Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Heurísticas constructivas y de mejora para el problema de ensamblado en dos etapas con objetivo just-in-time

Autor: Javier Núñez Zarza
Tutora: Carla Talens Fayos

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

Sevilla, 2022

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Heurísticas constructivas y de mejora para el problema de ensamblado en dos etapas con objetivo just-in-time

Autor:
Javier Núñez Zarza

Tutora:
Carla Talens Fayos

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Trabajo Final de Grado: Heurísticas constructivas y de mejora para el problema de ensamblado en dos etapas con objetivo just-in-time

Autor: Javier Núñez Zarza

Tutora: Carla Talens Fayos

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia por su apoyo incondicional durante estos últimos 5 años y darme la fuerza necesaria para no rendirme y seguir adelante. En especial a mis abuelos, por creer siempre en mí.

En segundo lugar, a mis amigos. Una segunda familia con la que podré contar toda la vida y que ha hecho que estos 5 años hayan sido más amenos y divertidos.

Por último, no me gustaría despedirme sin agradecer a todos los docentes que me han acompañado y formado durante mi vida académica.

Javier Núñez Zarza

Sevilla, 2022

Resumen

En un entorno de fabricación, son muchos los problemas que pueden dificultar la labor de servir los pedidos a los clientes en tiempo y forma, como pueden ser la falta de materias primas, falta de maquinaria, disponer de poco espacio para el almacenamiento, etc. Todos ellos tienen una raíz común, una mala planificación. Por ello, el objetivo de este proyecto será el de desarrollar herramientas que faciliten esta labor.

En esta ocasión, se ha decidido ponerle solución a un problema de ensamblado en dos etapas, fabricación y montaje, siguiendo una filosofía *just-in-time*, es decir, sin retrasos ni adelantos. Este problema puede encontrarse, por ejemplo, en una pequeña fábrica en la que tengan que fabricarse una serie de piezas (primera etapa) para, posteriormente, montar el producto final (montaje) y que, además, no disponga de mucho espacio para almacenar productos, por lo que estos deben ser entregados poco tiempo después de ser acabado.

Para encontrar esta solución, se pretende desarrollar, al menos, una heurística constructiva que proponga una secuencia de trabajos que permita minimizar retrasos y adelantos con respecto a sus fechas de entrega. Para ello, se propondrán varias heurísticas constructivas diferentes que se combinarán con algunas búsquedas locales. Estas heurísticas se desarrollarán empleando el lenguaje de programación C#.

Abstract

When it comes to manufacturing, there are many problems that can difficult the task of delivering orders to customers in due time and proper form, such as lack of raw materials, lack of machinery, limited storage space, etc. All of these share the same root, bad planning. Therefore, the main objective of this project is to develop a set of tools that facilitate this process.

In this case, the objective is to solve a two-stage assembly problem (manufacturing and assembly), that complies with the just-in-time philosophy, i.e., without delays or advances. This issue can be found, for example, in a small factory where a number of pieces have to be manufactured (first stage) to later assemble the final product (second stage). Moreover, this factory does not have enough storage space, which means that the products need to be delivered shortly after being finished.

In order to find this solution, the aim is to develop, at least, one constructive heuristic that proposes a sequence of jobs that minimizes delays and advances regarding their delivery dates. To do so, a number of different constructive heuristics will be proposed and combined with some local searches. These heuristics will be designed using the C# programming language.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

The great differentiator in business is when an organization stepts out and creates value from something never tried before.

- Kerry Baskins -

Uno de los principales caballos de batalla en la dirección de una planta de fabricación es la planificación de la producción, no tanto por la complejidad de llevarla a cabo, como por la cantidad de factores que influyen en ella. Para poder planificar la producción en un entorno de fabricación real, es indispensable conocer a la perfección los recursos de los que se dispone en cada momento, tanto materiales como humanos, y cuáles son los trabajos por realizar y sus plazos de entrega. Estos factores están sujetos a cambios de forma constante, ya sea por avería de maquinaria como por bajas de personal, lo que hace que se pueda decir que la planificación está viva. Por ello, para poder llevar a cabo este trabajo con éxito, es fundamental disponer de una gran capacidad de adaptación a estos inconvenientes sin que los costes de producción se vean significativamente afectados.

La planificación de la producción es un concepto relativamente reciente que ha evolucionado rápidamente gracias al avance de las nuevas tecnologías. Las primeras fábricas, debido a su simplicidad, trataban de maximizar la productividad de la maquinaria disponible (Herrmann, 1996). Para ello, producían una gran cantidad de un mismo producto durante mucho tiempo continuado, minimizando así el tiempo de preparación de la máquina (Herrmann, 1996). Sin embargo, en la actualidad existen multitud de softwares que son de gran utilidad para llevar a cabo esta tarea, como pueden ser *SAP* o *ManufactMe*. Además, el avance tecnológico permite que las máquinas cada vez se adapten más rápidamente a cambios en los productos a producir, permitiendo así fabricar en lotes más pequeños.

El objetivo de este proyecto es el de desarrollar, al menos, una heurística constructiva capaz de ofrecer una secuencia de asignación de trabajos que minimice el retraso o adelanto de estos en un entorno de ensamblado en dos etapas. Esto puede considerarse un pequeño grano de arena para el desarrollo de nuevos softwares de planificación que permita resolver problemas cada vez más complejos. Para tratar de lograr el objetivo, se propondrán una serie de heurísticas diferentes que, posteriormente, serán combinadas con varias búsquedas locales. De esta forma, se obtendrá un número considerable de posibilidades que ayuden a encontrar la mejor solución.

1.1 Estructura del trabajo

El desarrollo del trabajo se hará conforme a la siguiente estructura de capítulos:

- **Capítulo 2. Marco Teórico.** Se define qué es la programación de operaciones o planificación de la producción mediante unas nociones teóricas básicas. Esto facilitará la comprensión del problema que se desea resolver.
- **Capítulo 3. Descripción del problema.** Se centra en explicar el problema que se desea resolver en base a la teoría desarrollada en el anterior capítulo, definiendo claramente las premisas de las que se parten.
- **Capítulo 4. Metodología.** Describe todos los pasos que se darán para la resolución del problema. Centrándose especialmente en las heurísticas que se proponen.
- **Capítulo 5. Análisis de resultados.** Se mostrarán los resultados obtenidos hacienda uso de tablas y gráficos que faciliten su compresión. Además, se compararán los resultados de cada heurística para decider cuál ofrece mejores resultados.
- **Capítulo 6. Conclusiones.** Se resumirán brevemente las conclusiones obtenidas durante el desarrollo del problema y, especialmente, del análisis de resultados**.**

2 MARCO TEÓRICO

En este capítulo se tratará de realizar una breve introducción a la programación de la producción para facilitar así la compresión del problema objeto de este trabajo, que será descrito en el siguiente capítulo.

2.1 Definición de programación de la producción

El ejercicio de la programación de la producción consiste en asignar, a una serie de trabajos que se deben realizar, los recursos necesarios para completar las tareas de la forma más efectiva posible, es decir, minimizando los costes asociados a los recursos destinados. Es importante destacar que no solo se considera como recurso al material físico empleado para fabricar el producto, sino que el tiempo dedicado a la producción también será considerado como tal. (Pérez, P. et al., 2021)

2.2 Conceptos básicos

A continuación, se van a desarrollar algunos conceptos básicos de la programación de la producción, indicando cuál es la notación de cada uno de ellos. En primer lugar, se verán cuáles son los datos de partida que deben conocerse antes de abordar un problema de programación de operaciones.

- Máquinas (*machines*): cualquier recurso que se considere apto para realizar operaciones que contribuyan a la obtención del producto final. Nótese que el concepto de máquina puede representar a recursos tanto materiales como humanos.
    - Conjunto de máquinas disponibles: $M = \{1, \dots, m\}$
    - Índice para las máquinas: $i \in M$
- Trabajos (*jobs*): cualquier producto que requiera ser procesado en alguna de las máquinas disponibles.
    - Conjunto de trabajos a realizar: $N = \{1, \dots, n\}$
    - Índice para los trabajos: $j \in N$
- Tiempo de proceso (*processing time*): se denota como p_{ij} y es el tiempo que la máquina i tarda en procesar el trabajo j.
- Ruta de proceso (*processing route*): conjunto de máquinas ordenadas que deberán procesar un trabajo j para que este pueda darse por concluído. Se expresa como un vector que se denota como R_j.
- Fecha de llegada (*release date*): instante de tiempo a partir del que se puede comenzar a realizar un trabajo j. Se denota como r_j.
- Fecha de entrega (due date): instante de tiempo en que un trabajo j debe entregarse al cliente. Se denota como d_j.

- Peso: indica la prioridad de un trabajo j y se denota como w_j. Este valor puede ser función de muchos factores, como pueden ser el coste de proceso o la penalización por entrega tardía del trabajo.

Una vez conocidos y asimilados los datos de partida del problema a resolver, se procede a la resolución de este. Para ello, habrá que asignar a cada trabajo todos los recursos necesarios en instantes de tiempo concretos cumpliendo con las restricciones del problema. A esta asignación se le llama programa y, para su mejor comprensión, se suele presentar en un diagrama de Gantt como el que se muestra a continuación.

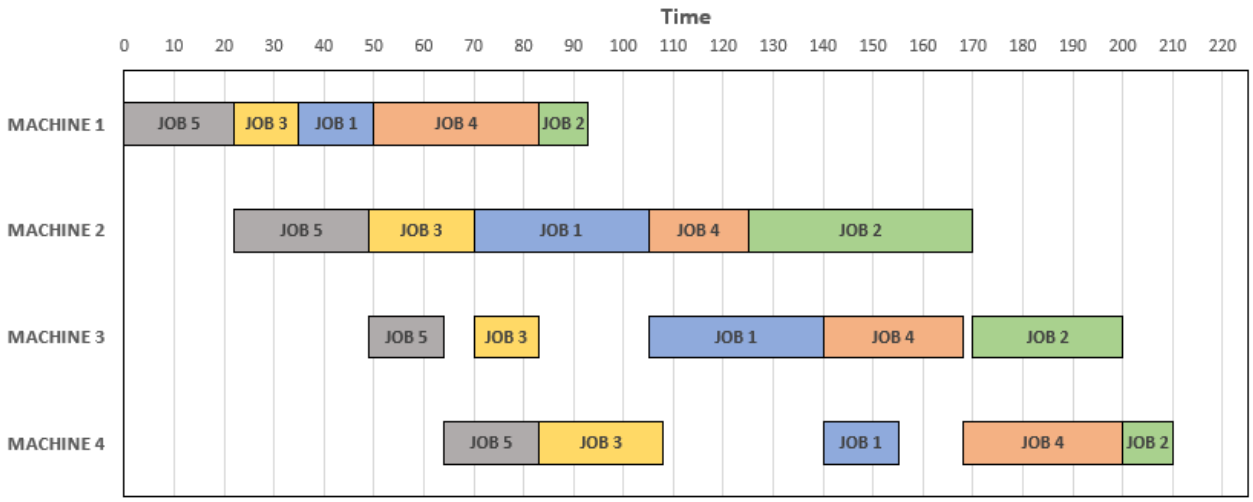

Figura 2-1. Diagram de Gantt.

Se llamará programa admisible a todo aquel que cumpla con las restricciones del problema. De esta forma, para un mismo problema puede haber inifintos programas admisibiles, de los que habrá que elegir uno como solución final. Para la toma de esta decisión, es útil conocer la información que se desarrolla a continuación:

- Secuencia: orden en el que cada máquina procesará los trabajos.
- Tiempo de finalización (*completion time*): instante de tiempo en el que un trabajo j se puede dar por finalizado. Se denota como C_j.
- Retraso (*lateness*): tiempo de retraso con el que se entrega un trabajo j. Se denota por:

$$L_j = C_j - d_j \tag{2–1}$$

- Tiempo de flujo (*flowtime*): tiempo que un trabajo j está en el sistema. Se denota como:

$$F_j = C_j - r_j \tag{2–2}$$

- Tardanza (*tardiness*): tardanza de un trabajo j. Su notación es:

$$T_j = max\{0, L_j\} \tag{2–3}$$

- Adelanto (*earliness*): adelanto de un trabajo j. Su notación es:

$$E_j = max\{0, -L_j\} \tag{2–4}$$

2.3 Modelos de Programación de la Producción

Tras conocer cuáles son los datos de los que parte el problema y los que se desean conocer para darle solución, se pueden clasificar los problemas en base a 3 conceptos: el entorno, las restricciones y el objetivo. Para ello, se emplea la notación $\alpha|\beta|\gamma$, cuyas variables representan respectivamente. (Graham et al., 1979)

2.3.1 Entornos (α)

El entorno en un problema de programación de la producción hace referencia a las máquinas que se emplean y sus características. De esta forma, se pueden encontrar los siguientes entornos:

- Una máquina (*Single machine*): solo hay una máquina por lo que cada trabajo debe pasar por la misma y solo tendrá una operación. La variable α se denota como $\alpha = 1$.
- Máquinas en paralelo (*Parallel machines*): entorno formado por 2 o más máquinas que realizan la misma operación, por lo que cada trabajo se procesará unicamente en una de ellas. Según las características de las máquinas, este entorno puede ser:
    - Máquinas paralelas idénticas (*Identical parallel machines*): todas las máquinas son iguales, por lo que el tiempo de proceso de cada trabajo será el mismo para todas. Este entorno se denota por $\alpha = P_m$, siendo m el número de máquinas.
    - Máquinas paralelas uniformes (*Uniform parallel machines*): las máquinas son similares entre ellas, pero tienen distintas velocidades, por lo que el tiempo de proceso de los trabajos dependerá de la máquina en la que se procese cada uno. Sin embargo, estos variarán de forma proporcional a la velocidad de la máquina. Este entorno se denota por $\alpha = Q_m$, siendo m el número de máquinas.
    - Máquinas paralelas no relacionadas (*Unrelated parallel machines*): las máquinas son diferentes entre sí, por lo que el tiempo de proceso de los trabajos dependerá de la máquina en la que se procese cada uno. Este entorno se denota por $\alpha = R_m$, siendo m el número de máquinas.

- Taller de flujo regular (*Flowshop*): todos los trabajos tienen la misma ruta, es decir, todos pasan por las máquinas en el mismo orden. Este entorno se denota por $\alpha = F_m$, siendo m el número de máquinas.
- Taller (*Jobshop*): cada trabajo tiene una ruta diferente y, por tanto, son un dato indispensable para resolver el problema. Este entorno se denota por $\alpha = J_m$, siendo m el número de máquinas.
- Taller abierto (*Openshop*): es el caso más general y complejo, ya que no existen rutas predeterminadas. Este entorno se denota por $\alpha = O_m$, siendo m el número de máquinas.
- Entornos híbridos (*Hybrid layouts*): es un taller formado por 2 o más etapas. Cada una de estas es un entorno de una máquina o máquinas paralelas. Para la notación, se indica primero qué taller se ha hibridado HF_m, HJ_m, HO_m, y después el entorno de cada una de las fases separados

por comas.

2.3.2 Restricciones (β)

Los modelos también se pueden clasificar en base a sus restricciones. Estas suelen ser:

- Interrupciones: los trabajos se pueden interrumpir mientras se procesan para continuar en otro momento. Existen tres tipos de interrupciones:
    - No reanudables (*non-resumable*): se pierde todo el progreso y en el momento de continuar, hay que empezar la operación desde el principio. Este tipo de restricción se denota de la forma: $\beta = pmtn - non - resumable$
    - Semi-reanudables (*semi-resumable*): se pierde parte del progreso y en el momento de continuar, hay que rehacer una parte de la operación. Este tipo de restricción se denota de la forma: $\beta = pmtn - semi - resumable$
    - Reanudables (*resumable*): en el momento de continuar, se sigue por donde se quedó. Este tipo de restricción se denota de la forma: $\beta = pmtn - resumable$

- Fechas de llegada: los trabajos no están disponibles desde el principio. Esta restricción se denotará como $\beta = r_j$
- Fechas de entrega: los trabajos deben entregarse en un instante determinado. Esta restricción se denotará como $\beta = d_j$
- Tiempos de setup antes de realizar cada trabajo, la máquina necesita de un tiempo de preparación, que puede ser anticipatorio, es decir, se podrá realizar antes de que el trabajo esté disponible, o no. Existen dos tipos:
    - Independientes de la secuencia: tiempo de setup en la máquina i para procesar el trabajo j. Se denota como $\beta = s_{ij}$.
    - Dependientes de la secuencia: tiempo de setup en la máquina i para procesar el trabajo k despues de procesar el trabajo j. Se denota como $\beta = s_{ijk}$.
- Lotes: las máquinas pueden procesar a la vez b trabajos. Existen dos restricciones asociadas a los lotes:
    - Lotes en paralelo (*Parallel batching*): el tiempo de proceso del lote será el mayor tiempo de proceso de los trabajos pertenecientes al lote. Se denota como $\beta = p - batch(b)$.
    - Lotes en serie (*Serial batching*): el tiempo de proceso del lote será la suma de los tiempos de proceso de los trabajos pertenecientes al lote. Se denota como $\beta = s - batch(b)$.

- Precedencia: un trabajo no se puede realizar hasta que no se hagan los que le preceden. Se denota como $\beta = prec$.
- Taller de flujo regular de permutación: todos los trabajos tienen la misma ruta y todas las máquinas la misma secuencia. Se denota como $\beta = prmu$.
- No se permiten tiempos ociosos de las máquinas, es decir, una vez comienza a trabajar no

puede parar hasta finalizar todos los trabajos. Se denota como $\beta = no - idle$.

- Los trabajos no pueden esperar entre máquinas. Se denota como $\beta = no - wait$.
- Las máquinas tienen un buffer de capacidad limitada. Se denota como $\beta = b_i$.

2.3.3 Objetivos (γ)

Por último, los modelos se pueden clasificar en base a sus objetivos. Los más comunes son los propuestos por Graham et al. en (1979):

- *Makespan*: tiempo de terminación más alto. Se denota como:

$$\gamma = Max\ C_j \tag{2–5}$$

- *Total Completion Time*: sumatorio de los tiempos de terminación. Se denota como:

$$\gamma = \sum C_j \tag{2–6}$$

- *Maximum flowtime*: tiempo de flujo más alto. El tiempo de flujo se define como el tiempo que pasa un trabajo en el entorno de fabricación, es decir, el tiempo entre el primer instante en el que se puede realizar el trabajo y el instante en que se da por terminado este. Se denota como:

$$\gamma = Max\ F_j \tag{2–7}$$

- *Total Flowtime*: sumatorio de los tiempos de flujo. Se denota como:

$$\gamma = \sum F_j \tag{2–8}$$

- *Maximum lateness*: tiempo de retraso más alto. Cuando un trabajo se adelanta, se expresa su Adelanto con un valor negative. Se denota como:

$$\gamma = Max\ L_j \tag{2–9}$$

- *Total Lateness*: sumatorio de los tiempos de retraso. Se denota como:

$$\gamma = \sum L_j \tag{2–10}$$

- *Maximum Tardiness*: tiempo de tardanza más alto. Cuando un trabajo se adelanta, la tardanza se considera nula. Se denota como:

$$\gamma = Max\ T_j \tag{2–11}$$

- *Total Tardiness*: sumatorio de los tiempos de tardanza. Se denota como:

$$\gamma = \sum T_j \tag{2–12}$$

- *Maximum Earliness*: tiempo de adelanto más alto. Si un trabajo se atrasa, el adelanto se considera nulo. Se denota como:

$$\gamma = Max\ E_j \tag{2–13}$$

- *Total Earliness*: sumatorio de los tiempos de adelanto. Se denota como:

$$\gamma = \sum E_j \tag{2–14}$$

- Todos los obejtivos anteriores pueden ponderarse multiplicando cada una de las medidas por su peso correspondiente w_j.

3 DESCRIPCIÓN DEL PROBLEMA

A lo largo de este capítulo, se va a describir, en base a lo visto en el capítulo anterior, el problema objeto de este proyecto.

3.1 Modelo

El modelo que describe el problema es:

$$HF_2, DP_m, R_2|d_j| \sum E_j + T_j \tag{3–1}$$

- Entorno: se trata de un taller de flujo regular hibridado con dos fases. En la primera fase existen m máquinas dedicadas. Cada una de ellas fabrica uno de los m componentes del producto final. Una vez todos los componentes están fabricados, pasan a la segunda etapa, donde hay 2 máquinas paralelas no relacionadas de ensamblado que serán las encargadas de formar el producto final. Al tratarse de un entorno de máquinas paralelas, cada trabajo solo deberá ser procesado por una de las máquinas. A la hora de decidir en cuál de las máquinas de ensamblado se procesará cada trabajo, hay que enfrentarse a un problema de asignación. Para resolverlo, se aplicarán dos criterios, *Earliest Completion Time* (ECT), en el que se elegirá la máquina en la que el trabajo finalice antes, o *First Available Machine* (FAM), en el que se asignará a la primera máquina disponible.
- Restricciones: en este problema existen unas fechas de entrega no obligatorias, es decir, pueden entregarse los trabajos antes o después de esa fecha. Sin embargo, para la resolución del problema se va a seguir una filosofía *just-in-time*, lo que significa que se tratará de entregar cada trabajo en la fecha indicada, ni antes ni después.
- Objetivo: tal y como se ha adelantado, este problema se va a resolver de acuerdo con la filosofía *just-in-time*, es decir, cada trabajo debe ser terminado justo en su fecha de entrega, ni antes ni después. En caso de no poder cumplirse, deberá ser terminado en el instante más cercano posible a su fecha de entrega. La penalización por entregar antes o después de la fecha es la misma, por lo que se minimizará la suma del total *earliness* y del total *tardiness*.

Además de esto, se han tomado varias hipótesis de partida:

- El *buffer* de cada máquina se considera ilimitado.
- El tiempo de transporte es despreciable.
- No hay tiempos de preparación de máquinas.
- Todos los trabajos están disponibles desde el principio.
- Los trabajos no pueden ser interrumpidos.
- Los tiempos de proceso y fechas de entrega de los trabajos son conocidos.

3.2 Resolución de ejemplo

En este apartado, para facilitar la comprensión del problema, se va a desarrollar un pequeño ejemplo resuelto de forma manual.

3.2.1 Datos de partida

Se van a procesar 6 trabajos (a partir de ahora $J_n \forall n = 1, \dots, 6$) en 4 máquinas dedicadas (denotadas a partir de ahora como $MD_m \forall m = 1, \dots, 4$) y, posteriormente, se ensamblarán en una de las 2 máquinas de ensamblado disponibles (denotadas a partir de ahora como $ME_m \forall m = 1,2$). Los tiempos de proceso de cada trabajo en cada máquina se muestran a continuación.

