,x1);
        if(x1==x1min)
        {
            copyIVector(sec,secMIN[0],n+1);
        }

        x2=CmaxMAXWAIT(m,n,P,sec,40);
        x2max=max(x2max,x2);
        x2min=min(x2min,x2);
        if(x2==x2min)
        {
            copyIVector(sec,secMIN[1],n+1);
        }

        x3=CmaxMAXWAIT(m,n,P,sec,30);
        x3max=max(x3max,x3);
        x3min=min(x3min,x3);
        if(x3==x3min)
        {
            copyIVector(sec,secMIN[2],n+1);
        }

        x4=CmaxMAXWAIT(m,n,P,sec,20);
        x4max=max(x4max,x4);
        x4min=min(x4min,x4);
        if(x4==x4min)
        {
            copyIVector(sec,secMIN[3],n+1);
        }
```

```
            x5=CmaxMAXWAIT(m,n,P,sec,10);
            x5max=max(x5max,x5);
            x5min=min(x5min,x5);
            if(x5==x5min)
            {
                copyIVector(sec,secMIN[4],n+1);
            }

            x6=CmaxNOWAIT(m,n,P,sec);
            x6max=max(x6max,x6);
            x6min=min(x6min,x6);
            if(x6==x6min)
            {
                copyIVector(sec,secMIN[5],n+1);
            }
        }
}
```

Para cada una de las posibles secuencias que se pueden tener, según el número de trabajos que haya que procesar, se obtiene el resultado de cada uno de los seis problemas estudiados. Se guardan los valores mínimo y máximo de la FO de cada problema en las variables creadas con ese propósito.

Asimismo, cuando se halla una secuencia que da una solución mejor que las que se han obtenido previamente, se copia dicha secuencia en la fila correspondiente de una matriz de 6 filas y *n* columnas, donde se recogen las secuencias óptimas de los seis problemas estudiados.

- Segunda aplicación de la enumeración completa

```
void enumeracionCompleta2(VECTOR_INT sec, int n, int par_length,
int m, MAT_INT P)
{
    int i;
    if(par_length>1)
    {
        for(i=1;i<par_length;i++)
        {
            int aux=sec[i];
            sec[i]=sec[par_length-1];
            sec[par_length-1]=aux;
            enumeracionCompleta2(sec,n,par_length-1,m,P);
            aux=sec[i];
            sec[i]=sec[par_length-1];
            sec[par_length-1]=aux;
        }
    }
    else
    {
        x1=Cmax(m,n,P,sec);
        x1tot=x1tot+x1;
        for(i=0;i<100;i++)
        {
            if((x1>=(x1min+0.01*i*(x1max-
x1min)))&(x1<=(x1min+0.01*(i+1)*(x1max-x1min))))
            {
                prob1[i]=prob1[i]+1;
            }
        }

        x2=CmaxMAXWAIT(m,n,P,sec,40);
        x2tot=x2tot+x2;
        for(i=0;i<100;i++)
        {
            if((x2>=(x2min+0.01*i*(x2max-
x2min)))&(x2<=(x2min+0.01*(i+1)*(x2max-x2min))))
            {
                prob2[i]=prob2[i]+1;
            }
        }

        x3=CmaxMAXWAIT(m,n,P,sec,30);
        x3tot=x3tot+x3;
        for(i=0;i<100;i++)
        {
            if((x3>=(x3min+0.01*i*(x3max-
x3min)))&(x3<=(x3min+0.01*(i+1)*(x3max-x3min))))
            {
                prob3[i]=prob3[i]+1;
            }
        }
```