Tabla 3–1. Tiempos de proceso

	J_1	J_2	J_3	J_4	J_5	J_6
MD_1	2	4	6	8	10	12
MD_2	1	3	5	7	9	11
MD_3	1	2	3	4	5	6
MD_4	5	2	3	7	1	6
ME_1	1	2	11	15	19	21
ME_2	15	20	14	9	22	17

Además de esto, cada trabajo tendrá que ser entregado en una fecha de terminada. Estas fechas se conocen como due dates (a partir de ahora d_j) y se muestran en la siguiente tabla.

Tabla 3–2. Fechas de entrega

	J_1	J_2	J_3	J_4	J_5	J_6
d_j	20	31	26	43	46	63

3.2.2 Resolución

Una vez conocidos y asimilados todos los datos de partida, se procede a resolver el problema en cuestión. En primer lugar, hay que elegir una secuencia de asignación de trabajos a cada máquina. Habitualmente, este es el mayor problema de la programación de operaciones, puesto que una mala secuencia de asignación hace que se quede muy lejos de lograr el objetivo. Sin embargo, al tratarse este problema de un ejemplo, se elegirá una secuencia (S) sencilla como S = [1, 2, 3, 4, 5, 6], sin importar que ofrezca malos resultados.

Teniendo la secuencia, se asignan todos los trabajos en el mismo orden a cada máquina dedicada, cuidando que se haya terminado el trabajo anterior antes de introducir el siguiente. De esta forma, se obtiene el siguiente diagrama de Gantt.

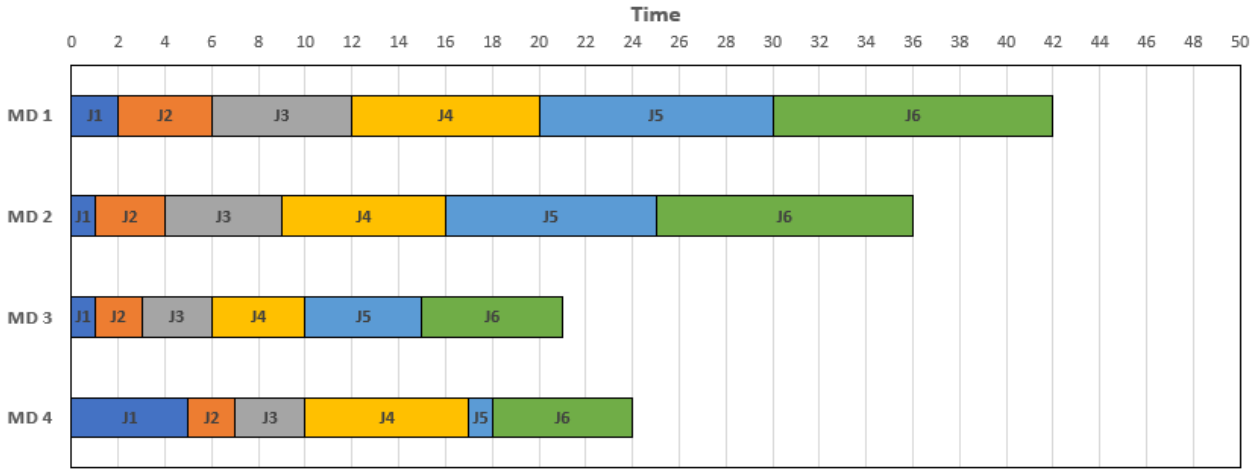

Figura 3-1. Diagrama de máquinas dedicadas

Tras esto, podemos conocer en qué instante termina cada trabajo la primera fase y, por tanto, están listos para comenzar la etapa de ensamblado. En esta fase, los trabajos se irán procesando, en el mismo orden en el que van terminando la primera etapa, en una de las dos máquinas disponibles. Para asignar cada trabajo a una máquina, se aplicará el criterio ECT (Earliest Completion Time), es decir, se estudiará cuál sería el completion time del trabajo en cada máquina y se asignará a la que tenga un menor valor de C_j. De esta forma, la asignación queda como se muestra a continuación.

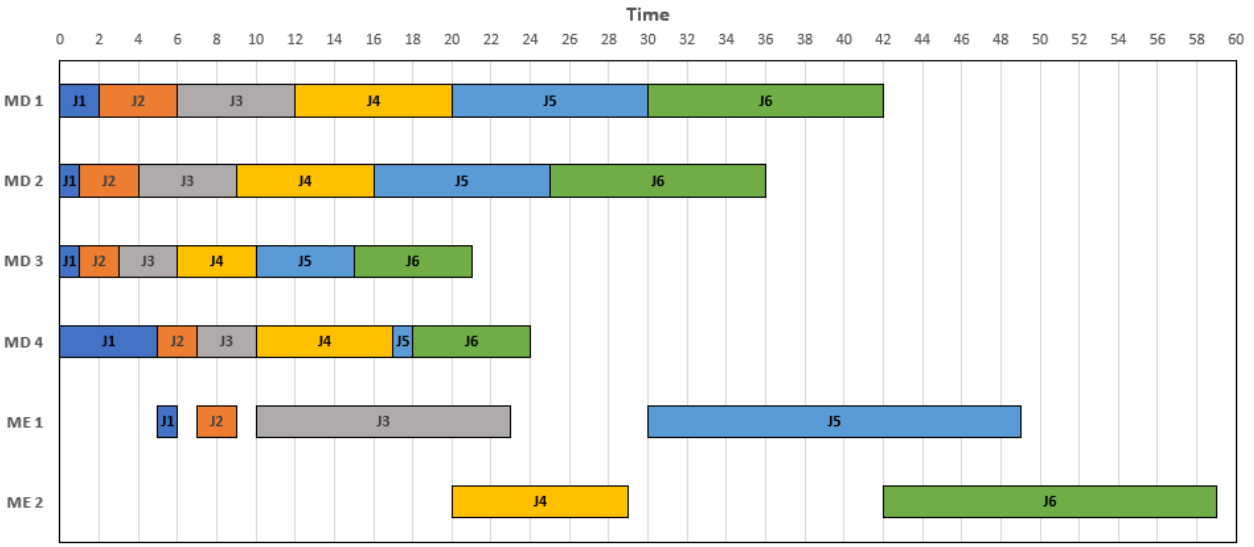

Figura 3-2. Diagrama de máquinas dedicadas y ensamblado

De este diagrama se puede obtener gran cantidad de información, como los completion time de todos los trabajos en cada máquina y etapa o el tiempo ocioso de cada máquina, es decir, cuánto tiempo está disponible la máquina sin realizar ningún trabajo. Sin embargo, para el estudio de la función objetivo del problema basta con saber en qué instante se finaliza cada trabajo. Con estos datos y con las fechas de entrega vistas previamente, se puede calcular el *earliness* y el *tardiness* de cada trabajo para, posteriormente, calcular el sumatorio de todos. Para facilitar esta labor, se puede organizar la información en una tabla como la que se muestra.

Tabla 3–1. Resumen cálculo de función objetivo

	J_1	J_2	J_3	J_4	J_5	J_6
C_j	6	9	23	29	49	59
dd_j	20	31	26	43	46	63
E_j	14	22	3	14	0	4
T_j	0	0	0	0	3	0
$E_j + T_j$	14	22	3	14	3	4

Con esta información podemos ver que el sumatorio del *Total Earliness* mas el *Total Tardiness* es 60 y se puede dar por concluido el problema.

4 METODOLOGÍA

En este capítulo se describirá la metodología empleada para el estudio del problema objeto de este proyecto. Como se puede deducir de lo descrito en los anteriores capítulos, la resolución de un caso real de este problema se puede dividir en tres etapas: elección de una secuencia de asignación, cálculo de la función objetivo y análisis de los resultados obtenidos.

En primer lugar, hay que tomar la decisión más importante, que no es otra que determinar en qué orden se van a realizar los trabajos y, en caso de varias opciones, en qué maquina se hará cada uno. Esta es la única etapa en la resolución del problema en la que se puede participar de forma activa y, por tanto, alterar los resultados que se obtendrán.

En segundo lugar, cuando se conoce la secuencia de asignación, habrá que calcular el valor de la función objetivo, previamente definida, que ofrece la secuencia elegida. Esta etapa se llevará a cabo siempre de la misma forma, tal y como se describió en el capítulo anterior.

Por último, se procederá con el análisis de los resultados. En un entorno de fabricación real, este es un punto de gran importancia, ya que puede ofrecer las claves de la calidad del servicio ofrecido a los clientes. En este caso, el resultado final será el sumatorio del total earliness más el total tardiness, lo que mostrará una visión global del cumplimiento de las fechas de entrega impuestas. Sin embargo, centrarse en este valor final únicamente puede acabar siendo engañoso, ya que no se puede saber si el resultado final se obtiene por el retraso o adelanto de un único trabajo, o de todos ellos. Es por esto, que resultará de gran valor el análisis de cada caso concreto para así conocer cuál puede ser el grado de satisfacción de cada cliente en lo que al cumplimiento de plazos se refiere.

Conociendo todas las etapas de resolución de un caso real de este problema, se observa claramente que, si la secuencia de asignación de trabajos es suficientemente buena, se cumplirán con éxito todos, en caso de ser imposible casi todos, los plazos de entrega deseados. Es por ello por lo que este capítulo se dedicará mayormente a detallar exhaustivamente una serie de heurísticas propuestas para el cálculo de la mejor secuencia posible en base a los datos de partida del problema.

4.1 Generación de instancias

Para poder conocer cuál de las 42 heurísticas disponibles es la mejor para cada caso, es necesario saber cómo se comporta cada una de ellas ante diferentes tipos de problemas, por lo que habrá que generar instancias de diferentes tamaños. Para generarlas, es necesario definir los parámetros del problema:

- Número de trabajos (n): $n = \{50, 100, 150, 200, 250\}$.
- Número de máquinas dedicadas (m_{ded}): $m_{ded} = \{2, 4, 6\}$.
- Número de máquinas de ensamblado (m_{ens}): $m_{ens} = 2$.
- Tiempos de proceso en la primera etapa (t_{ij}): $t_{ij} \in U[1, 100]$
- Tiempos de proceso en la segunda etapa (t_{ij}): $at_j \in U[1, m_2 \cdot 100]$

Con estos parámetros, se podrán generar de forma aleatoria los tiempos de proceso de cada trabajo en cada máquina. Sin embargo, el cálculo de las fechas de entrega es algo más complejo, ya que unas fechas de entrega inadecuadas pueden hacer que se trate de un problema que no sea factible resolver en un entorno real. Por ello, las fechas de entrega se generarán siguiendo el procedimiento descrito por

Potts y VanWassenhove (1982), siguiendo una distribución uniforme entre $P(1 - T - \frac{R}{2})$ y $P(1 - T + \frac{R}{2})$. Donde *P* es un límite inferior para la función objetivo, y *R* y *T* son parámetros que toman los valores $R = \{0.2, 0.6, 1\}$ y $T = \{0.25, 0.5, 0.75\}$.

De esta forma, teniendo en cuenta todos los valores posibles de los parámetros, se obtienen 135 combinaciones posibles para las que se generarán 5 instancias diferentes, es decir, se dispondrá de 675 instancias diferentes para evaluar cada una de las heurísticas.

4.2 Heurísticas constructivas

Con el objetivo de encontrar la mejor secuencia posible sean cuales sean los datos de partida del problema, se proponen 6 alternativas diferentes que, combinadas con algunas búsquedas locales que se desarrollarán posteriormente, resultarán en 42 heurísticas diferentes. Todas ellas toman como punto de partida la heurística propuesta por Framiñán y Pérez-Gónzalez (2017) en su artículo *Order scheduling with tardiness objective: Improved approximate solutions.*

Procedure *FP*
- $\Pi := \varnothing$;
- Obtain sequence $\Omega := (\omega_1, \ldots, \omega_n)$ by sorting orders according to EDD rule;
- Set $ct_i = 0$ $(1 \leq i \leq m)$;
- // Append orders one by one
- **for** $j = 1$ **to** n **do**
  - // Compute indicator for each unscheduled order
  - **for each** $\omega_l \in \Omega$ **do**
    - // Compute ect_i the estimated completion times of w_l for each machine i;
    - **for** $i = 1$ **to** m **do**
      - $ect_i = ct_i + p_{i,w_l}$;
    - **end**
    - // Compute first term of indicator η_l, see Equation (8)
    - $\eta_l = \max\left\{\max_{1\leq i\leq m}\{ect_i\} - d_{w_l}; 0\right\}$;
    - // Compute second term of indicator η_l
    - **for each** $w_k \in \Omega - \{w_l\}$ **do**
      - **for** $i = 1$ **to** m **do**
        - $ect_i = ect_i + p_{i,w_k}$;
      - **end**
      - $\eta_l = \eta_l + \max\left\{\max_{1\leq i\leq m}\{ect_i\} - d_{w_k}; 0\right\}$;
    - **end**
  - **end**
  - // Select order with minimum value of η_l
  - $r := \text{argmin}_{1\leq l\leq n-j+1}\, \eta_l$;
  - Append ω_r at the end of Π, i.e. $\Pi := (\pi_1, \ldots, \pi_{j-1}, \omega_r)$;
  - Extract ω_r from Ω, i.e. $\Omega := (\omega_1, \ldots, \omega_{r-1}, \omega_{r+1}, \ldots, \omega_{n-j+1})$;
  - // Update completion times
  - **for** $i = 1$ **to** m **do**
    - $ct_i = ct_i + p_{i,w_r}$;
  - **end**
- **end**

end

Figura 4-1. Pseudocódigo heurística FP (Perez-Gonzalez y Framiñán, 2017)

Para adaptar esta heurística al problema que se trata en este proyecto, se realizan los cambios que se enumeran.

- Al tratarse de un entorno con dos etapas, el cálculo de los completion time se realiza de forma diferente. En primer lugar, hay que calcular el completion time del trabajo en cuestión en cada una de las máquinas dedicadas y, al mayor de estos tiempos, sumarle el tiempo de proceso del trabajo en la máquina de ensamblado a la que se haya asignado. Esto solo se hará así si la máquina de ensamblado elegida esta libre en el instante en el que el trabajo termina por completo la primea etapa. Si la máquina estuviera ocupada, habrá que tomar el instante de tiempo en el que se quede libre y sumarle el tiempo de proceso del trabajo en la máquina en cuestión.
- Para almacenar los valores de los completion time de los trabajos ya secuenciados, serán necesarias más variables auxiliares.
- Como la función objetivo del problema es distinta del problema para el que se construyó la heurística, el cálculo del indicador será diferente al propuesto. Este será el factor diferenciador entre las 6 heurísticas que se van a desarrollar a lo largo de este capítulo.

De esta forma, la estructura principal de las heurísticas que se van a proponer es la siguiente.

Procedure *Constructive Heuristic*

$\prod := \emptyset$; [1]
Obtain sequence $\Omega := (\omega_1, \ldots, \omega_n)$ by sorting orders according to EDD rule; [2]
Set $ct1def_i = 0$ $(1 \leq i \leq m_{ded})$; [3]
Set $ct2def_i = 0$ $(1 \leq i \leq m_{ens})$; [4]
for $j = 1$ **to** n **do** [5]
  $ct1 := ct1def$; [6]
  $ct2 := ct2def$; [7]
  for each $\omega_l \in \Omega$ **do** [8]
    for $i = 1$ **to** m_{ded} **do** [9]
      $ct1_i = ct1_i + pt_{ded_{i,\omega_l}}$; [10]
    end [11]
    for $i = 1$ **to** m_{ens} **do** [12]
      $ect_i = \max\{\max\{ct1\}; ct2_i\} + pt_{ens_{i,\omega_l}}$; [13]
    end [14]
    $r := \text{argmin}_{1 \leq i \leq m_{ens}} ect_i$; [15]
    $ct2_r = ect_r$; [16]
    //Indicator's first phase [17]
    for each $\omega_k \in \Omega$ **do** [18]
      for $i = 1$ **to** m_{ded} **do** [19]
        $ct1_i = ct1_i + pt_{ded_{i,\omega_k}}$; [20]
      end [21]
      for $i = 1$ **to** m_{ens} **do** [22]
        $ect_i = \max\{\max\{ct1\}; ct2_i\} + pt_{ens_{i,\omega_k}}$; [23]
      end [24]
      $r := \text{argmin}_{1 \leq i \leq m_{ens}} ect_i$; [25]
      $ct2_r = ect_r$; [26]
      //Indicator's second phase [27]
    end [28]
  end [29]
  $r := \text{argmin}_{1 \leq l \leq n-j+1} \eta_l$; [30]
  Append ω_r at the end of $\prod$; [31]
  Extract ω_r from Ω; [32]
  for $i = 1$ **to** m_{ded} **do** [33]
    $ct1def_i = ct1def_i + pt_{ded_{i,\omega_r}}$; [34]
  end [35]
  for $i = 1$ **to** m_{ens} **do** [36]
    $ect_i = \max\{\max\{ct1def\}; ct2def_i\} + pt_{ens_{i,\omega_r}}$; [37]
  end [38]
  $r := \text{argmin}_{1 \leq i \leq m_{ens}} ect_i$; [39]
  $ct2def_r = ect_r$ [40]
end [41]
end

Figura 4-2. Estructura principal heurísticas

4.2.1 Heurística 1

En esta primera heurística, se opta por penalizar tanto el tardiness como el earliness de cada trabajo. De esta forma, el indicador se calculará como el valor absoluto de la diferencia entre la fecha de entrega y el completion time de cada trabajo. Además, el cálculo del índice se realiza en dos fases: una primera en la que se toma en cuenta el retraso o adelanto del trabajo que se quiere secuenciar, y una segunda en la que, al valor anterior, se le suman las penalizaciones del resto de trabajos que quedan sin secuenciar si fueran secuenciados según el criterio EDD.

En consecuencia con lo explicado, habrá que incluir en el pseudocódigo las siguientes ecuaciones en las líneas 17 y 27 respectivamente:

$$\eta_{\omega_l} = \left| ct2_r - d_{\omega_l} \right| \tag{4–1}$$

$$\eta_{\omega_l} = \eta_{\omega_l} + \left| ct2_r - d_{\omega_k} \right| \tag{4–2}$$

4.2.2 Heurística 2

Para la segunda heurística, se ha considerado obviar el cálculo de la segunda etapa del indicador, reduciendo así el número de operaciones de la heurística y, por tanto, su tiempo de ejecución. El objetivo que se persigue con esta heurística es comprobar si penalizar los trabajos que no se tienen en cuenta para la secuenciación en ese instante añaden un valor significativo. En caso de no añadirlo, se lograría reducir notablemente el tiempo de ejecución de esta, traduciéndose esto en mayor eficacia para el usuario.

De esta forma, el pseudocódigo cambiaría notablemente con respecto al de la primera heurística, quedando como se muestra en la *Figura 4-3. Pseudocódigo de la heurística 2*.

4.2.3 Heurística 3

En este caso, el indicador solo penalizará el retraso, si lo hubiera, de los trabajos, calculándose como el mayor valor entre la diferencia del completion time menos la fecha de entrega y cero. Además, el cálculo se realizará en las mismas dos fases que en la heurística 1.

Así, habrá que añadir las siguientes ecuaciones, en las líneas 17 y 27 respectivamente, al pseudocódigo mostrado en la *Figura 4-2. Estructura principal heurísticas*.

$$\eta_{\omega_l} = max\{ct2_r - d_{\omega_l};\ 0\} \tag{4–3}$$

$$\eta_{\omega_l} = \eta_{\omega_l} + max\{ct2_r - d_{\omega_k};\ 0\} \tag{4–4}$$

4.2.4 Heurística 4

Al igual que en la heurística 2, en este caso el indicador se calculará únicamente en una fase, la primera de la heurística 3. De esta forma, habrá que sustituir en el pseudocódigo mostrado en la *Figura 4-3. Pseudocódigo de la heurística 2*, y en la línea 17, la siguiente ecuación:

$$\eta_{\omega_l} = max\{ct2_r - d_{\omega_l};\ 0\} \tag{4–5}$$

Procedure *Constructive Heuristic*

```
∏ := Ø;                                                                    [1]
Obtain sequence Ω := (ω1, ... , ωn) by sorting orders according to EDD rule;   [2]
Set ct1def_i = 0 (1 ≤ i ≤ m_ded);                                          [3]
Set ct2def_i = 0 (1 ≤ i ≤ m_ens);                                          [4]
for j = 1 to n do                                                          [5]
    ct1 := ct1def;                                                         [6]
    ct2 := ct2def;                                                         [7]
    for each ωl ∈ Ω do                                                     [8]
        for i = 1 to m_ded do                                              [9]
            ct1_i = ct1_i + pt_ded_i,ωl;                                   [10]
        end                                                                [11]
        for i = 1 to m_ens do                                              [12]
            ect_i = max{ max{ct1}; ct2_i} + pt_ens_i,ωl;                   [13]
        end                                                                [14]
        r := argmin_1≤i≤m_ens ect_i;                                       [15]
        ct2_r = ect_r;                                                     [16]
        η_ωl = |ct2_r - d_ωl|                                              [17]
    end                                                                    [18]
    r := argmin_1≤l≤n-j+1 η_l;                                             [19]
    Append ω_r at the end of ∏;                                            [20]
    Extract ω_r from Ω;                                                    [21]
    for i = 1 to m_ded do                                                  [22]
        ct1def_i = ct1def_i + pt_ded_i,ω_r;                                [23]
    end                                                                    [24]
    for i = 1 to m_ens do                                                  [25]
        ect_i = max{ max{ct1def}; ct2def_i} + pt_ens_i,ω_r;                [26]
    end                                                                    [27]
    r := argmin_1≤i≤m_ens ect_i;                                           [28]
    ct2def_r = ect_r                                                       [29]
end                                                                        [30]
end
```

Figura 4-3. Pseudocódigo heurística 2

4.2.5 Heurística 5

Ahora, a diferencia de la decisión tomada en las anteriores heurísticas, solo se tendrá en cuenta el instante de finalización de los trabajos. Para proceder de esta forma, el indicador se calculará, en las mismas dos fases que las heurísticas 1 y 3, como el *earliest completion time* de cada trabajo en la etapa de ensamblado. Como resultado de lo desarrollado, se incluirán, en las líneas 17 y 27 respectivamente, las siguientes ecuaciones al pseudocódigo de la *Figura 4-2. Esqueleto principal heurísticas*.

$$\eta_{\omega_l} = ect_r \tag{4–6}$$

$$\eta_{\omega_l} = \eta_{\omega_l} + ect_r \tag{4–7}$$

4.2.6 Heurística 6

Para calcular el indicador de esta heurística, al igual que con la 2 y la 4, se realizará solo la primera fase del indicador de la heurística 5. Por tanto, al pseudocódigo de la *Figura 4-3. Pseudocódigo de la heurística 2* habrá que añadirle, en la línea 17, la ecuación:

$$\eta_{\omega_l} = ect_r \tag{4–8}$$

4.3 Búsquedas locales

El fin último de cada una de las heurísticas es encontrar una secuencia que ofrezca el mejor valor posible de la función objetivo. Con el objetivo de que esta secuencia se acerque lo máximo posible a la secuencia óptima, si es que esta existiese, se va a estudiar la posibilidad de encontrar una secuencia mejor que la obtenida, partiendo de esta como solución inicial. Para ello, es necesario conocer el concepto de vecindad, que se define como el conjunto de secuencias que se pueden obtener de realizar un intercambio o movimiento de alguno de los trabajos secuenciados. (Pérez, P. et al., 2021)

De esta forma, se van a tratar tres tipos de vecindades para combinarlas con cada una de las heurísticas: *Adjacent Swap*, *General Swap*, *Insertion*.