```
            x4=CmaxMAXWAIT(m,n,P,sec,20);
            x4tot=x4tot+x4;
            for(i=0;i<100;i++)
            {
                if((x4>=(x4min+0.01*i*(x4max-
x4min)))&(x4<=(x4min+0.01*(i+1)*(x4max-x4min))))
                {
                    prob4[i]=prob4[i]+1;
                }
            }

            x5=CmaxMAXWAIT(m,n,P,sec,10);
            x5tot=x5tot+x5;
            for(i=0;i<100;i++)
            {
                if((x5>=(x5min+0.01*i*(x5max-
x5min)))&(x5<=(x5min+0.01*(i+1)*(x5max-x5min))))
                {
                    prob5[i]=prob5[i]+1;
                }
            }

            x6=CmaxNOWAIT(m,n,P,sec);
            x6tot=x6tot+x6;
            for(i=0;i<100;i++)
            {
                if((x6>=(x6min+0.01*i*(x6max-
x6min)))&(x6<=(x6min+0.01*(i+1)*(x6max-x6min))))
                {
                    prob6[i]=prob6[i]+1;
                }
            }
        }
}
```

Cuando se aplica por segunda vez la enumeración completa, ya se saben los valores máximos y mínimos de la FO de cada problema.

Para cada una de las posibles secuencias, se halla el resultado que se obtiene en cada problema, y se calcula en cuál de los 100 tramos en los que se divide el intervalo entre los valores extremos se encuentra el resultado, y por tanto, la secuencia en cuestión.