- *Adjacent Swap*: esta vecindad consiste en obtener todas las secuencias resultantes de intercambiar las posiciones de cada trabajo con cada uno de los trabajos adyacentes a estos, de forma que, para una secuencia de n trabajos, se obtendrán n – 1 secuencias vecinas.

  Por ejemplo, si partieramos de una secuencia inicial S_0 = [1, 2, 3, 4], las tres secuencias vecinas obtenidas serían: [2, 1, 3, 4], [1, 3, 2, 4] y [1, 2, 4, 3].

- *General Swap*: en este caso se procede de la misma forma que en *Adjacent Swap*, con la salvedad de que los intercambios no se realizan solo con los trabajos adyacentes, si no con todos los demás. Así, de una secuencia de n trabajos, se obtendrán $\frac{n(n-1)}{2}$ secuencias vecinas.

  Si se parte de una secuencia inicial S_0 = [1, 2, 3, 4], las seis secuencias vecinas obtenidas serán las que se enumeran a continuación: [2, 1, 3, 4], [3, 2, 1, 4], [4, 2, 3, 1], [1, 3, 2, 4], [1, 4, 3, 2] y [1, 2, 4, 3].

- *Insertion*: a diferencia de las vecindades anteriores, en este caso no se intercambian las posiciones entre trabajos, si no que se intercala cada uno de los trabajos en el resto de las posiciones posibles, resultando en $(n – 1)^2$ secuencias vecinas para una secuencia de n trabajos.

  Partiendo de una secuencia inicial S_0 = [1, 2, 3, 4], las nueve secuencias vecinas obtenidas serán las siguientes: [2, 1, 3, 4], [2, 3, 1, 4], [2, 3, 4, 1], [1, 3, 2, 4], [1, 3, 4, 2], [3, 1, 2, 4], [1, 2, 4, 3], [4, 1, 2, 3] y [1, 4, 2, 3].

A partir de estas secuencias vecinas, se puede comenzar el proceso de búsqueda local, que consiste en evaluar las secuencias en busca de un mejor valor de la función objetivo. Para ello, tendremos dos formas de proceder:

- *Fisrt Improvement*: consiste en explorar las soluciones obtenidas hasta encontrar una mejor que la solución de partida del proceso de búsqueda. A partir de ese momento, se dejan de explorar las demás.

- *Best Improvement*: consiste en evaluar todas las secuencias obtenidas de las vecindades y quedarse con la que mejor valor de la función objetivo ofrezca.

Combinando estas dos formas de proceder con cada una de las tres vecindades descritas, se obtienen 6 búsquedas locales diferentes que serán combinadas con cada una de las 6 heurísticas desarrolladas para así obtener 36 nuevas heurísticas. En las tablas que se muestran a continuación, se describe la notación que se empleará de aquí en adelante para cada búsqueda local y para cada heurística.

Tabla 4–1. Notación búsquedas locales

	Adjacent Swap	***General Swap***	***Insertion***
First Improvement	FAS	FGS	FI
Best Improvement	BAS	BGS	BI

Para agregar la búsqueda local a las heurísticas basta con buscar los vecinos deseados de la secuencia que ofrece cada secuencia y evaluarlos para seleccionar el mejor de ellos o el primero que mejore, según la búsqueda local que se quiera incorporar.

5 ANÁLISIS DE RESULTADOS

Para resolver el problema, es necesario adoptar un criterio de asignación de los anteriormente mencionados. De esta forma, el valor de la función objetivo dependrá, en gran medida, del criterio adoptado. Por este motivo, se ha decidido evaluar cada una de las 42 heurísticas desarrolladas con ambos criterios de asignación, de forma que se pueda elegir la mejor heurística para cada criterio.

Para una mayor comodidad, se definirá la notación empleada para cada heurística según el criterio de asignación adoptado en la *Tabla 5–1. Notación heurísticas*.

Además del criterio de asignación, el número de trabajos y el número de máquinas dedicadas influirán directamente sobre la solución ofrecida por las heurísticas. Esto puede llegar a producir que una heurística que ofrezca buenos resultados para ciertos valores de los parámetros mencionados no resulte ser tan eficiente para otros valores de los parámetros. Para conocer de una forma más exacta el comportamiento de cada heurística, se evaluará cada una de ellas de forma global y, después, de una forma más concreta para cada uno de los valores que pueden tomar ambos parámetros, obteniéndose así, para cada heurística, 9 valores del indicador que se explicará posteriormente.

Por último, de nada serviría que una heurística ofrezca resultados excelentes si su tiempo de ejecución es demasiado elevado, de forma que no sea práctica su utilización. Esto lleva a que, el otro indicador que se empleará para analizar la calidad de las heurísticas esté relacionado con el tiempo de ejecución.

Tabla 5–1. Notación heurísticas

ECT						
	H1	**H2**	**H3**	**H4**	**H5**	**H6**
FAS	H1_FAS	H2_FAS	H3_FAS	H4_FAS	H5_FAS	H6_FAS
BAS	H1_BAS	H2_BAS	H3_BAS	H4_BAS	H5_BAS	H6_BAS
FGS	H1_FGS	H2_FGS	H3_FGS	H4_FGS	H5_FGS	H6_FGS
BGS	H1_BGS	H2_BGS	H3_BGS	H4_BGS	H5_BGS	H6_BGS
FI	H1_FI	H2_FI	H3_FI	H4_FI	H5_FI	H6_FI
BI	H1_BI	H2_BI	H3_BI	H4_BI	H5_BI	H6_BI
FAM						
	H1	**H2**	**H3**	**H4**	**H5**	**H6**
FAS	FAM_H1_FAS	FAM_H2_FAS	FAM_H3_FAS	FAM_H4_FAS	FAM_H5_FAS	FAM_H6_FAS
BAS	FAM_H1_BAS	FAM_H2_BAS	FAM_H3_BAS	FAM_H4_BAS	FAM_H5_BAS	FAM_H6_BAS
FGS	FAM_H1_FGS	FAM_H2_FGS	FAM_H3_FGS	FAM_H4_FGS	FAM_H5_FGS	FAM_H6_FGS
BGS	FAM_H1_BGS	FAM_H2_BGS	FAM_H3_BGS	FAM_H4_BGS	FAM_H5_BGS	FAM_H6_BGS
FI	FAM_H1_FI	FAM_H2_FI	FAM_H3_FI	FAM_H4_FI	FAM_H5_FI	FAM_H6_FI
BI	FAM_H1_BI	FAM_H2_BI	FAM_H3_BI	FAM_H4_BI	FAM_H5_BI	FAM_H6_BI

5.1 Indicadores para el análisis

Para medir la calidad de las soluciones ofrecidas por las heurísticas se empleará el *Average Relative Percentage Deviation* (ARPD) según la siguiente ecuación:

$$ARPD_h = \frac{\sum_{\forall s} RPD_{h_s}}{S}, \forall\, s = 1,\ldots,S \tag{5–1}$$

donde S es el número total de instancias y RPD_{h_s} se define como:

$$RPD_{h_s} = \frac{FO_{h_s} - FO_{min}}{FO_{min}} \cdot 100 \tag{5–2}$$

siendo FO_{h_s} el valor de la función objetivo ofrecido por la heurística h para la instancia s y FO_{min} el mínimo valor de la función objetivo obtenido para la instancia s.

Por otra parte, para medir el tiempo de ejecución medio de cada heurística, se utilizará el *Average CPU* (ACPU) que se calculará como:

$$ACPU_h = \frac{\sum_{\forall s} T_{h_s}}{S} \tag{5–3}$$

siendo T_{h_s} el tiempo (s) requerido por la heurística h para alcanzar la solución de la instancia s.

5.2 Análisis global

En primer lugar, se va a realizar un análisis de los resultados de cada heurística para todas las instancias generadas para conocer, en términos generales, cuál es la mejor heurística para cada criterio de asignación.

Para el criterio ECT, la *Figura 5-1. ACPU + ECT Global* muestra los valores del indicador ARPD obtenidos para cada heurística. Como se podía predecir, todas las variantes de la H1 son claramente mejores que el resto de las heurísticas, ya que son las que más penalizan un aumento del valor de la función objetivo. Sin embargo, si se comparan las variantes de H3 con las de H5, se observa que H3 ofrece valores de la función objetivo mucho mejores que H5. Esto indica que, en términos generales, es mejor penalizar los retrasos de los trabajos en lugar de secuenciar primero los que tengan menor tiempo de finalización, ya que esto puede originar adelantos.

Por otra parte, observando los valores de las variantes de H3 y H4, se puede ver una clara diferencia entre ambas, lo que muestra que el cálculo del indicador de la heurística en dos fases ofrece mejores resultados que el cálculo en una sola fase. Esto también se puede ver comparando las variantes de H1 y H2. Sin embargo, esta diferencia no se ve reflejada en la comparación de H5 y H6 con sus respectivas variantes debido a que, a juzgar por los resultados obtenidos, un indicador que no tiene en cuenta las fechas de entrega no es adecuado.

En la *Figura 5-2. ACPU + ECT Global* se observan los valores de ACPU para cada una de las heurísticas evaluadas aplicando el criterio de asignación ECT. Comparando los tiempos de ejecución de las heurísticas con cálculo de indicador en dos fases con los de las heurísticas con cálculo de indicador en una fase, se puede ver cómo se consigue reducir el número de operaciones a realizar, aunque ninguno de los tiempos medios de ejecución llega a un segundo. Además, con los tiempos de ejecución máximos de cada heurística mostrados en la *Tabla 5–1. Tiempos de ejecución máximos ECT,* se puede ver que, el mayor de ellos no alcanza los 3 segundos, es decir, todos los tiempos de ejecución son factibles para la resolución de un problema real.

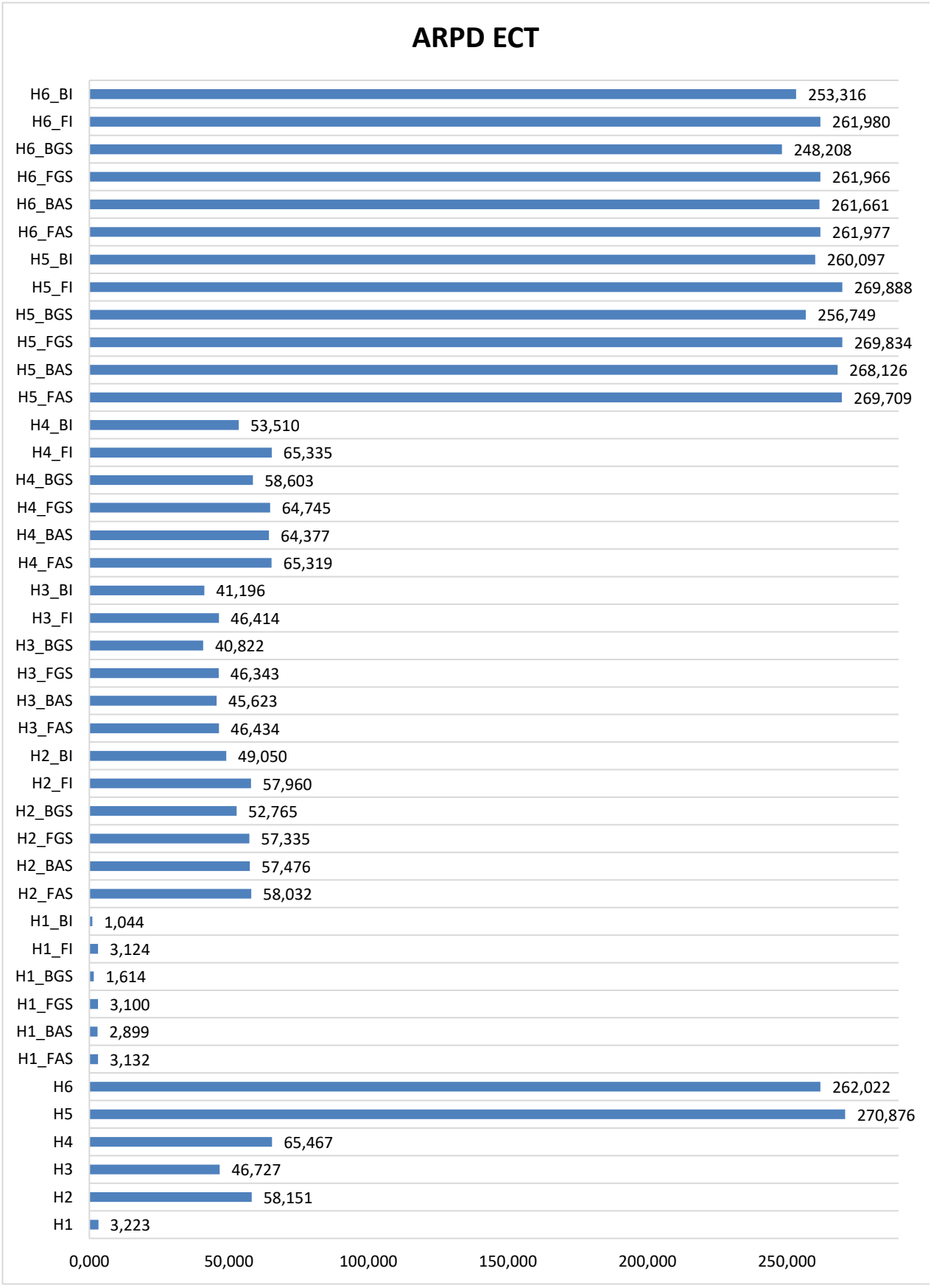

Figura 5-1. ACPU + ECT Global

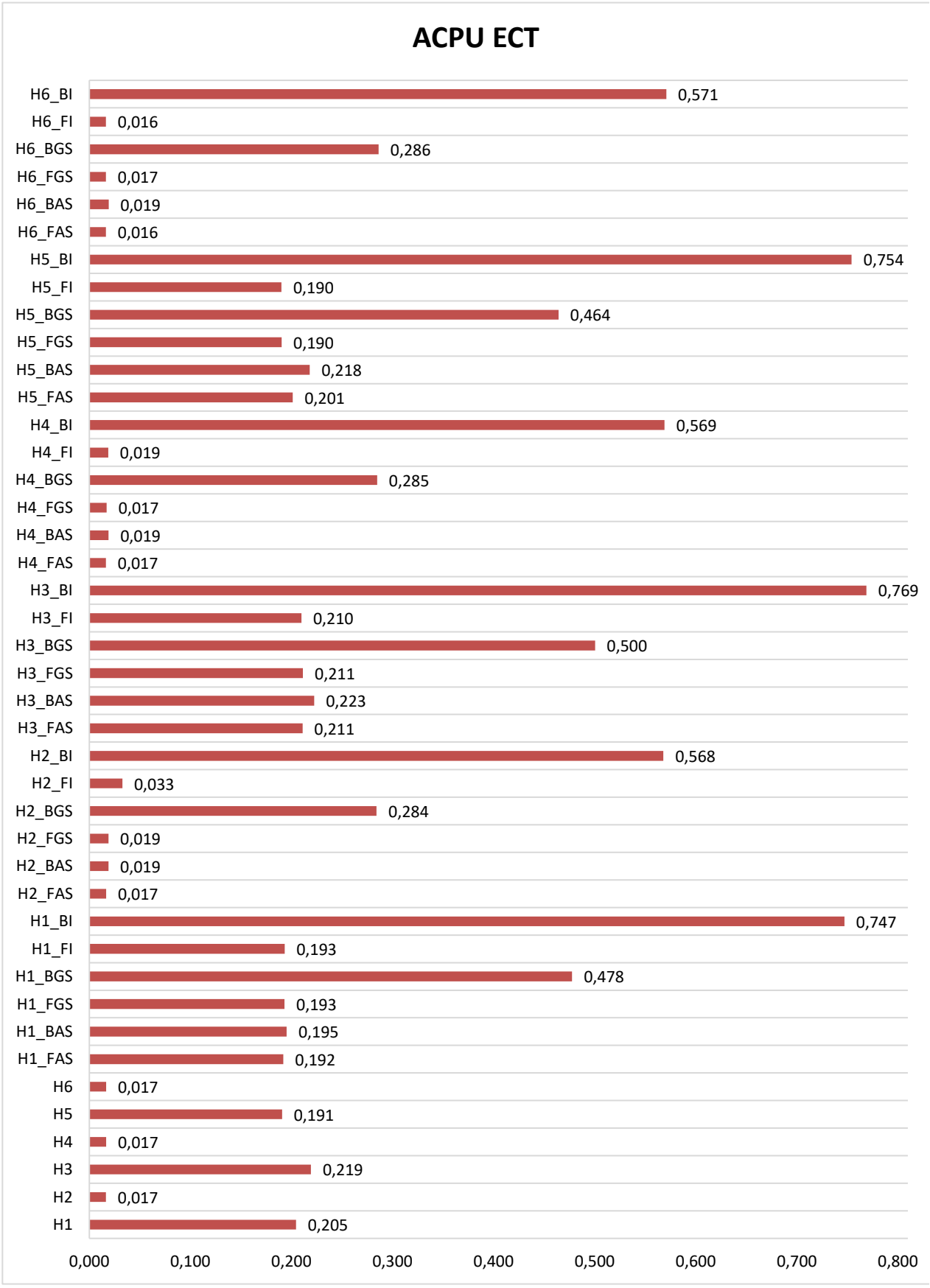

Figura 5-2. ACPU + ECT Global

Tabla 5–1. Tiempos de ejecución máximos ECT

Heurística	T_{max} (s)	Heurística	T_{max} (s)
H1	0,6838	H3_BGS	1,8580
H2	0,0461	H3_FI	0,7006
H3	0,7264	H3_BI	2,7134
H4	0,0465	H4_FAS	0,0456
H5	0,6452	H4_BAS	0,0531
H6	0,0461	H4_FGS	0,0624
H1_FAS	0,6502	H4_BGS	1,0106
H1_BAS	0,6621	H4_FI	0,1135
H1_FGS	0,6551	H4_BI	2,0148
H1_BGS	1,6814	H5_FAS	0,6720
H1_FI	0,6632	H5_BAS	0,7244
H1_BI	2,6865	H5_FGS	0,6440
H2_FAS	0,0474	H5_BGS	1,6356
H2_BAS	0,0527	H5_FI	0,6426
H2_FGS	0,0983	H5_BI	2,6712
H2_BGS	1,0049	H6_FAS	0,0455
H2_FI	0,6774	H6_BAS	0,0532
H2_BI	2,0060	H6_FGS	0,0457
H3_FAS	0,7056	H6_BGS	0,9996
H3_BAS	0,7368	H6_FI	0,0456
H3_FGS	0,7091	H6_BI	1,9952

En el caso del criterio de asignación FAM, los valores del indicador ARPD mostrados en la *Figura 5-3. ARPD + FAM Global* son similares a los obtenidos para ECT, salvo algunas excepciones. No obstante, el mejor valor obtenido es sustancialmente mayor que el obtenido según el criterio ECT, lo que deja ver que este último es mucho mejor que FAM para lograr el objetivo del problema. Además, viendo los resultados de las variantes de cada heurística y comparándolos entre ellos, se observa que los mejores resultados no dependen tanto del indicador calculado en la heurística como sí lo hace de la búsqueda local empleada. De esta forma, las mejores búsquedas locales para el criterio FAM son la BGS y la BI.

Por otra parte, si se analizan los tiempos de ejecución de cada heurística evaluada con FAM que se muestran en la *Figura 5-4. ACPU + FAM Global*, se observa que, al igual ocurría con ECT, aquellas heurísticas que calculan su indicador en una fase tienen un tiempo de ejecución menor que las que lo hacen en dos. Además, continuando con el análisis de las búsquedas locales para este criterio, si se comparan los tiempos de BGS y BI, se observa que, en todos los casos, las heurísticas que aplican

BGS son más rápidas que las que aplican BI. Esto, sumado a que también ofrecen valores del ARPD ligeramente menores que las otras, hace que se pueda decir que las mejores heurísticas para el criterio FAM sean aquellas que realizan la búsqueda local BGS.

Por último, tal y como se hizo con ECT, se muestran en la *Tabla 5–2. Tiempos de ejecución máximos FAM* los tiempos de ejecución máximos para cada heurística. En este caso, el máximo tiempo medio de ejecución vuelve a no superar los 3 segundos, lo que hace que todas las heurísticas tengas tiempos de ejecución factibles para la resolución de un problema real.

Tabla 5–2. Tiempos de ejecución máximos FAM

Heurística	**T_{max} (s)**	**Heurística**	**T_{max} (s)**
FAM_H1	0,6492	FAM_H3_BGS	1,6128
FAM_H2	0,0457	FAM_H3_FI	0,7072
FAM_H3	0,7389	FAM_H3_BI	2,5414
FAM_H4	0,0457	FAM_H4_FAS	0,0493
FAM_H5	0,6459	FAM_H4_BAS	0,0572
FAM_H6	0,0455	FAM_H4_FGS	0,0534
FAM_H1_FAS	0,6514	FAM_H4_BGS	0,9056
FAM_H1_BAS	0,6834	FAM_H4_FI	0,0703
FAM_H1_FGS	0,6579	FAM_H4_BI	1,7585
FAM_H1_BGS	1,5217	FAM_H5_FAS	0,6672
FAM_H1_FI	0,6509	FAM_H5_BAS	0,6685
FAM_H1_BI	2,3893	FAM_H5_FGS	0,7107
FAM_H2_FAS	0,0457	FAM_H5_BGS	1,5714
FAM_H2_BAS	0,0527	FAM_H5_FI	0,7330
FAM_H2_FGS	0,0513	FAM_H5_BI	2,4668
FAM_H2_BGS	0,8817	FAM_H6_FAS	0,0466
FAM_H2_FI	0,0896	FAM_H6_BAS	0,0530
FAM_H2_BI	1,7664	FAM_H6_FGS	0,0457
FAM_H3_FAS	0,7029	FAM_H6_BGS	0,8864
FAM_H3_BAS	0,7867	FAM_H6_FI	0,0461
FAM_H3_FGS	0,7748521	FAM_H6_BI	1,76447

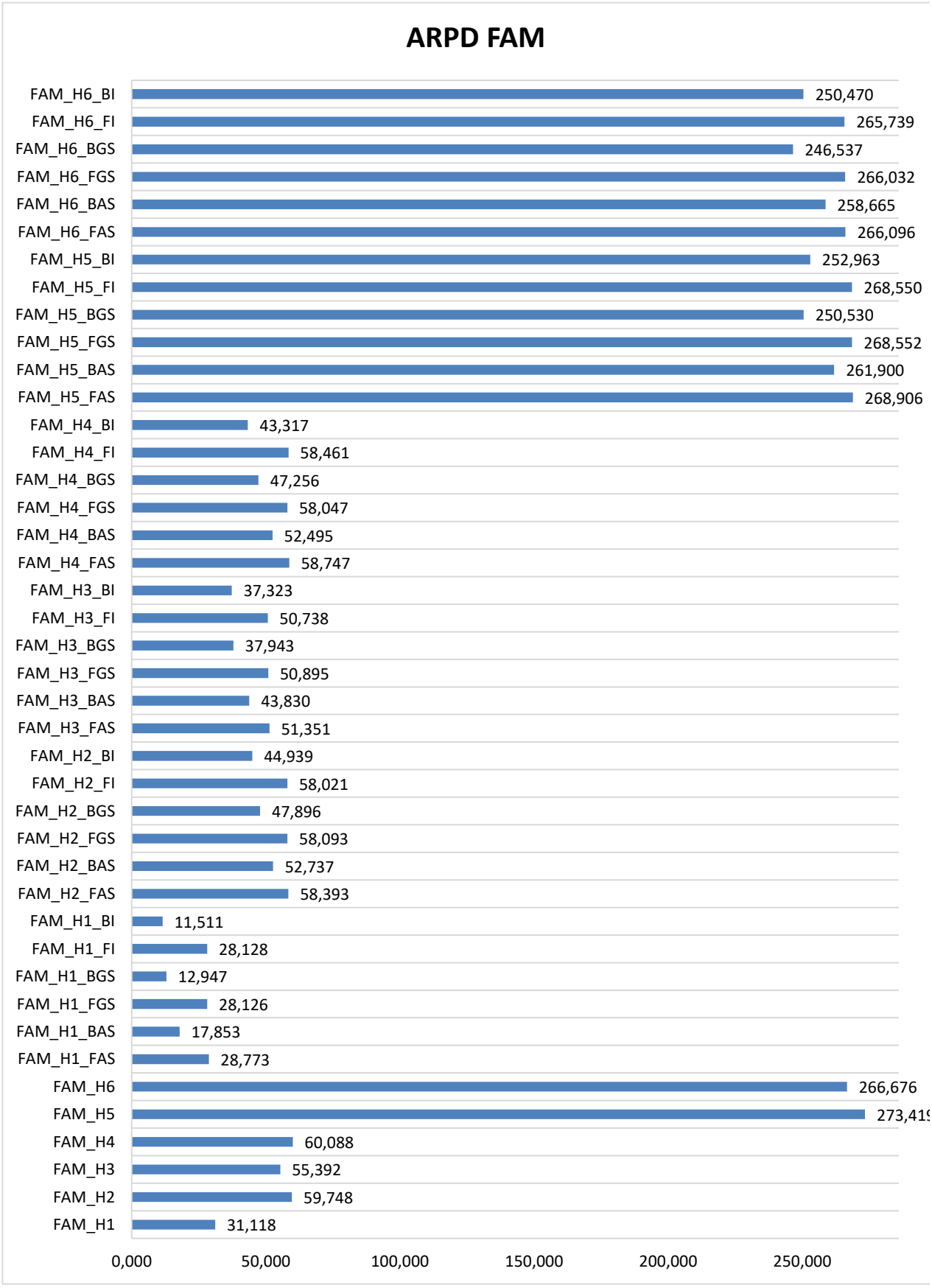

Figura 5-3. ARPD + FAM Global

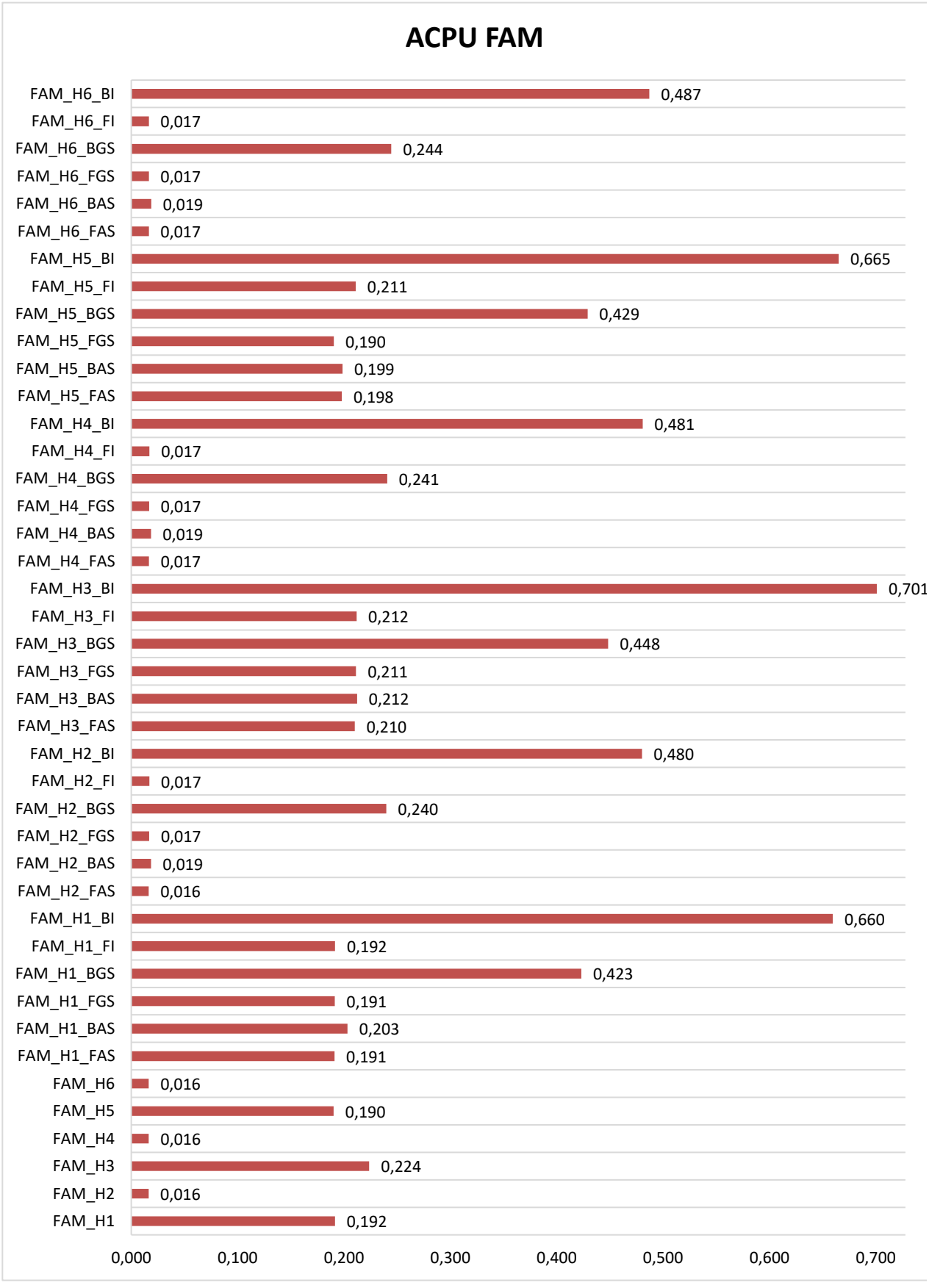

Figura 5-4. ACPU + FAM Global

En vista de los resultados obtenidos de este análisis generalizado de cada heurística, se puede concluir que, aunque el esfuerzo computacional necesario para ejecutar las heurísticas es algo a tener muy en cuenta, todos los tiempos de ejecución obtenidos son los suficientemente pequeños como para que el empleo de todas las heurísticas para resolver un problema real sea factible.

Además de esto, si se comparan los resultados del criterio ECT con los de FAM, se observa claramente que FAM es algo peor para minimizar el objetivo deseado. Esto se debe a que, al asignar cada trabajo a su máquina, no se tiene en cuenta el instante de finalización de este, si no que solo se tiene en consideración que la máquina esté en funcionamiento el mayor tiempo posible. Este criterio puede ser útil adoptarlo en entornos en los que, por ejemplo, las máquinas necesiten largos periodos de preparación antes de ponerse en funcionamiento, por lo que se intenten minimizar las paradas.

Para continuar con un análisis en mayor profundidad, se analizarán los valores obtenidos en algunos casos más concretos.

5.3 Análisis para los valores del parámetro n

A lo largo de esta sección se analizará el comportamiento de las heurísticas en función del número de trabajos de la instancia a resolver.

Para abordar el análisis de estas situaciones concretas, se ha calculado el ARPD de cada caso teniendo en cuenta únicamente aquellas heurísticas que cumplen con los requisitos necesarios del parámetro n. Estos valores se muestran en las figuras: *Figura 5-5. ARPD para n=50 + ECT, Figura 5-6. ARPD para n=100 + ECT, Figura 5-7. ARPD para n=150 + ECT, Figura 5-8. ARPD para n=200 + ECT* y *Figura 5-9. ARPD para n=250 + ECT*.

Observando los resultados de todos los escenarios, al igual que ocurría en el análisis general, las heurísticas derivadas de H1 y ella misma ofrecen siempre los mejores resultados. Además de esto, cabe destacar que, para todos los casos, si se ordenasen las heurísticas de menor a mayor valor de ARPD obtenido, el orden sería casi siempre el mismo, lo que permite ver de forma clara qué heurísticas son mejores independientemente del número de trabajos a realizar. En este caso, se puede afrimar que la mejor es H1_BI.

Para apreciar de mejor forma el comportamiento de cada heurística frente a variaciones del número de trabajos se dispone de las siguientes figuras: *Figura 5-10. ARPD H1 y derivadas para valores de n + ECT, Figura 5-11. ARPD H2 y derivadas para valores de n + ECT, Figura 5-12. ARPD H3 y derivadas para valores de n + ECT, Figura 5-13. ARPD H4 y derivadas para valores de n + ECT, Figura 5-14. ARPD H5 y derivadas para valores de n + ECT* y *Figura 5-15. ARPD H6 y derivadas para valores de n + ECT*.

De estos comportamientos se pueden sacar las siguientes conclusiones:

- Las heurísticas H1 y sus derivadas mejoran según aumenta el número de trabajos. Esto no significa que los valores de la función objetivo sean menores, si no que, comparándose con el resto de las heurísticas, están más cerca del menor valor obtenido. Es decir, que son mejores con respecto a las demás.
- Las heurísticas derivadas de H3 y ella misma, empeora con respecto a las demás conforme aumenta el número de trabajos. Sin embargo, cuanto mayor es este, la diferencia con respecto al valor anterior es menor, es decir, empeora cada vez menos.
- El comportamiento tanto de H2 y sus derivadas como de H4 y sus derivadas es difícil de predecir, ya que presentan mucos intervalos de crecimiento y decrecimiento de sus valores de

ARPD. Sin embargo, a pesar de las oscilaciones, se podría decir, en líneas generales, que H4 y sus heurísticas derivadas se acercan al mínimo cuantos más trabajos hay que procesar.

- Las heurísiticas H5, H6 y las derivadas de cada una empeoran con el aumento del número de trabajos.

De igual forma que para el criterio ECT, se va a realizar el mismo análisis para FAM. Para ello, se muestran los resultados de cada heurística para cada valor de n en las figuras: *Figura 5-16. ARPD para n=50 + FAM, Figura 5-17. ARPD para n=100 + FAM, Figura 5-18. ARPD para n=150 + FAM, Figura 5-19. ARPD para n=200 + FAM* y *Figura 5-20. ARPD para n=250 + FAM*.

Tal y como ocurre con ECT, las heurísticas H1 y sus derivadas ofrecen siempre los mejores resultados y H5, H6 y las derivadas de cada una, los peores. Sin embargo, existe una gran alternancia entre H2, H3 y H4 y todas sus derivadas, de forma que, para cada número de trabajos hay una heurística ganadora diferente. Para profundizar en esta comparación es conveniente estudiar la evolución de cada una de ellas con el aumento de n. Para ello, se muestran las siguientes figuras: *Figura 5-21. ARPD H1 y derivadas para valores de n + FAM, Figura 5-22. ARPD H2 y derivadas para valores de n + FAM, Figura 5-23. ARPD H3 y derivadas para valores de n + FAM, Figura 5-24. ARPD H4 y derivadas para valores de n + FAM, Figura 5-25. ARPD H5 y derivadas para valores de n + FAM* y *Figura 5-26. ARPD H6 y derivadas para valores de n + FAM*.

Viendo la evolución de cada una de ellas, se puede ver que, a pesar de las oscilaciones, todas ellas van empeorando conforme crece el número de trabajos. Sin embargo, al igual que sucede aplicando ECT, H3 y las derivadas de esta cada vez empeoran menos, lo que hace pensar que, a partir de cierto valor de n, estas serán las mejores heurísticas tras H1 y sus derivadas.

Por otra parte, las heurísticas derivadas de H1 y esta misma tienen un mínimo, local o absoluto, del valor de ARPD para 150 trabajos y, a partir de ahí, comienza a crecer, pudiendo incluso llegar a ser mayor que el de otras heurísticas para números de trabajos lo suficientemente altos.