## 9.2 Anexo – Análisis de los resultados mediante distribuciones estadísticas

En el capítulo 6.2.4 se adjuntaba la figura 6.20, donde se representaba la función de distribución de probabilidad de las secuencias de los problemas PFSP y *no-wait* para aquéllos con 5 trabajos. Como el propósito de dicho capítulo era describir el procedimiento para obtener los resultados hallados en él e interpretarlos, se insertó solamente esa figura, pues no procedía incluir también las del resto de problemas. Sin embargo, sí se considera de utilidad hacerlo en este anexo, pues es una oportunidad más para comparar la dificultad de los problemas atendiendo al número de trabajos que se procesan en ellos.

La siguiente figura es la que se mencionaba en el párrafo anterior, que ya aparece previamente en el documento, pero se adjunta de nuevo con el fin de comparar las distribuciones de los distintos problemas según la cantidad de trabajos que se han de procesar.

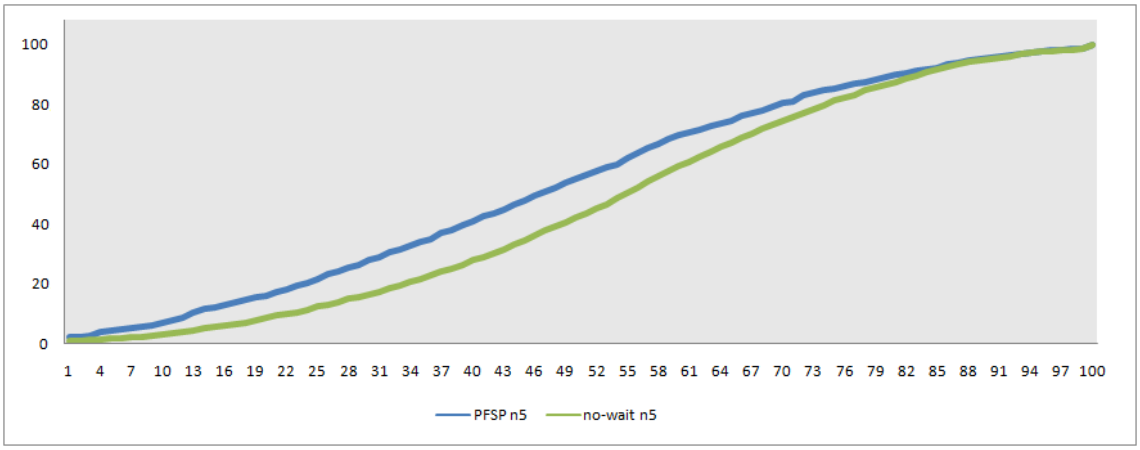