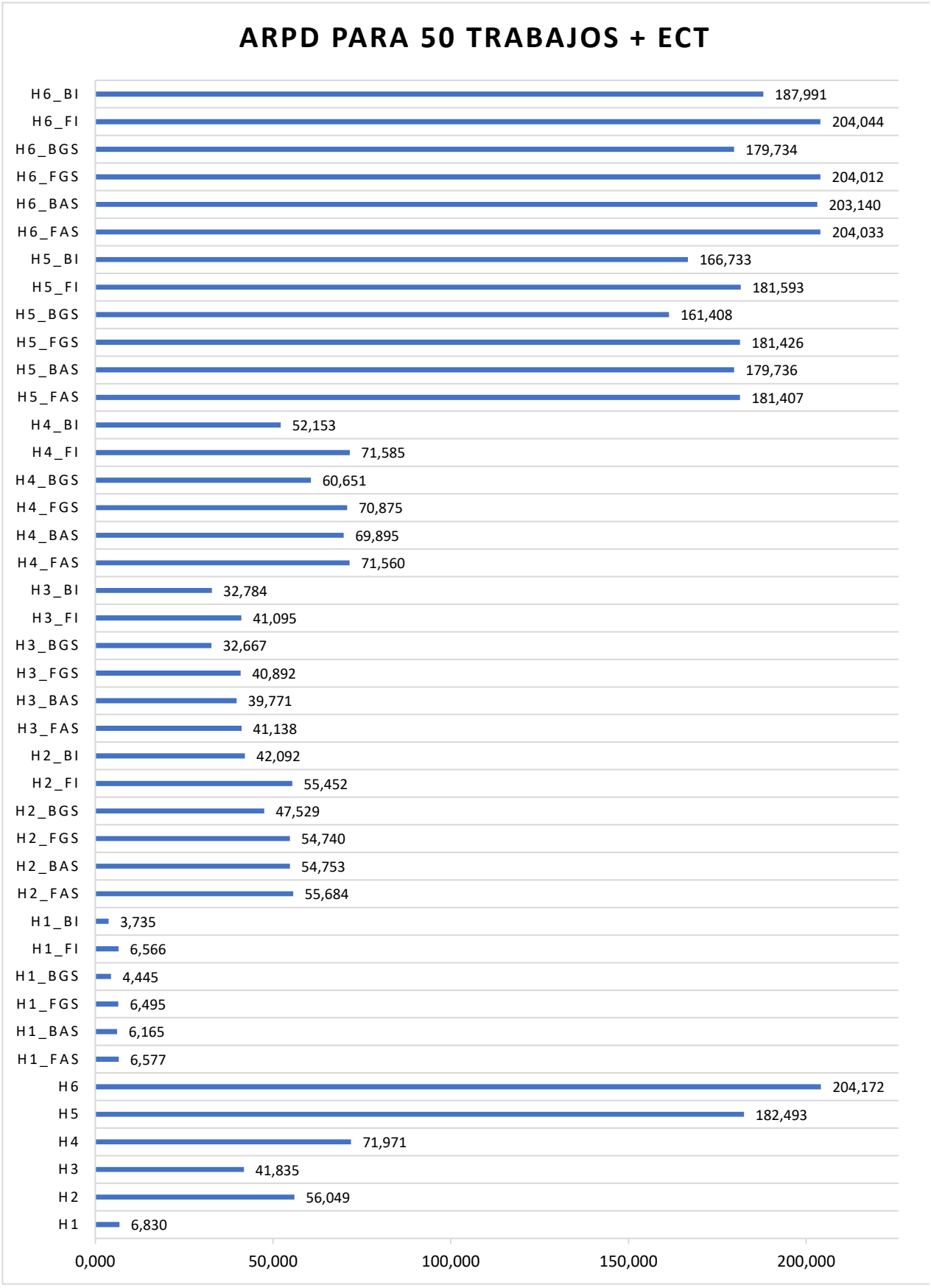

Figura 5-5. ARPD para n=50 + ECT

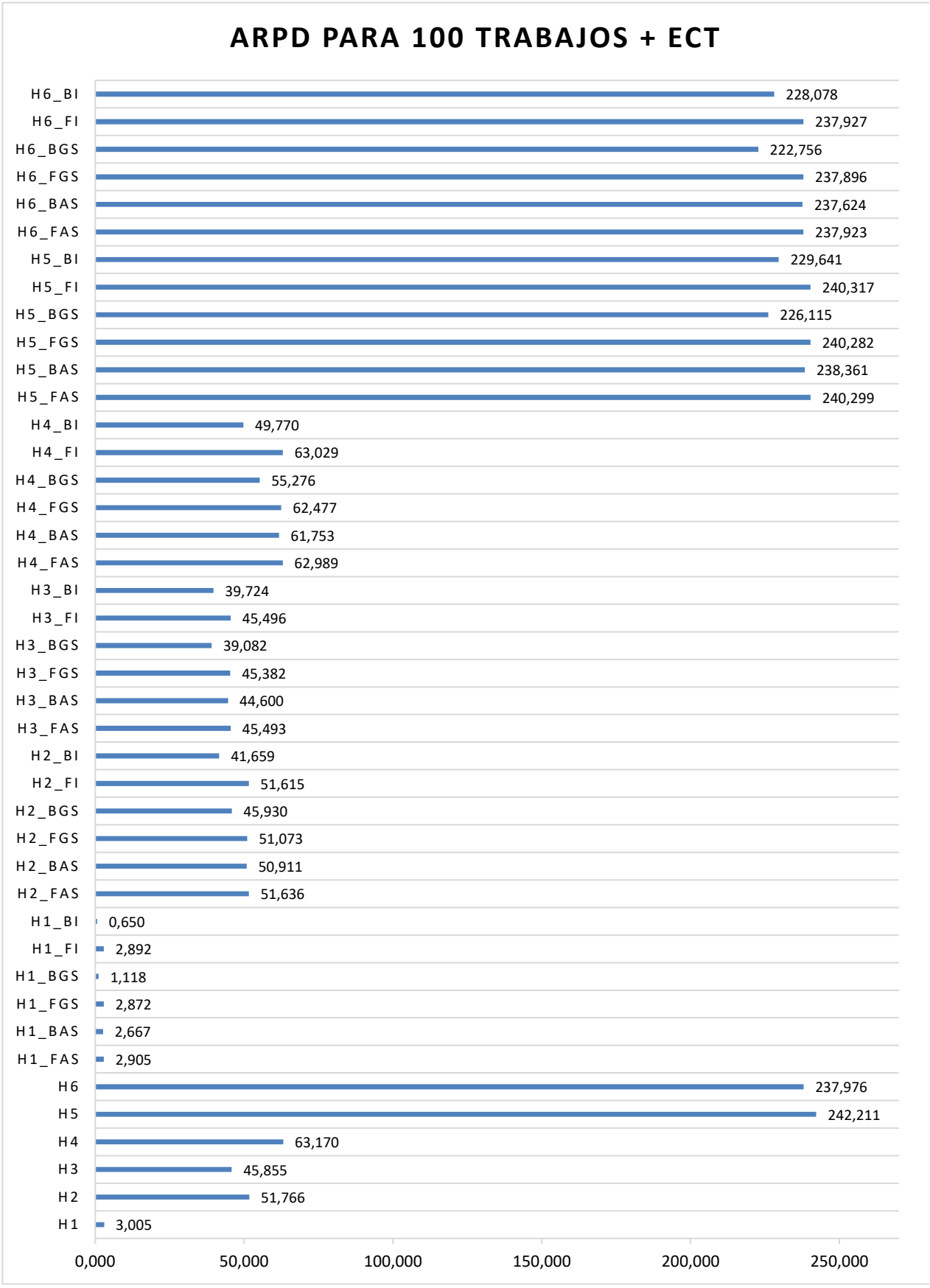

Figura 5-6. ARPD para n=100 + ECT

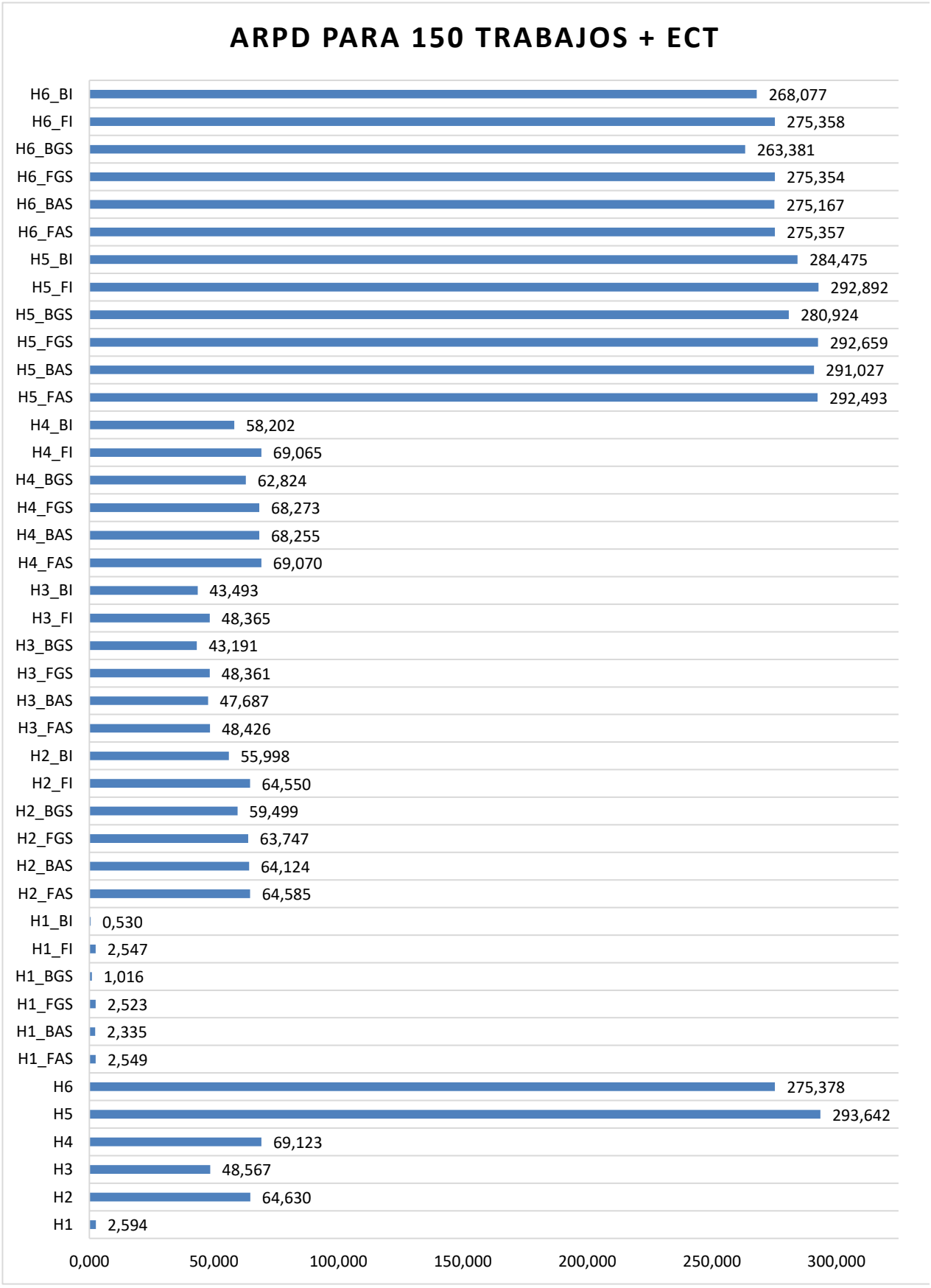

Figura 5-7. ARPD para n=150 + ECT

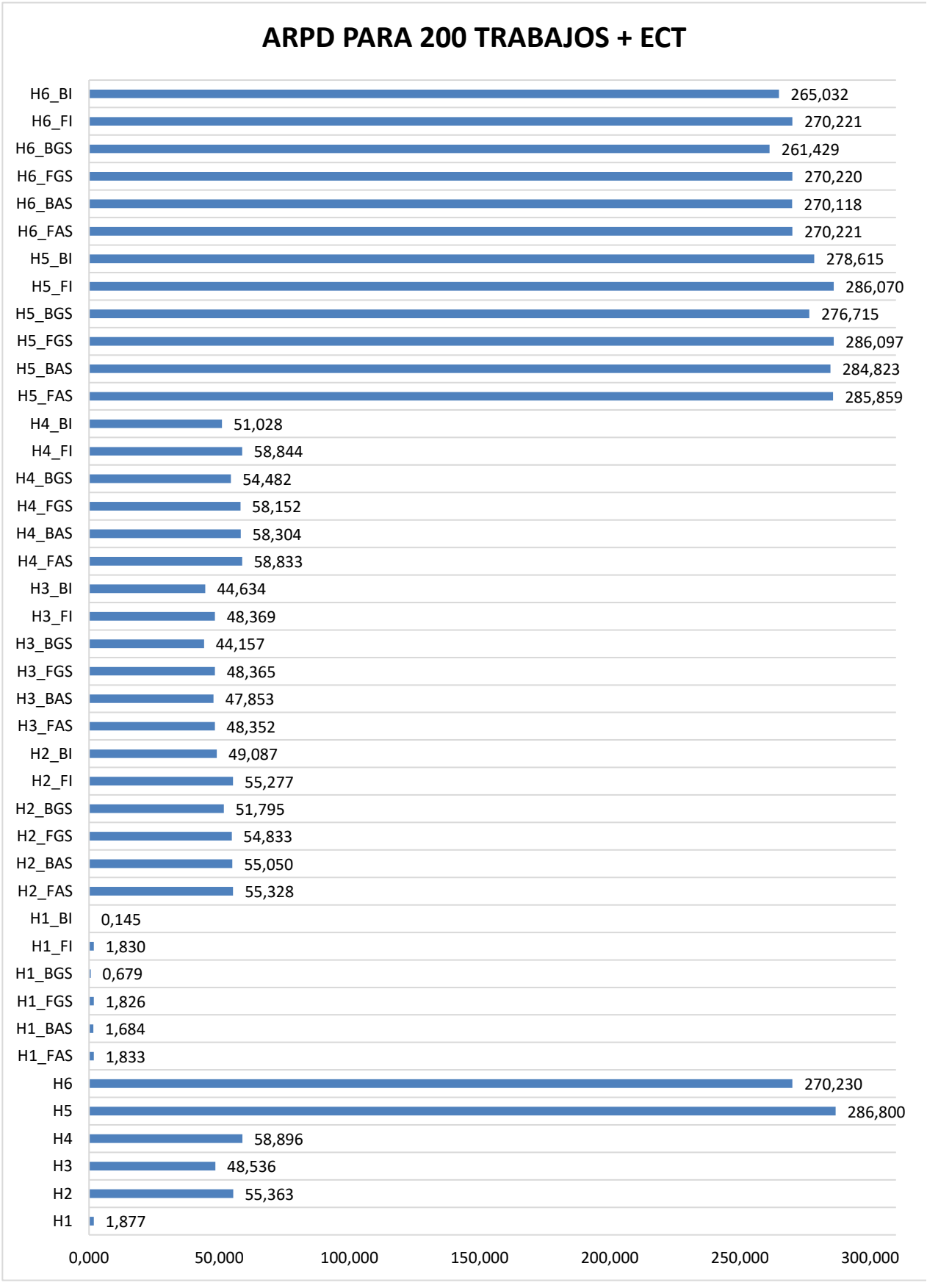

Figura 5-8. ARPD para n=200 + ECT

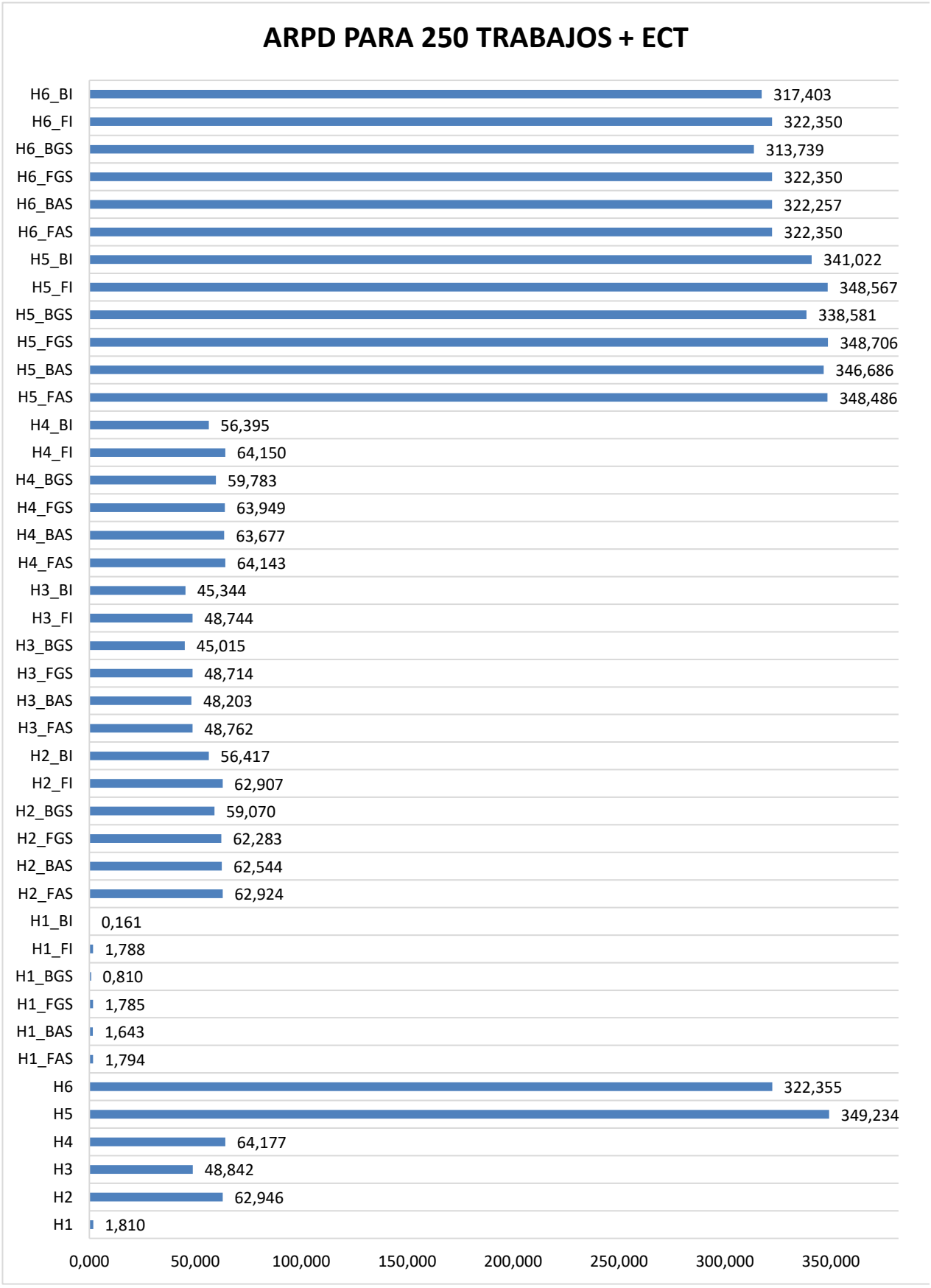

Figura 5-9. ARPD para n=250 + ECT

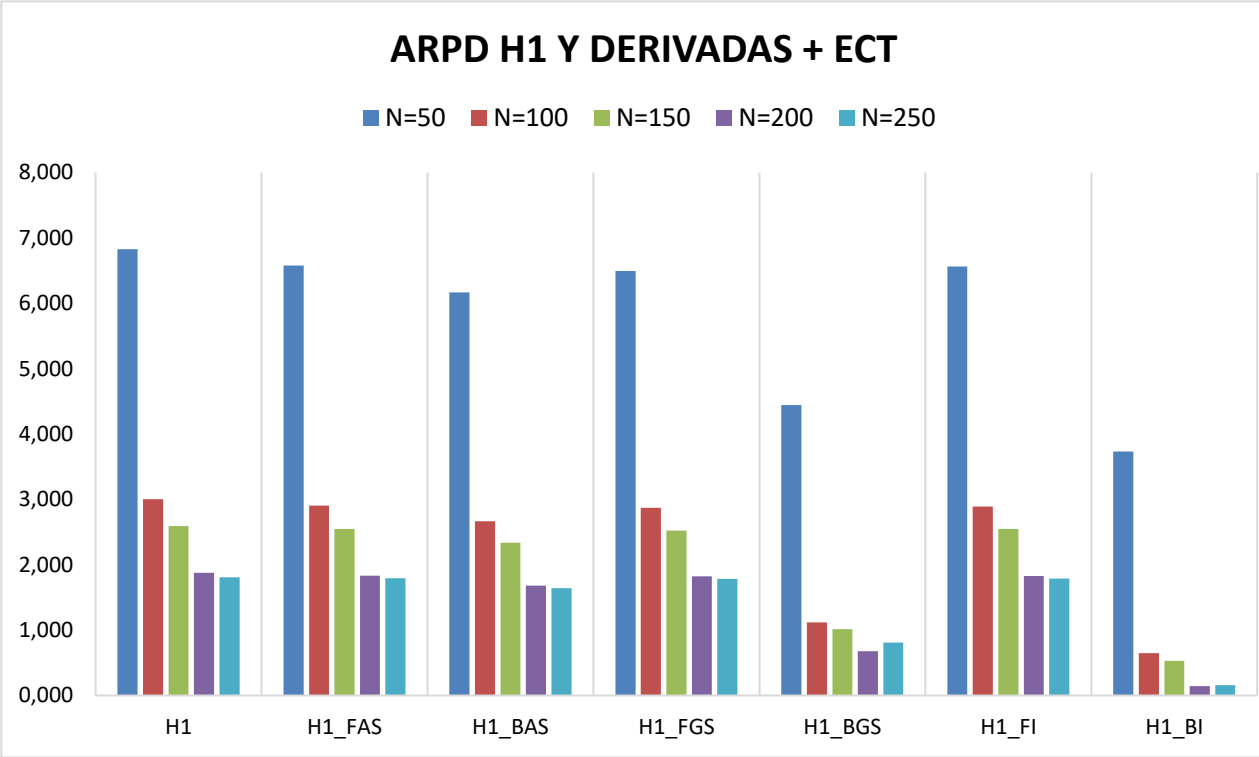

Figura 5-10. ARPD H1 y derivadas para valores de *n* + ECT

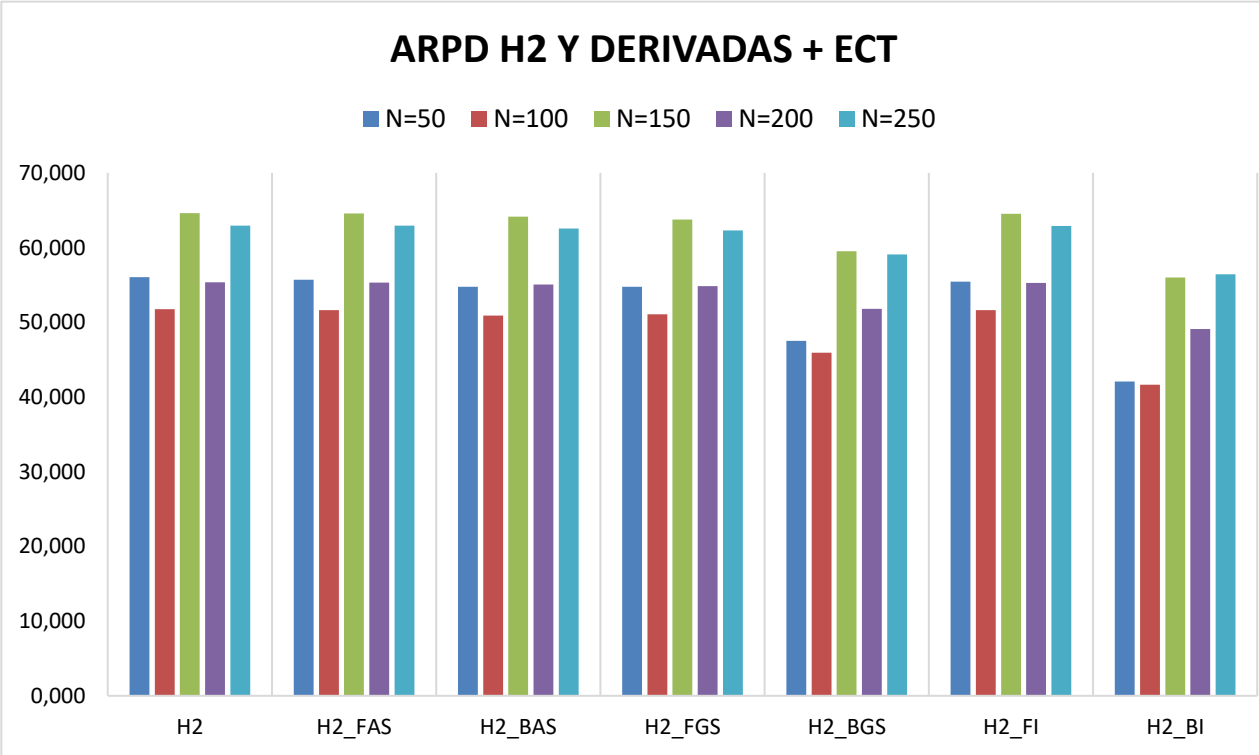

Figura 5-11. ARPD H2 y derivadas para valores de *n* + ECT

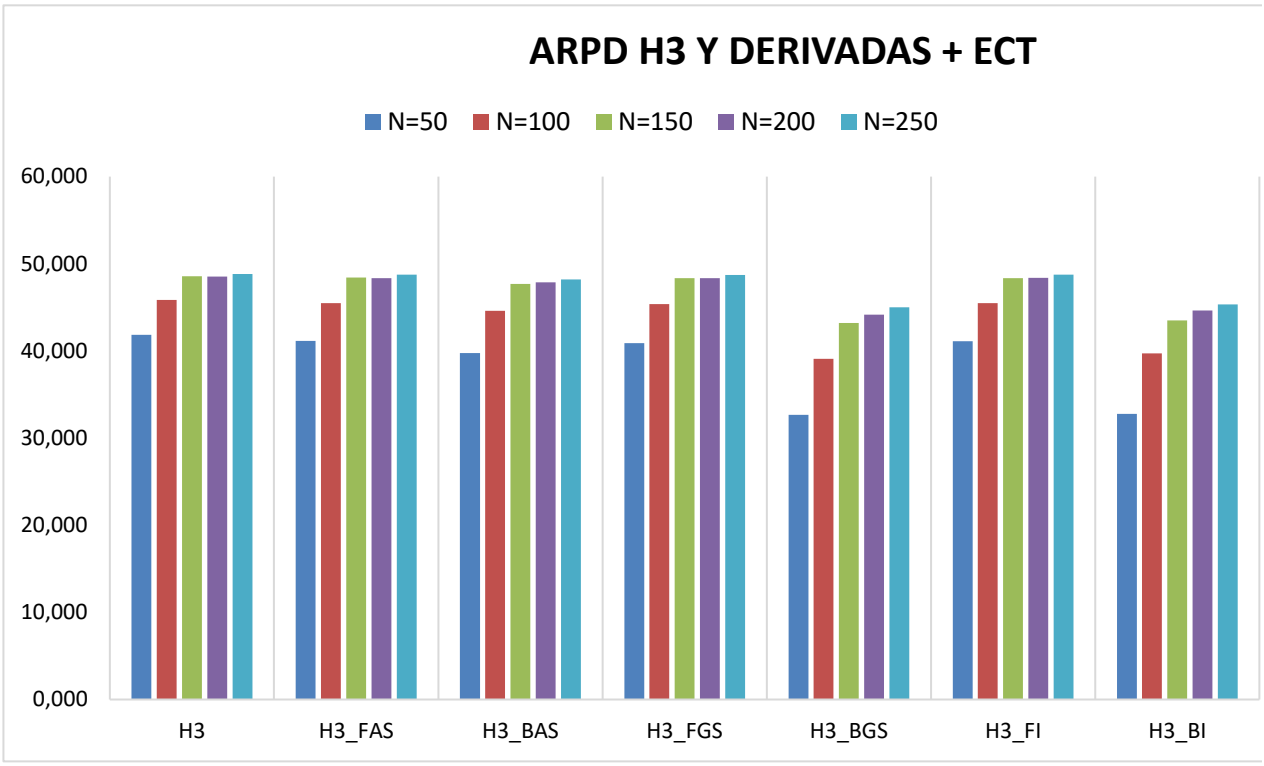

Figura 5-12. ARPD H3 y derivadas para valores de *n* + ECT

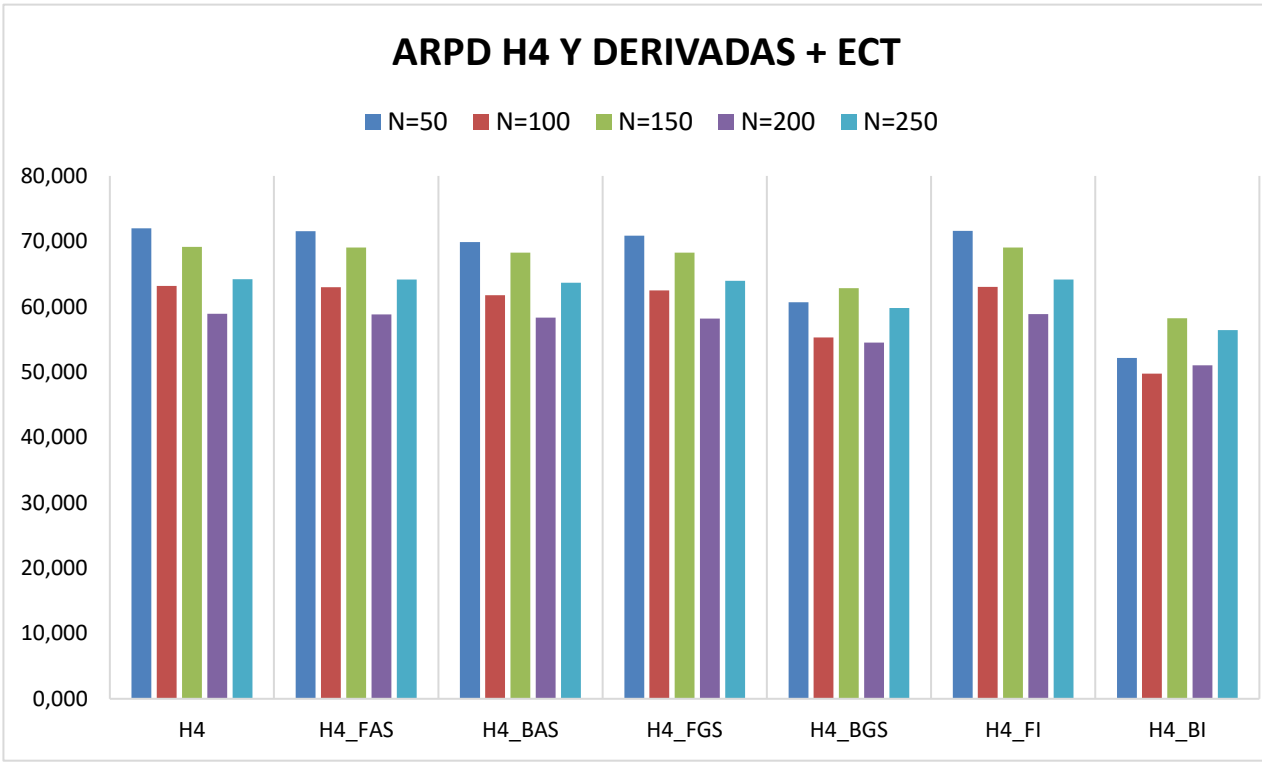

Figura 5-13. ARPD H4 y derivadas para valores de *n* + ECT

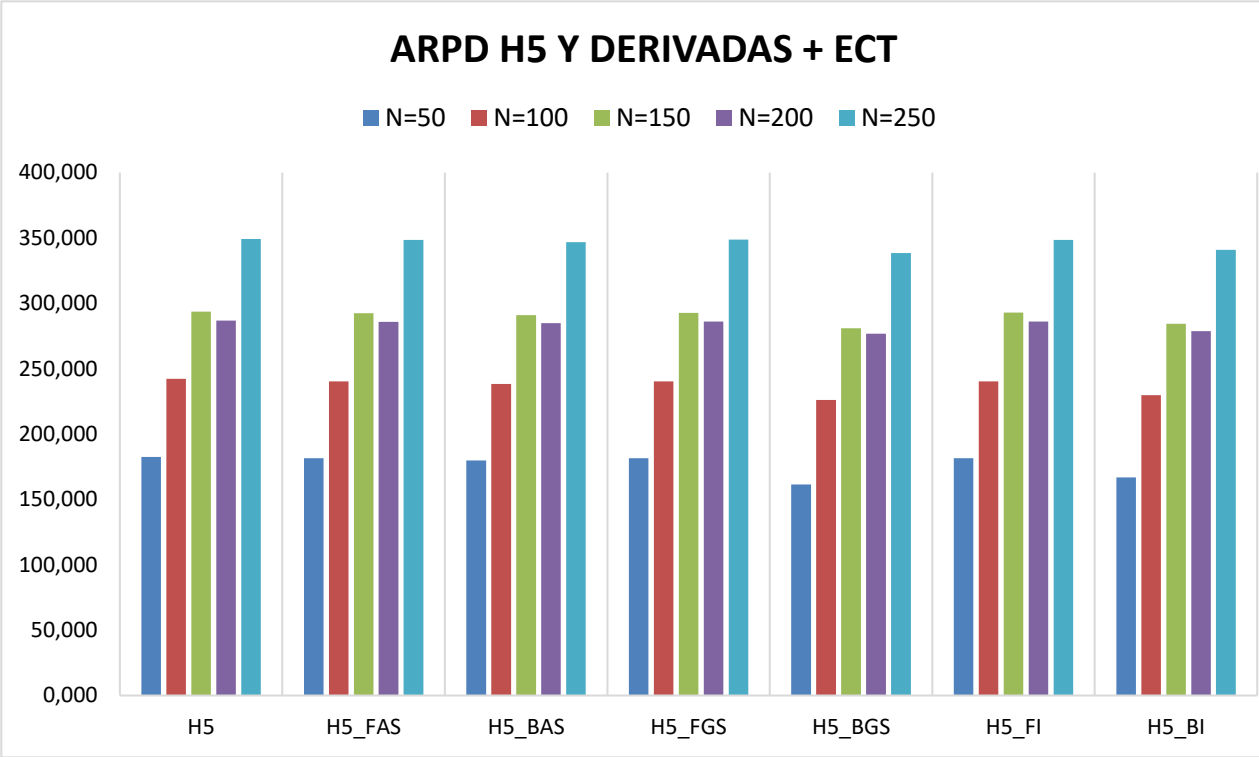

Figura 5-14. ARPD H5 y derivadas para valores de *n* + ECT

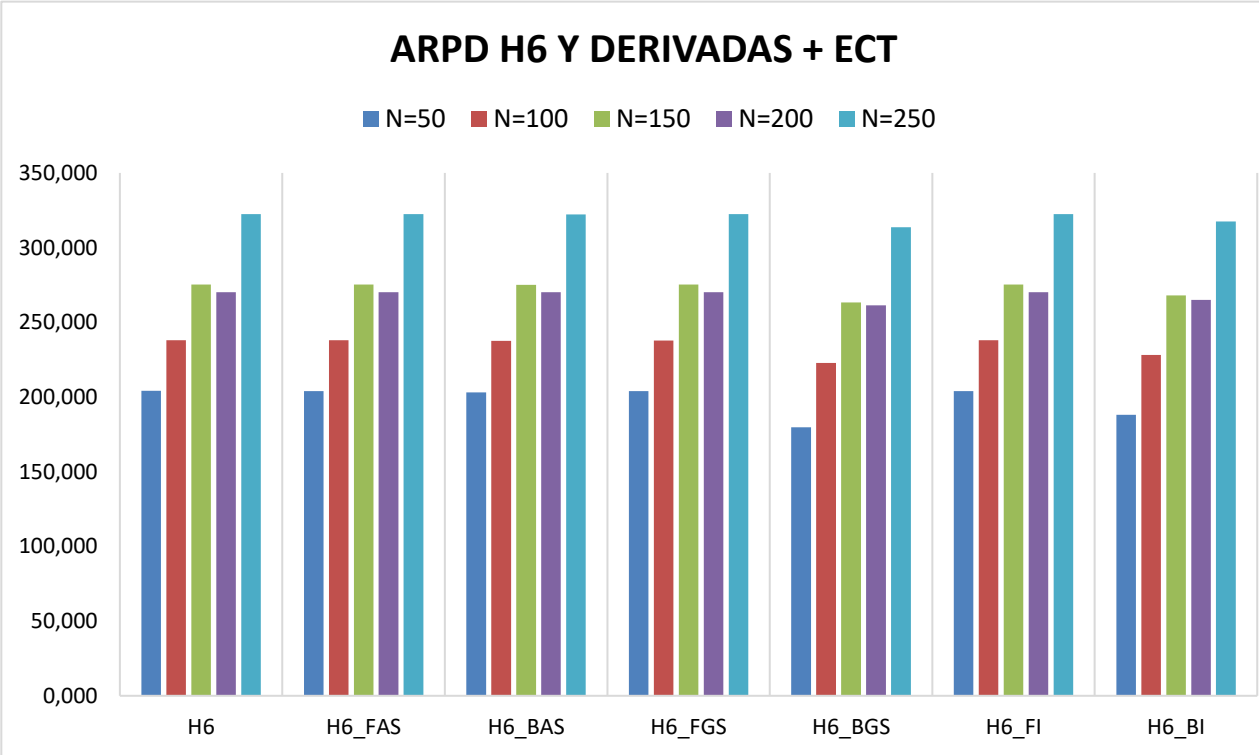

Figura 5-15. ARPD H6 y derivadas para valores de *n* + ECT

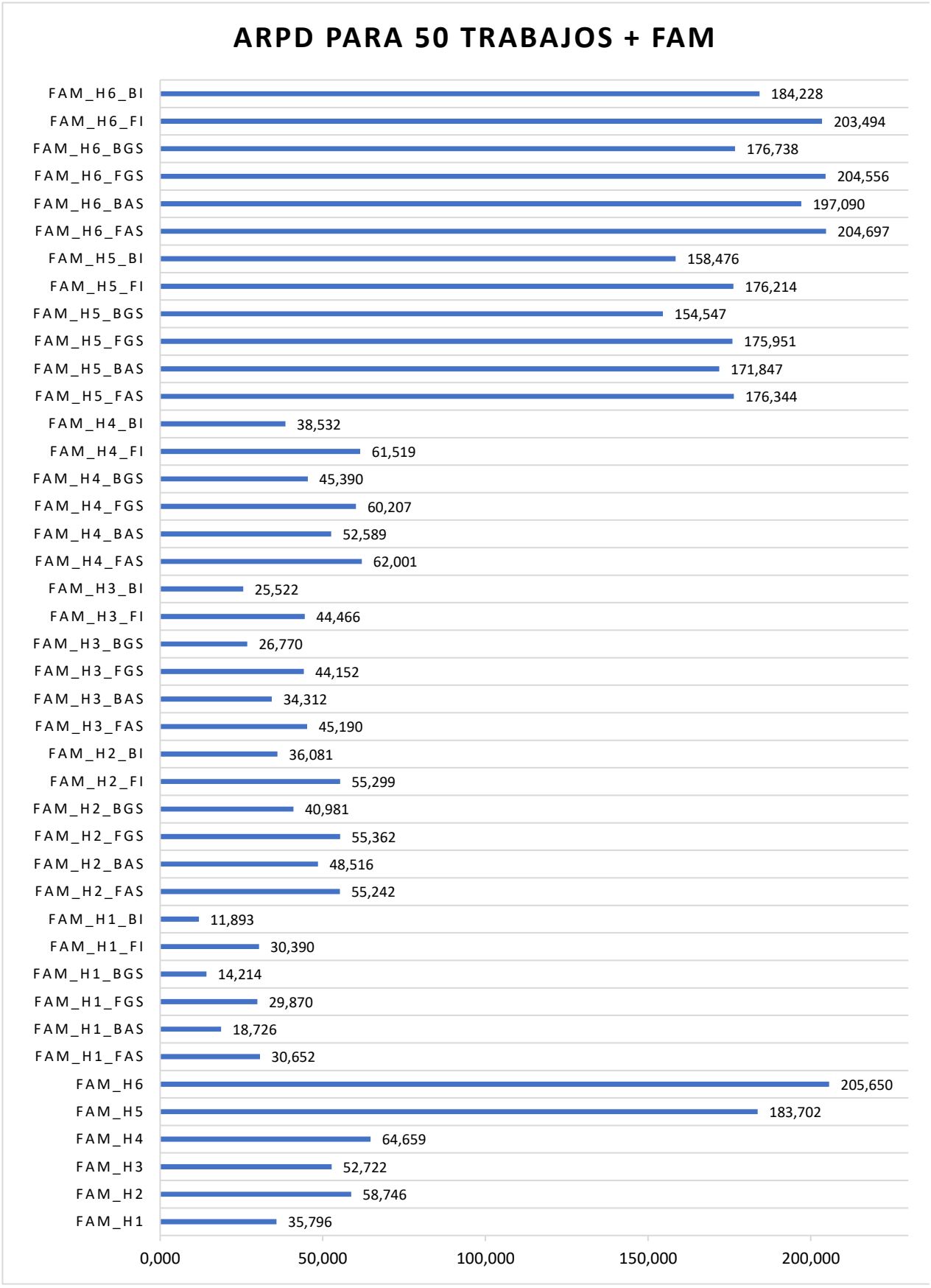

Figura 5-16. ARPD para n=50 + FAM

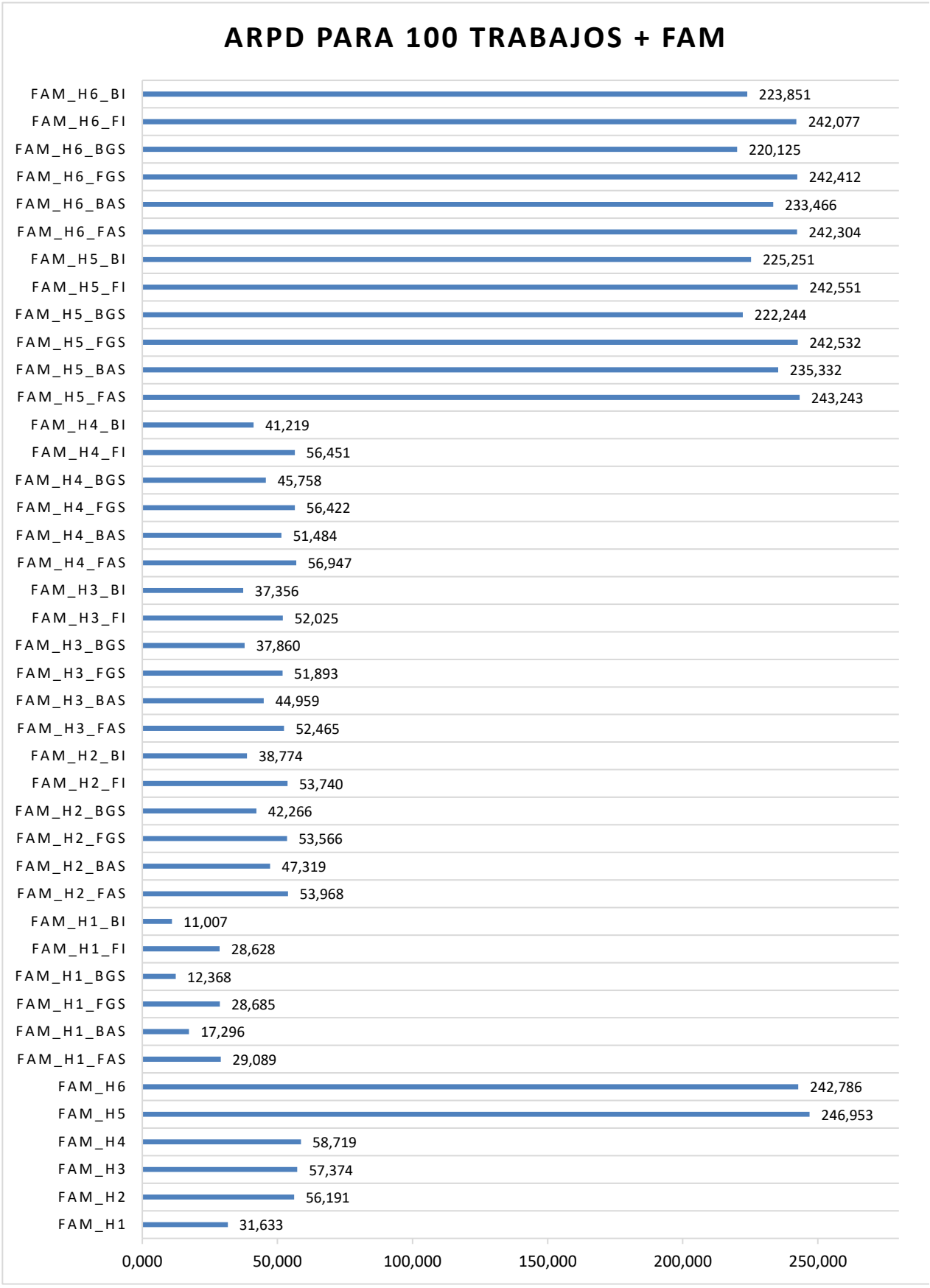

Figura 5-17. ARPD para n=100 + FAM

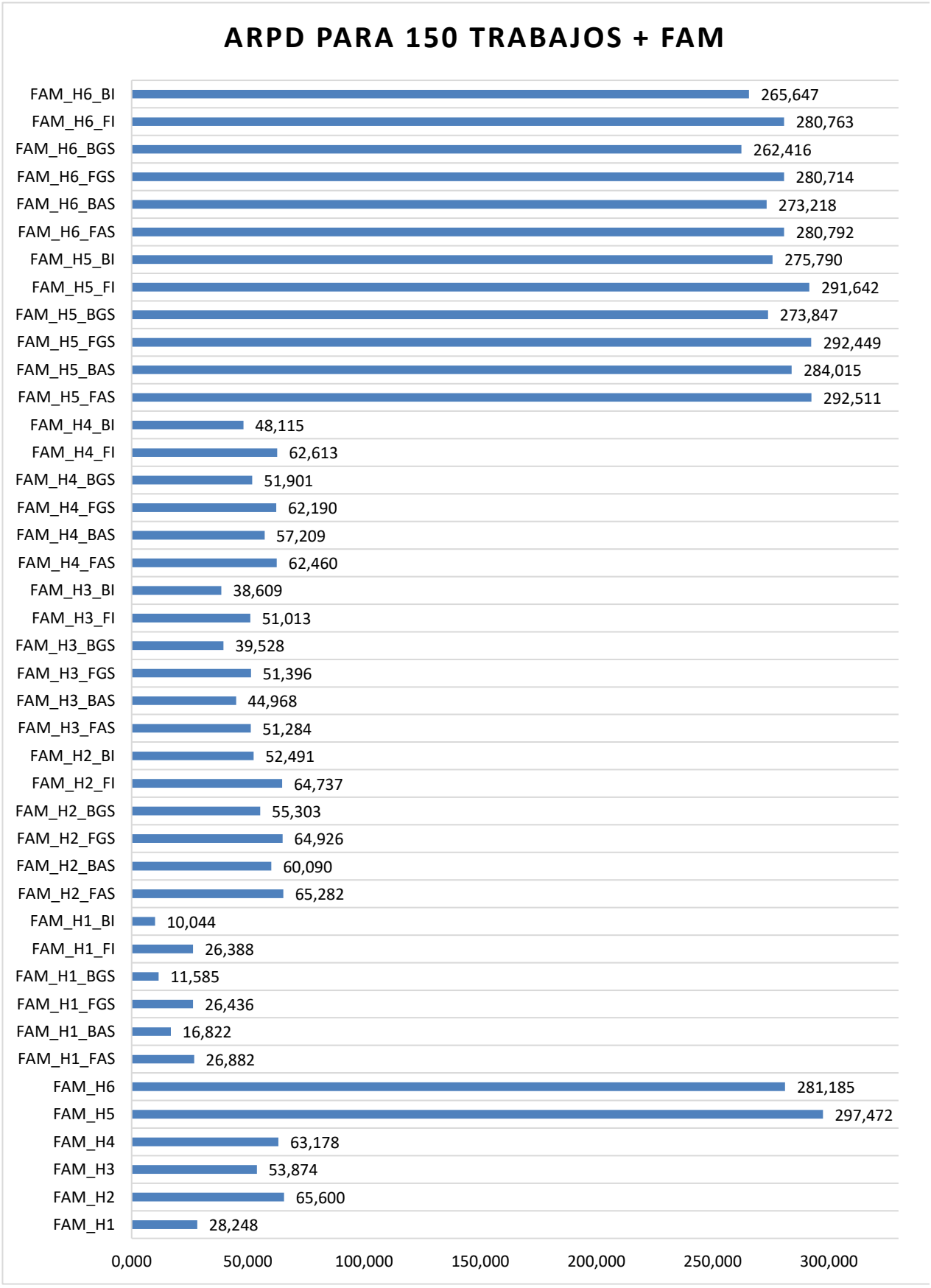

Figura 5-18. ARPD para n=150 + FAM

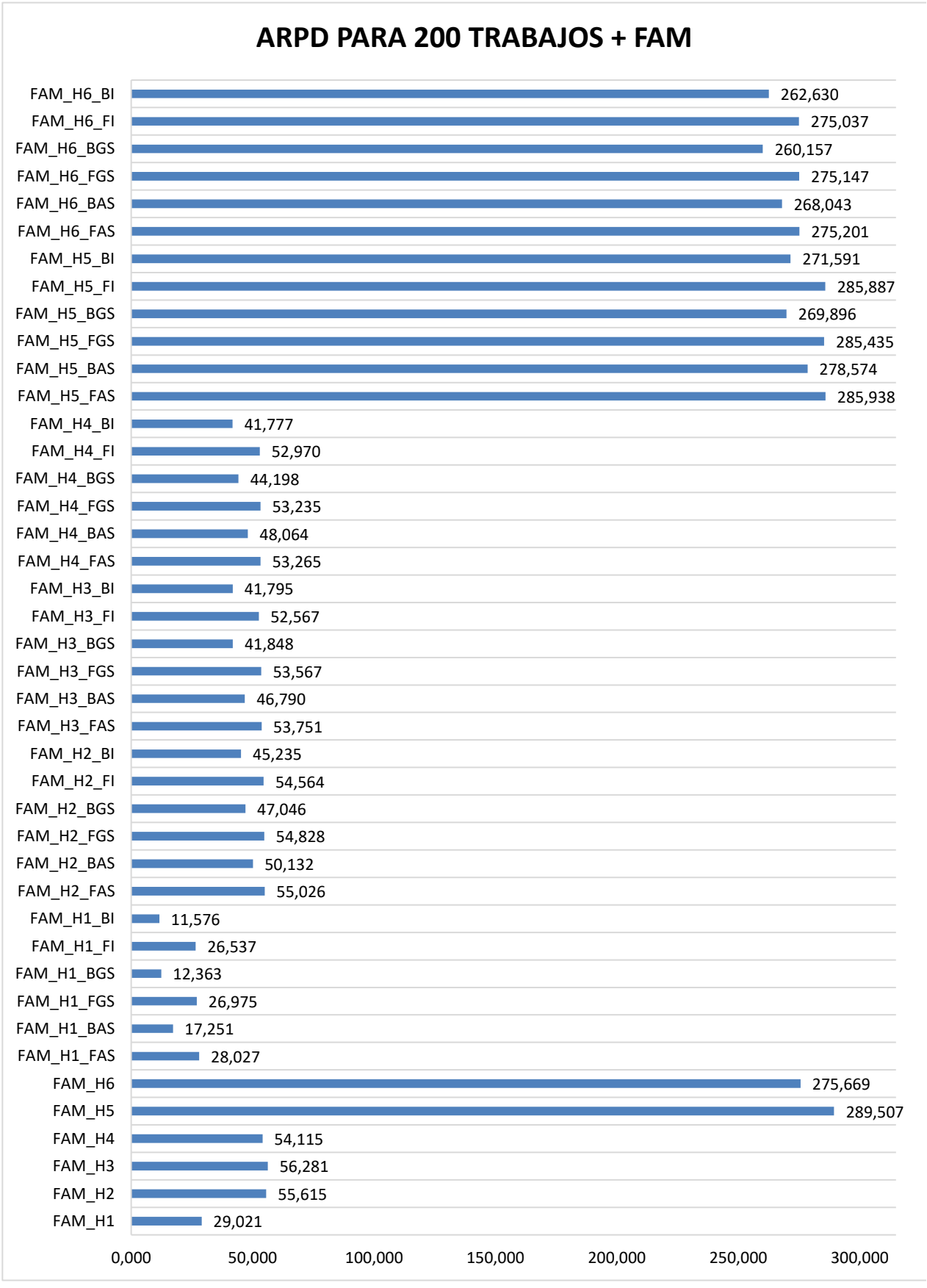

Figura 5-19. ARPD para n=200 + FAM

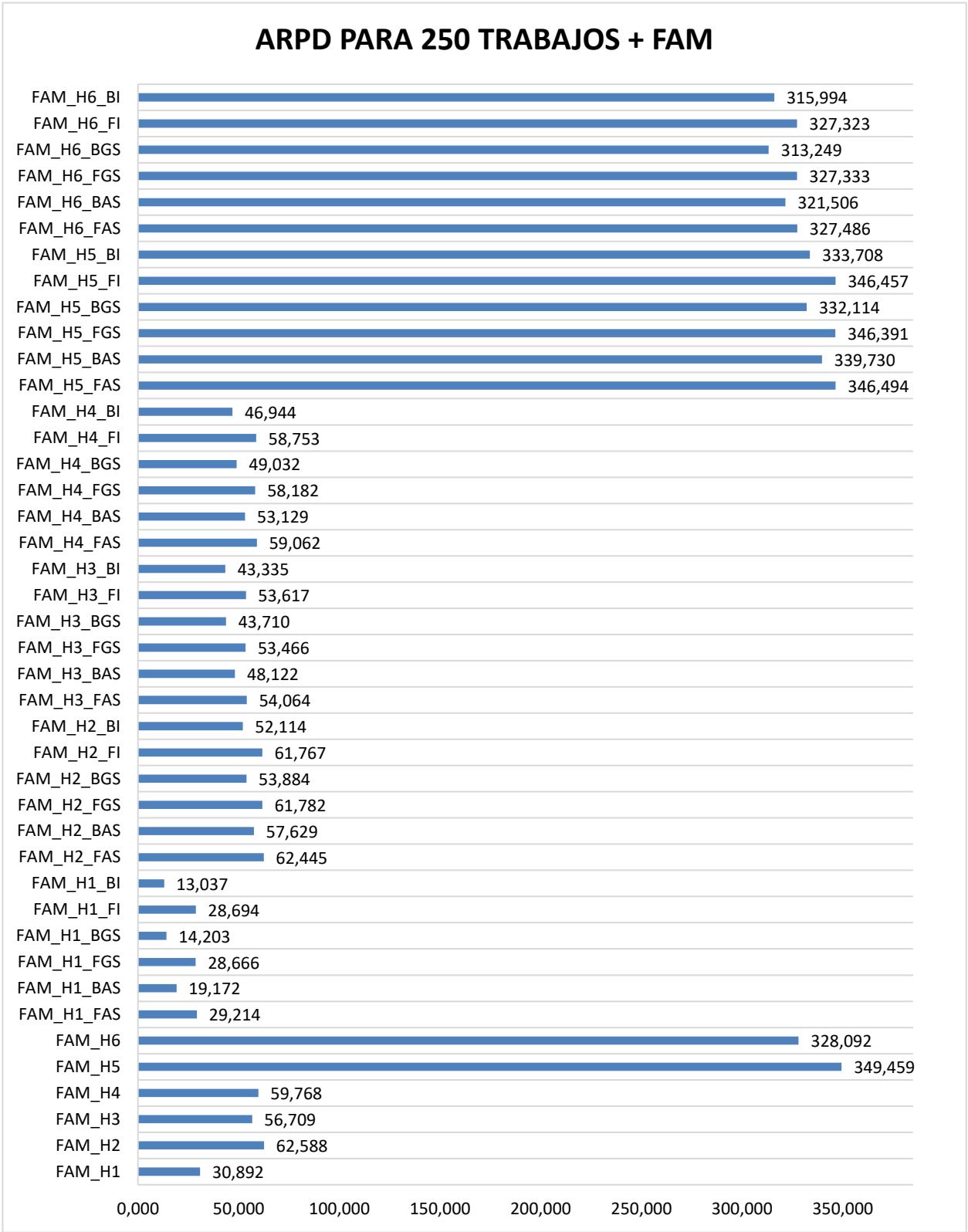

Figura 5-20. ARPD para n=250 + FAM

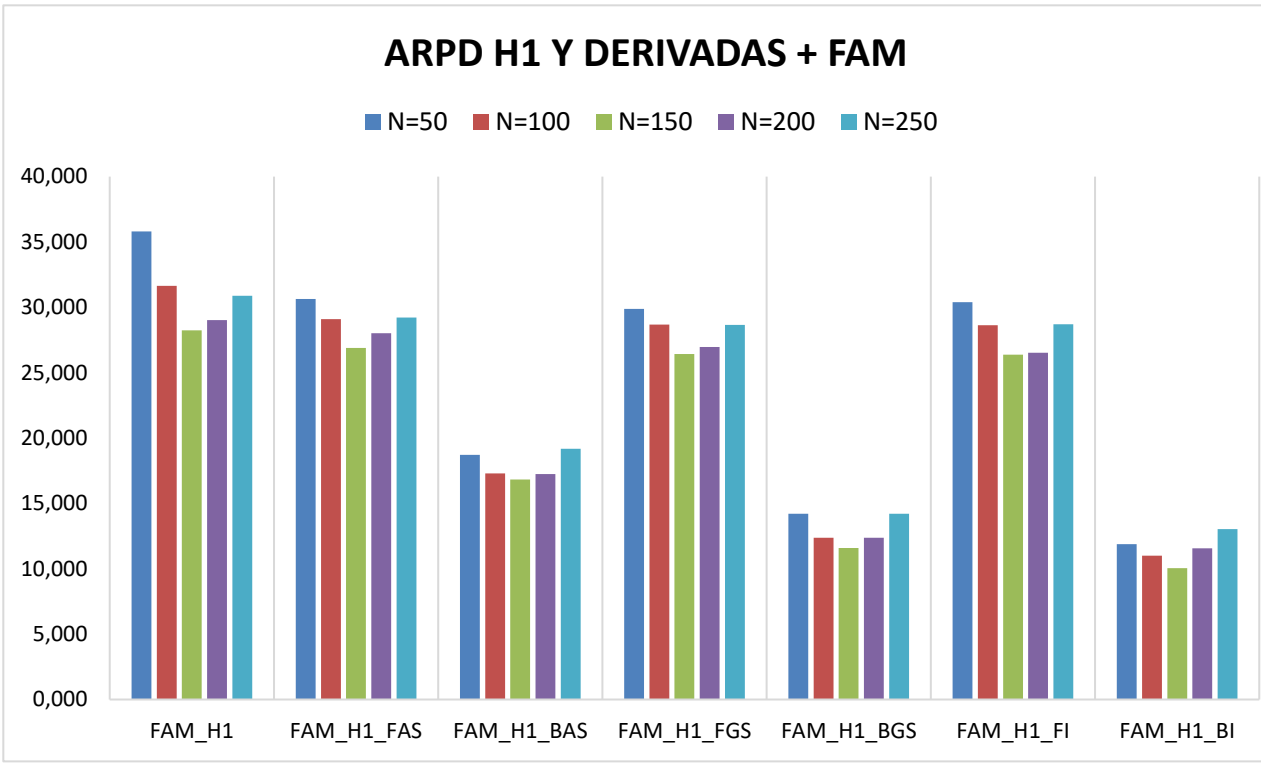