**Figura 9.1** – Funciones de distribución de probabilidad de las secuencias para los problemas PFSP y *no-wait* con 5 trabajos (Elaboración propia)

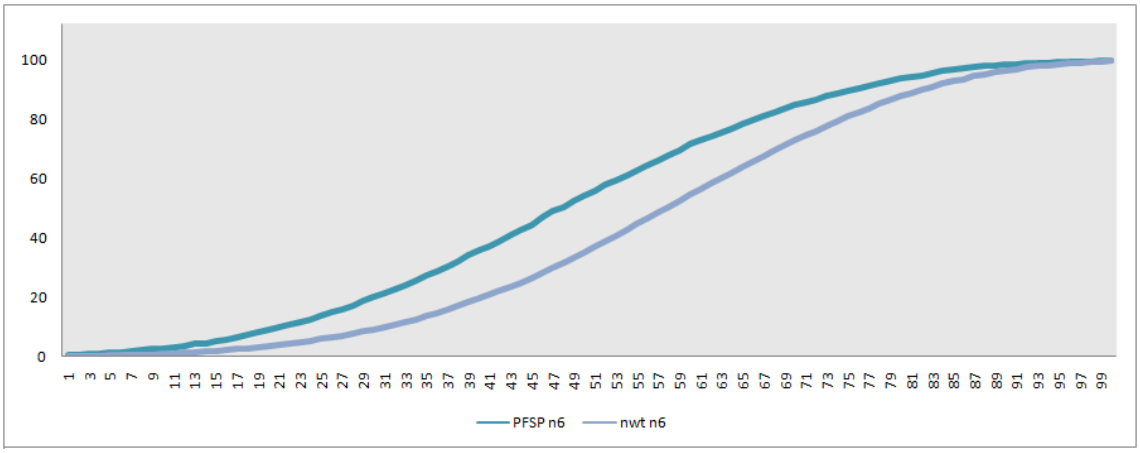


**Figura 9.2** – Funciones de distribución de probabilidad de las secuencias para los problemas PFSP y *no-wait* con 6 trabajos (Elaboración propia)

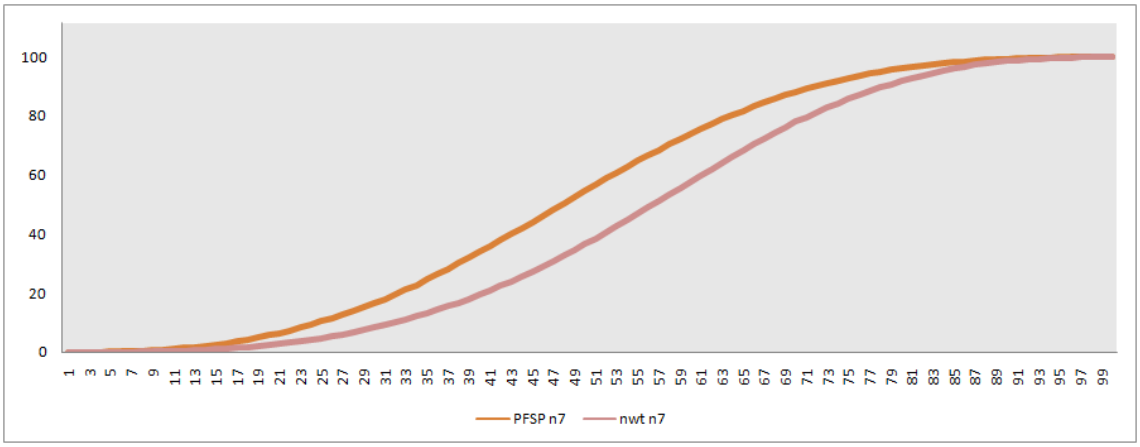


**Figura 9.3** – Funciones de distribución de probabilidad de las secuencias para los problemas PFSP y *no-wait* con 7 trabajos (Elaboración propia)

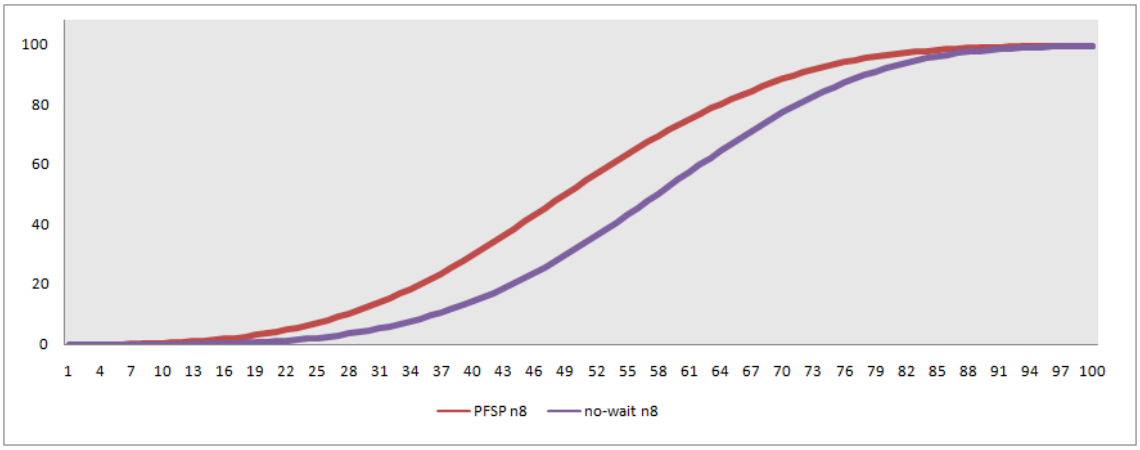


**Figura 9.4** – Funciones de distribución de probabilidad de las secuencias para los problemas PFSP y *no-wait* con 8 trabajos (Elaboración propia)

Al igual que cuando se tenían las distribuciones para los problemas con todas las distintas restricciones contempladas en este proyecto (figura 6.9, por ejemplo), no parece que haya mucha diferencia entre los problemas con 5 y 8 trabajos a simple vista, pues las curvas parecen tener una forma similar. Sin embargo, eso no es así, como se demostró con los cálculos estadísticos del capítulo 6.4, y es que la parte de las gráficas que se debe considerar para estudiar las secuencias próximas al óptimo es la izquierda, y hay una diferencia clara entre los problemas con 5 y 8 máquinas: que en estos últimos la proporción de secuencias que aportan buenos resultados es considerablemente inferior.