Figura 5-21. ARPD H1 y derivadas para valores de *n* + FAM

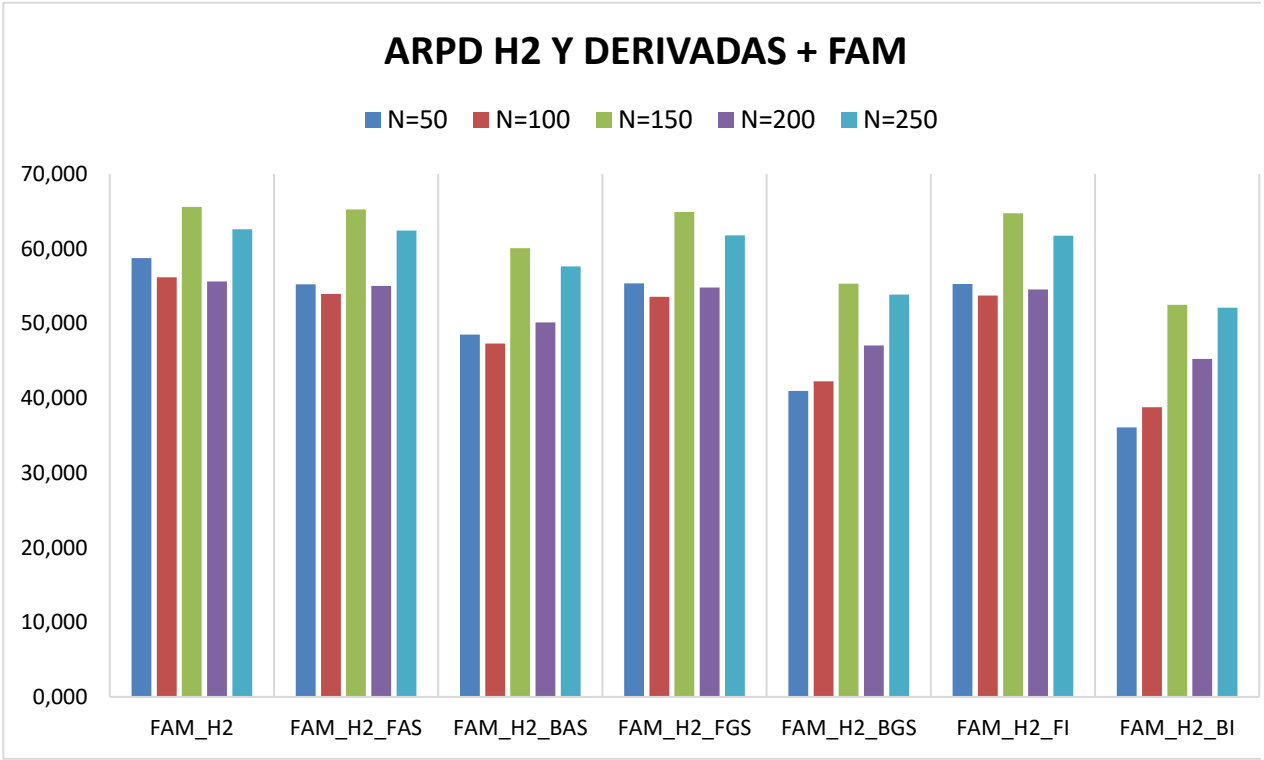

Figura 5-22. ARPD H2 y derivadas para valores de *n* + FAM

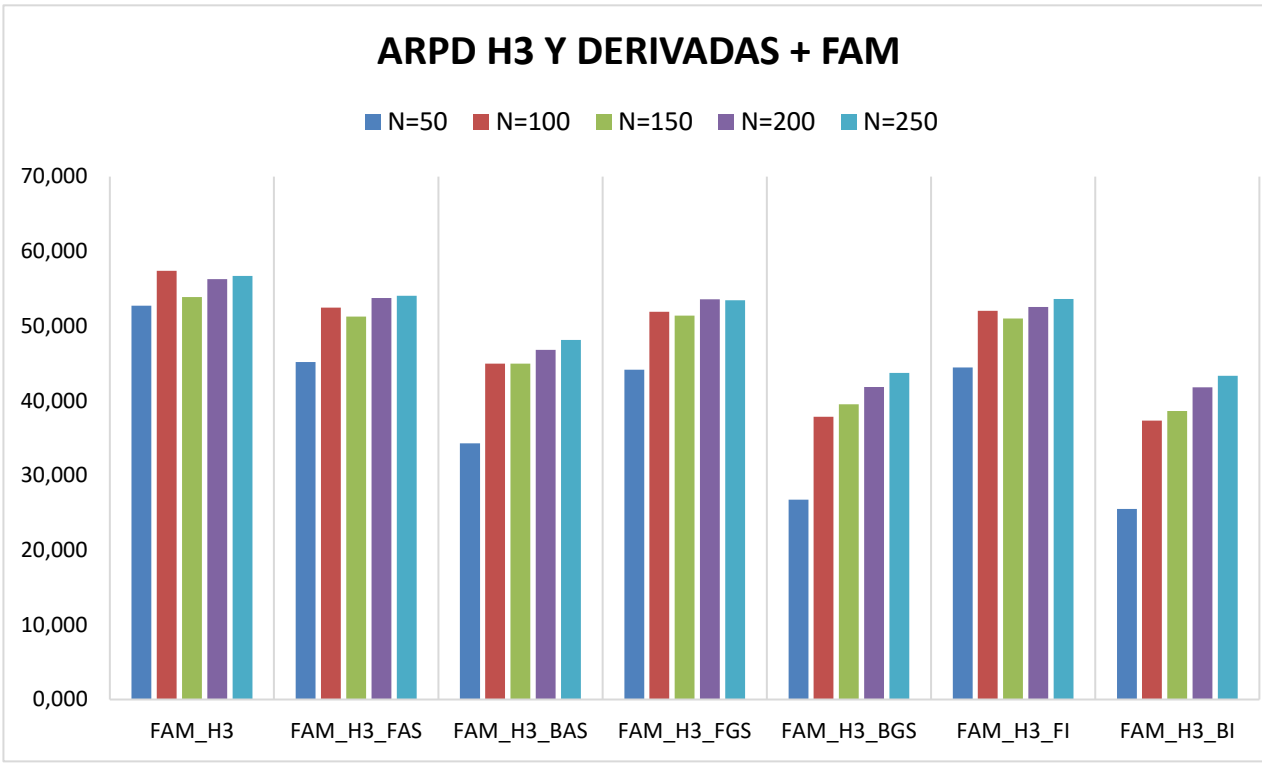

Figura 5-23. ARPD H3 y derivadas para valores de *n* + FAM

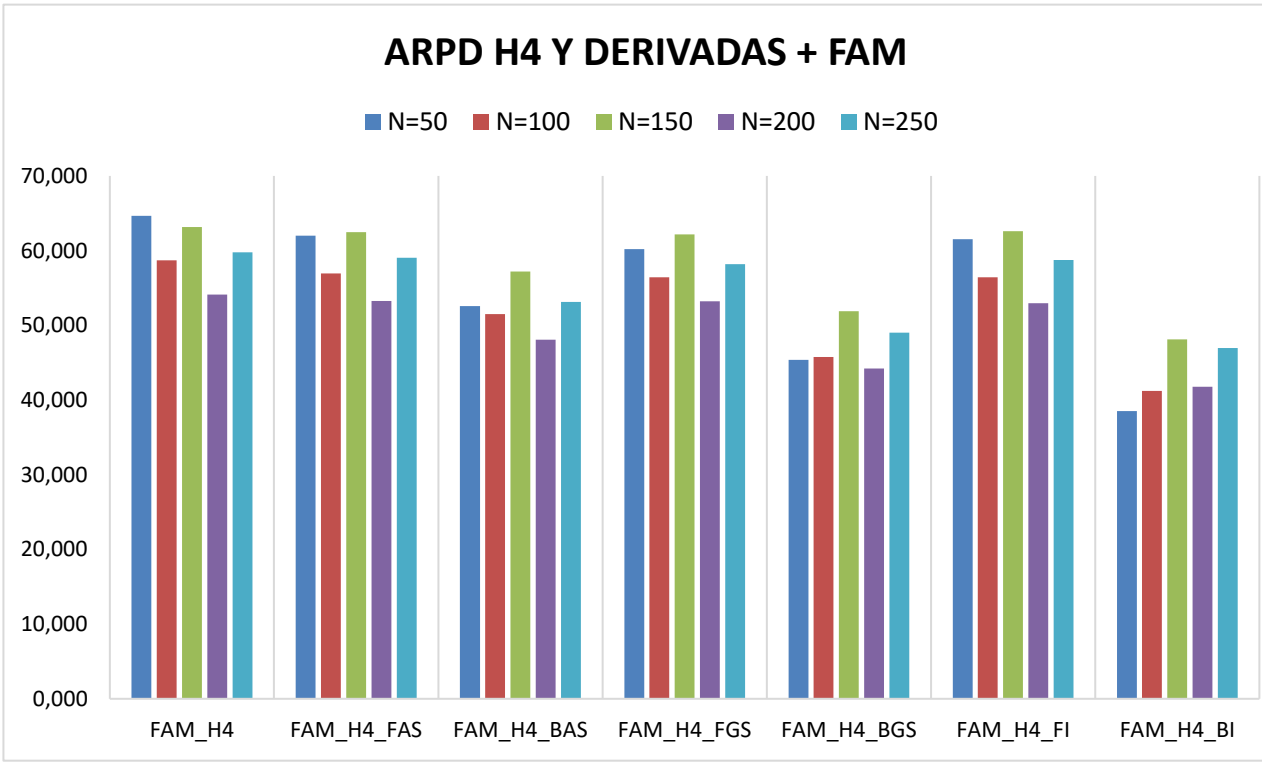

Figura 5-24. ARPD H4 y derivadas para valores de *n* + FAM

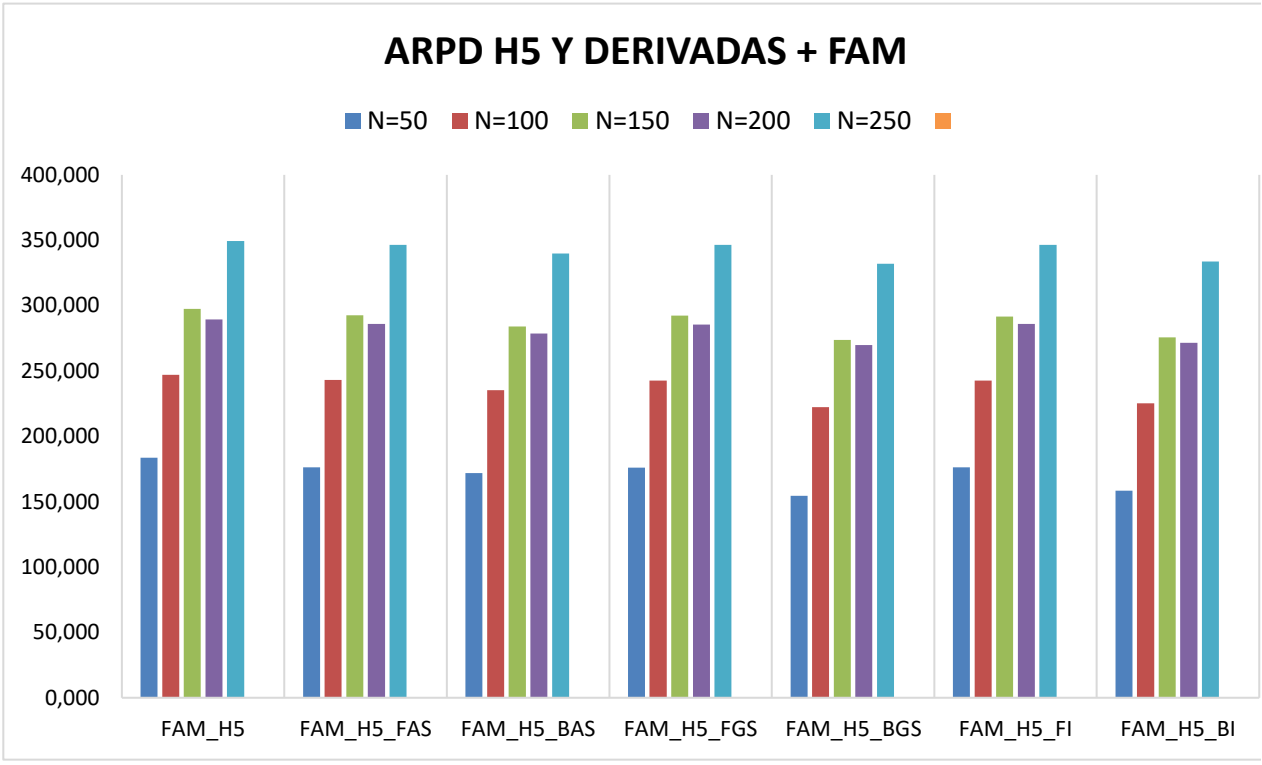

Figura 5-25. ARPD H5 y derivadas para valores de *n* + FAM

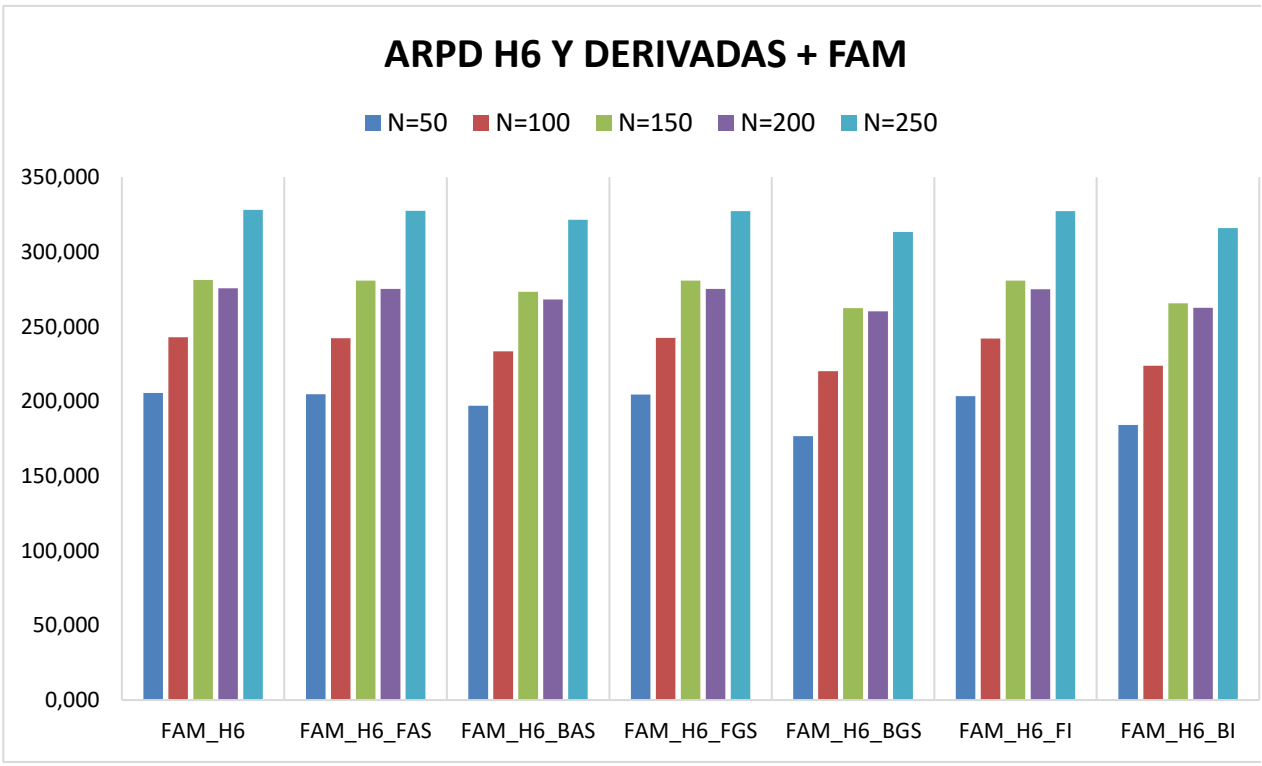

Figura 5-26. ARPD H6 y derivadas para valores de *n* + FAM

5.4 Análisis para los valores del parámetro m_{ded}

Para completar el análisis, se van a considerar todos los casos resultantes de los valores de máquinas dedicadas que se han valorado.