## 9.3 Anexo – Secuencias óptimas de problemas con limitaciones en un PFSP

En el capítulo 6.3, cuando se analizaban los resultados que se obtenían al seguir las secuencias óptimas de los problemas con esperas limitadas o prohibidas en un PFSP, se comparaba el incremento relativo del valor del objetivo obtenido aplicando dicha secuencia con el valor del ARPD para el PFSP. Las tablas 6.22, 6.23, 6.24 y 6.25 servían para apreciar que el ARPD era claramente superior al incremento relativo de *Cmax* (6.22 y 6.23) o $\sum C_j$ (6.24 y 6.25), pues se consideró que esa era la conclusión más importante que se podía extraer al respecto. Sin embargo, al haber tanta diferencia entre los valores de las curvas representadas (especialmente en las figuras donde el objetivo era $\sum C_j$), no se apreciaba del todo bien el efecto que el número de trabajos y máquinas tenían en el incremento relativo del valor de la función objetivo al seguir las óptimas de los problemas con esperas limitadas en el PFSP.

Se adjuntan, a continuación, esas mismas cuatro gráficas, pero ahora sin el ARPD de cada problema, de forma que se aprecie de forma mucho más evidente tanto la dependencia positiva entre el número de trabajos y dicho incremento, así como el nulo efecto del número de máquinas. También se observa que cuando el objetivo a minimizar es el tiempo total de finalización, el incremento relativo de su valor es considerablemente inferior al que se da cuando es el *makespan* lo que se trata de minimizar.

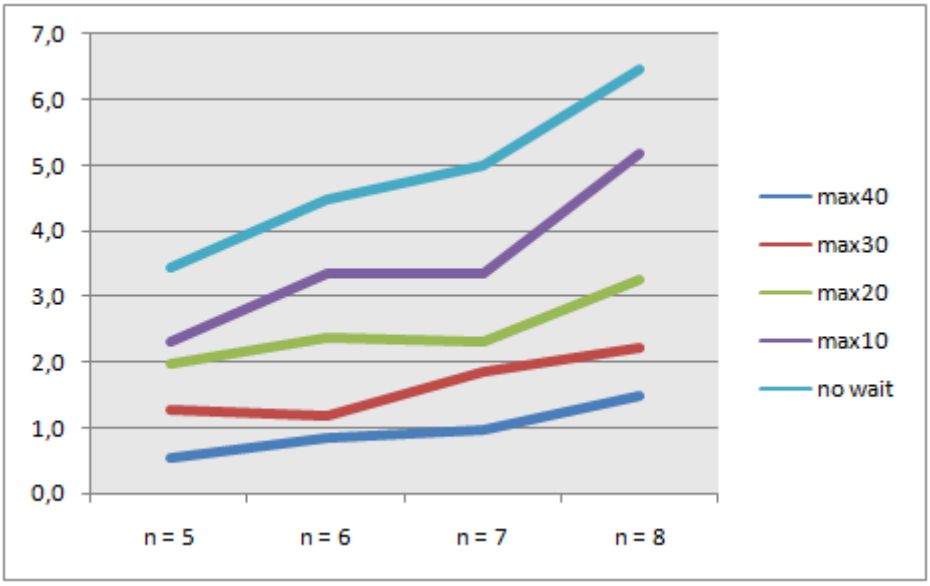


**Figura 9.5** – Incremento relativo de *Cmax* al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, según el número de trabajos (Elaboración propia)

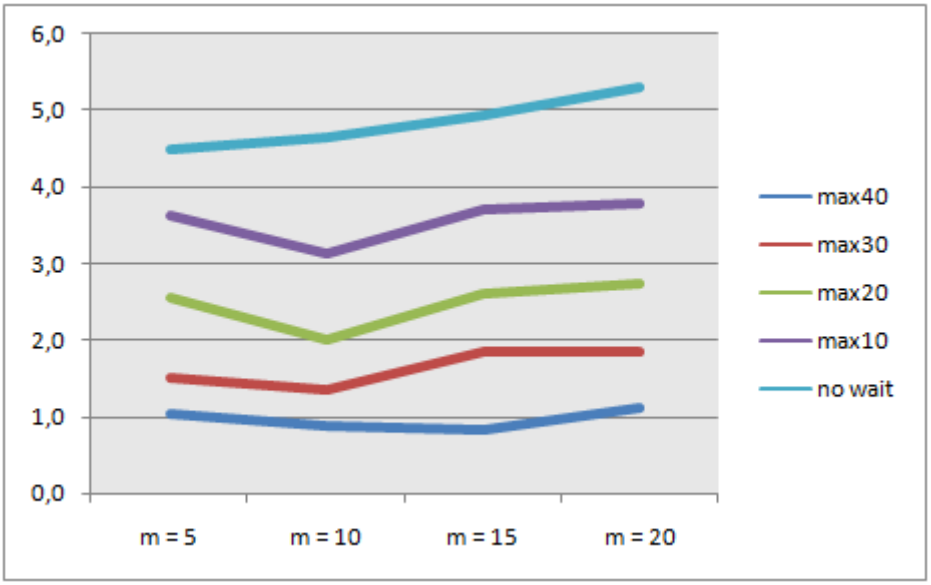


**Figura 9.6** – Incremento relativo de *Cmax* al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, según el número de máquinas (Elaboración propia)

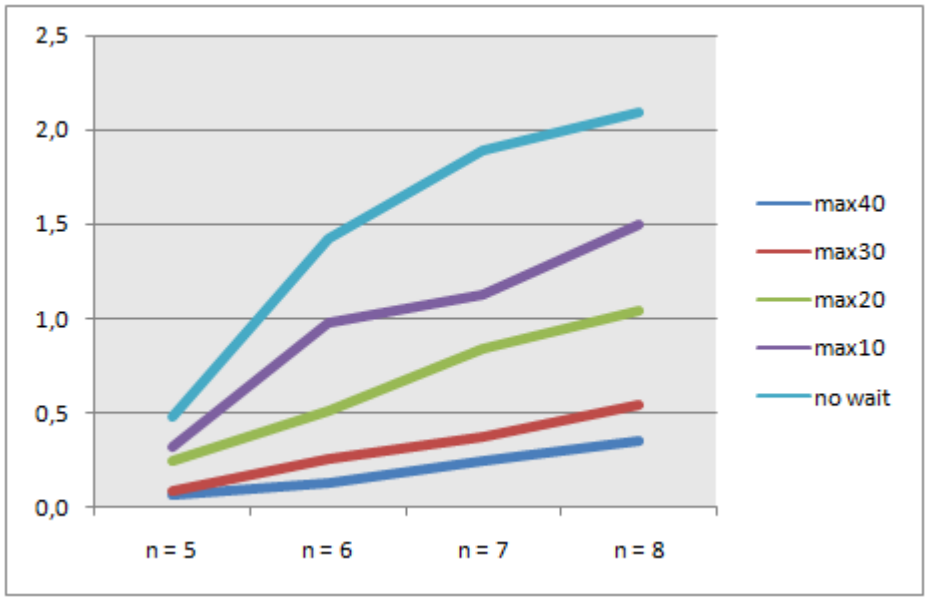


**Figura 9.7** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, según el número de trabajos (Elaboración propia)

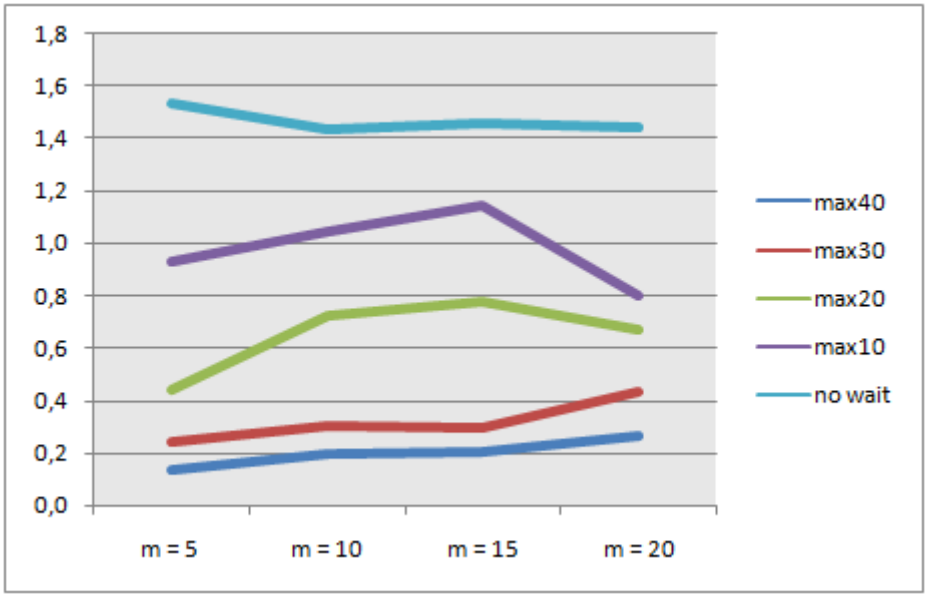


**Figura 9.8** – Incremento relativo de $\sum C_j$ al usar la secuencia óptima de los problemas con esperas limitadas en el PFSP, según el número de máquinas (Elaboración propia)