Para ello, se ha calculado el ARPD de cada caso teniendo en cuenta únicamente aquellas heurísticas que cumplen con los requisitos necesarios del parámetro m_{ded}. Estos valores se muestran en las figuras: *Figura 5-27. ARPD para mded = 2 + ECT, Figura 5-28. ARPD para mded = 4 + ECT* y *Figura 5-29. ARPD para mded = 6 + ECT*.

Como se podía esperar, las heurísticas H1 y sus derivadas vuelven a ser las que mejores resultados ofrecen, ya que penalizan los retrasos y adelantos en cada trabajo. De estas siete heurísticas, la que mejores valores de ARPD obtiene es, al igual que en todos los casos anteriores, H1_BI.

Por otra parte, si se tratara de intuir la evolución de los valores de la función objetivo con respecto al crecimiento del número de máquinas dedicadas, todo haría indicar que este debería disminuir, ya que cuantas más máquinas hay disponibles, más trabajos pueden procesarse a la vez. Sin embargo, en algunas ocasiones esto no es así. Puede llegar a ocurrir que, la mejora producida por un aumento del número de máquinas no sea suficiente para compensar los retrasos producidos por un aumento del número de trabajos a procesar. Además, hay que destacar que, si las máquinas que se incorporan tienen tiempos de proceso de los trabajos muy altos en comparación con las máquinas ya existentes, esto puede resultar en el retraso de los trabajos procesados en estas máquinas nuevas. El comportamiento de todas las heurísticas principales y sus derivadas se muestra en las figuras: *Figura 5-30. ARPD H1 y derivadas para valores de mded + ECT, Figura 5-31. ARPD H2 y derivadas para valores de mded + ECT, Figura 5-32. ARPD H3 y derivadas para valores de mded + ECT, Figura 5-33. ARPD H4 y derivadas para valores de mded + ECT, Figura 5-34. ARPD H5 y derivadas para valores de mded + ECT* y *Figura 5-35. ARPD H6 y derivadas para valores de mded + ECT*.

Este análisis ocurre de igual manera para el criterio FAM. Así, la heurística que mejores valores de ARPD ofrece es FAM_H1_BI. Además de esto, al igual que ocurre para distintos valores del número de trabajo, es muy difícil establecer qué heurísitica es mejor entre H2, H3, H4 y todas sus derivadas, ya que hay muchas oscilaciones. Esto se puede observar en las figuras: *Figura 5-36. ARPD para mded = 2 + FAM, Figura 5-37. ARPD para mded = 4 + FAM* y *Figura 5-38. ARPD para mded = 6 + FAM*

En cuanto a la evolución de cada heurística, se puede sacar una única conclusión. En términos generales, un aumento del número de máquinas disponibles favorecerá la asignación de trabajos a las máquinas, independientemente del tiempo de finalización. Esto, en la mayoría de los casos, afecta positivamente a la función objetivo. Sin embargo, existe la posibilidad de que, por asignar el trabajo a una máquina rápidamente, se produzca un adelanto que perjudique el objetivo del problema. El comportamiento de cada heurística se muestra en las figuras: *Figura 5-39. ARPD H1 y derivadas para valores de mded + FAM, Figura 5-40. ARPD H2 y derivadas para valores de mded + FAM, Figura 5-41. ARPD H3 y derivadas para valores de mded + FAM, Figura 5-42. ARPD H4 y derivadas para valores de mded + FAM, Figura 5-43. ARPD H5 y derivadas para valores de mded + FAM* y *Figura 5-44. ARPD H6 y derivadas para valores de mded + FAM*.

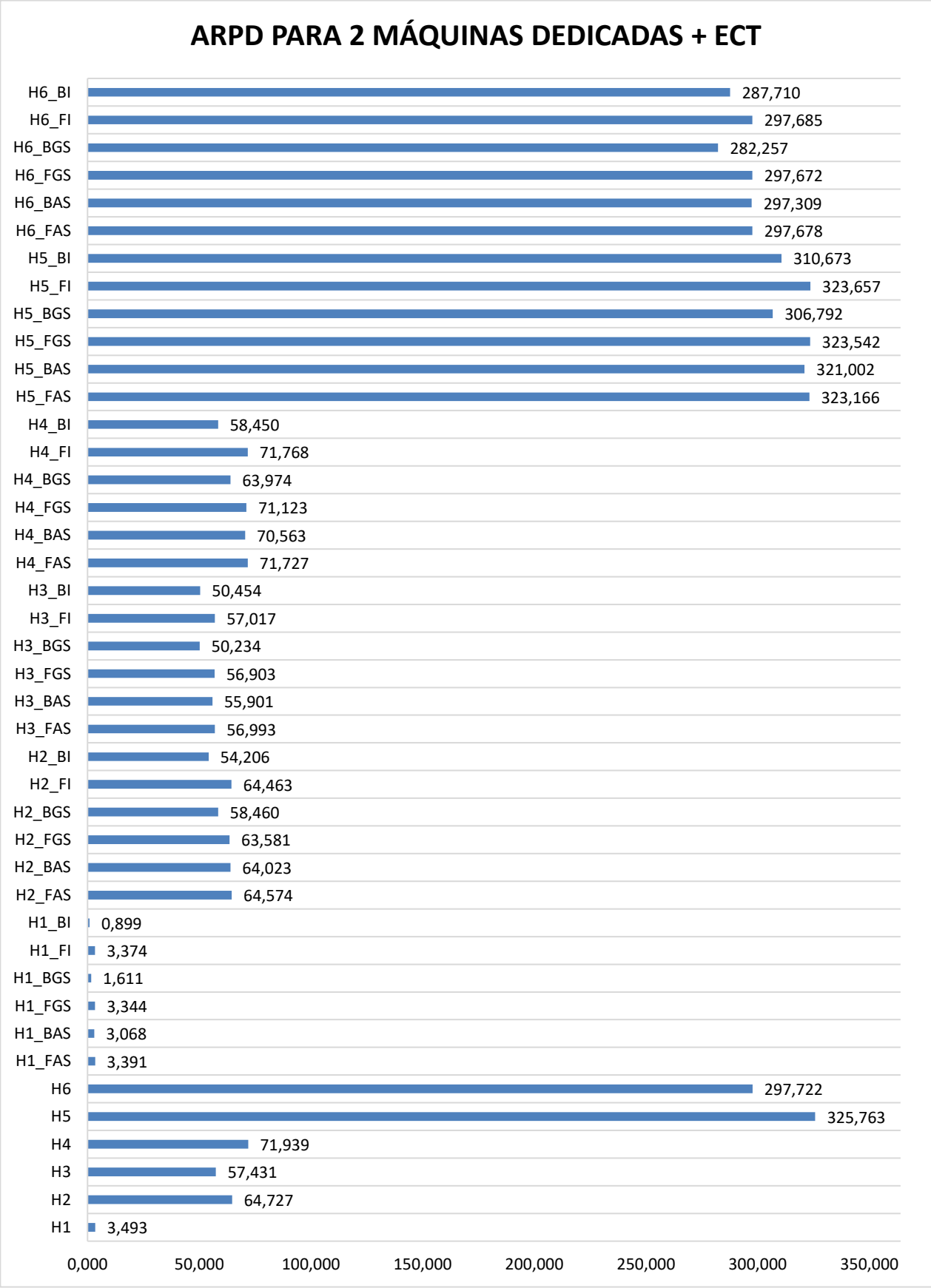

Figura 5-27. ARPD para $m_{ded} = 2$ + ECT

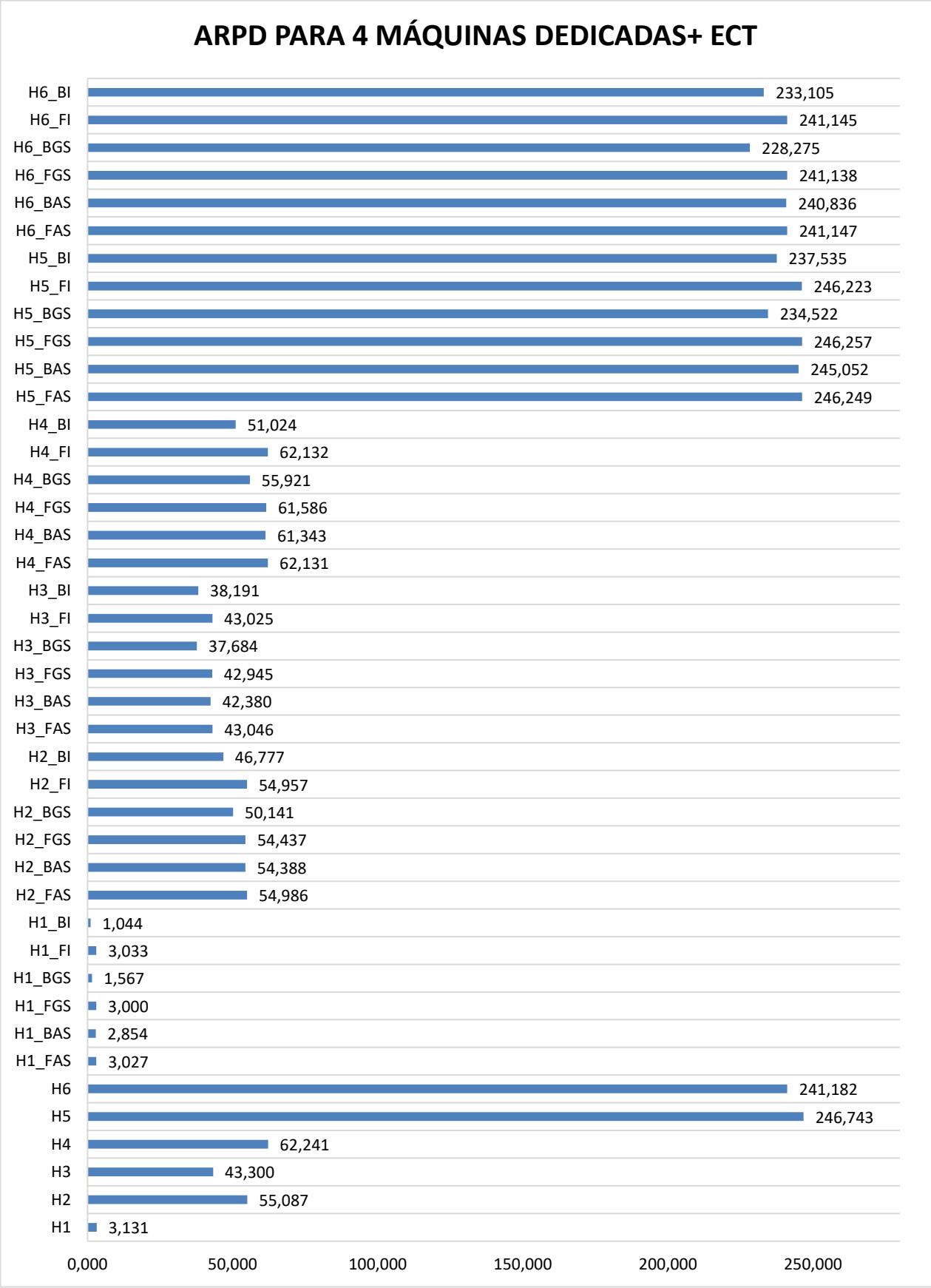

Figura 5-28. ARPD para $m_{ded} = 4 + ECT$

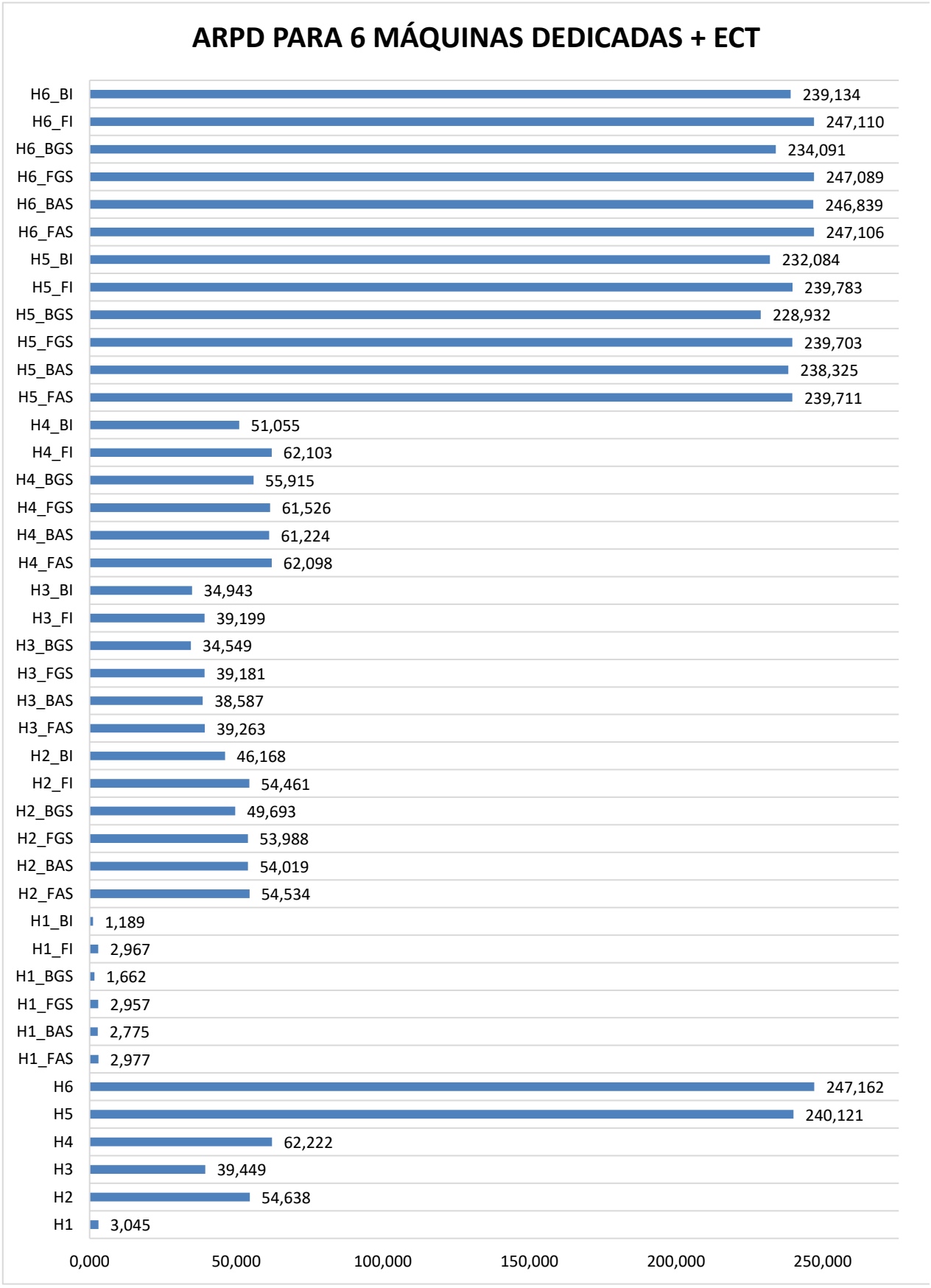

Figura 5-29. ARPD para m_{ded} = 6 + ECT

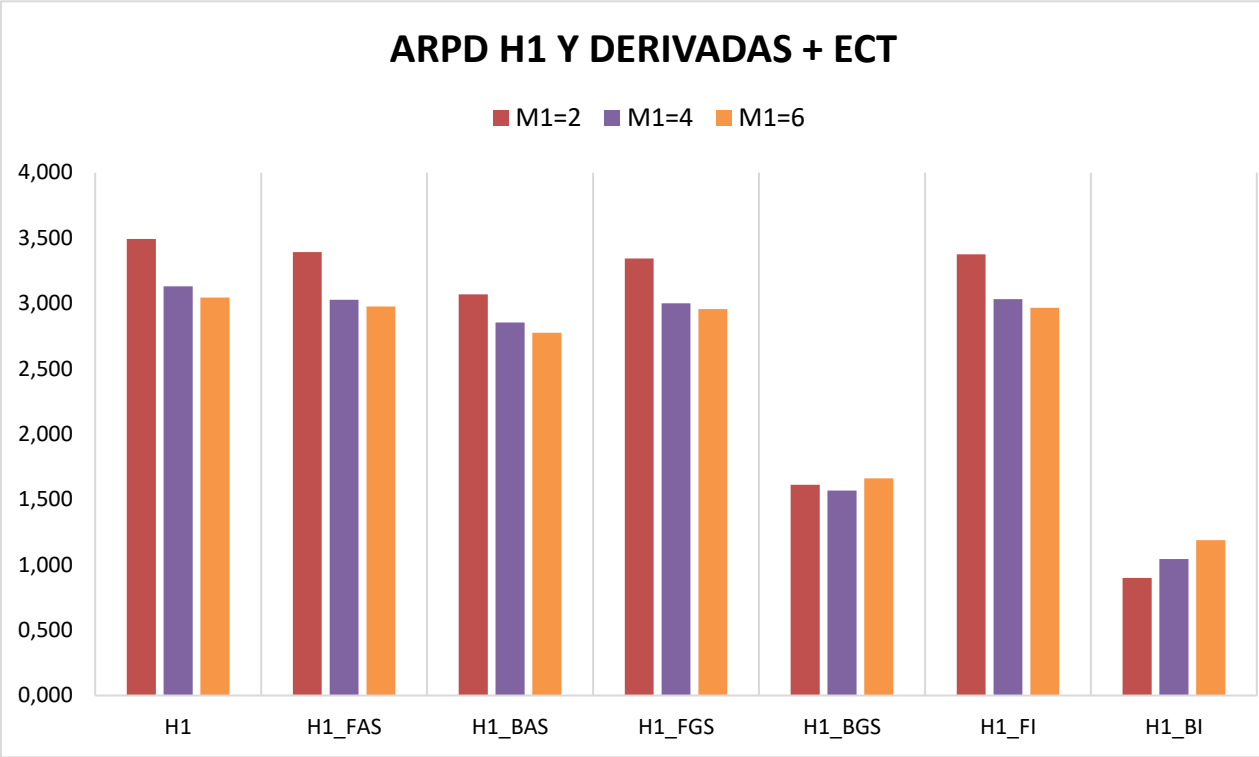

Figura 5-30. ARPD H1 y derivadas para valores de m_{ded} + ECT

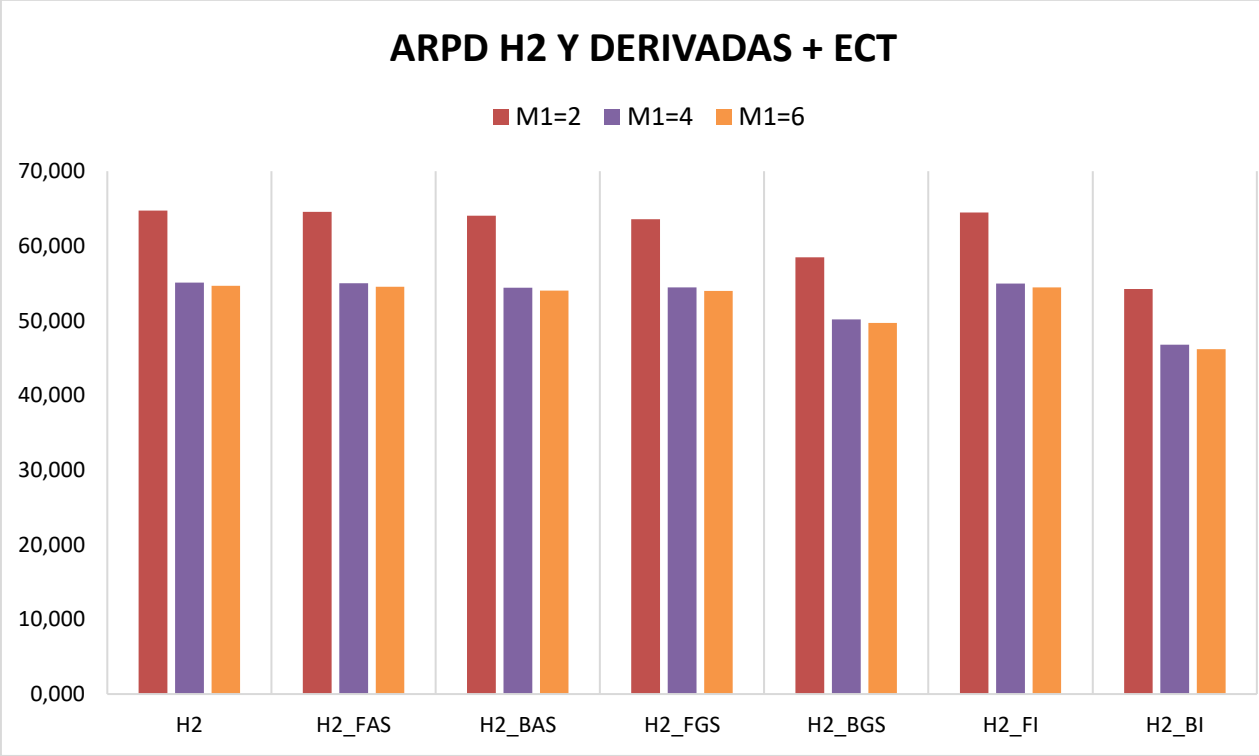

Figura 5-31. ARPD H2 y derivadas para valores de m_{ded} + ECT

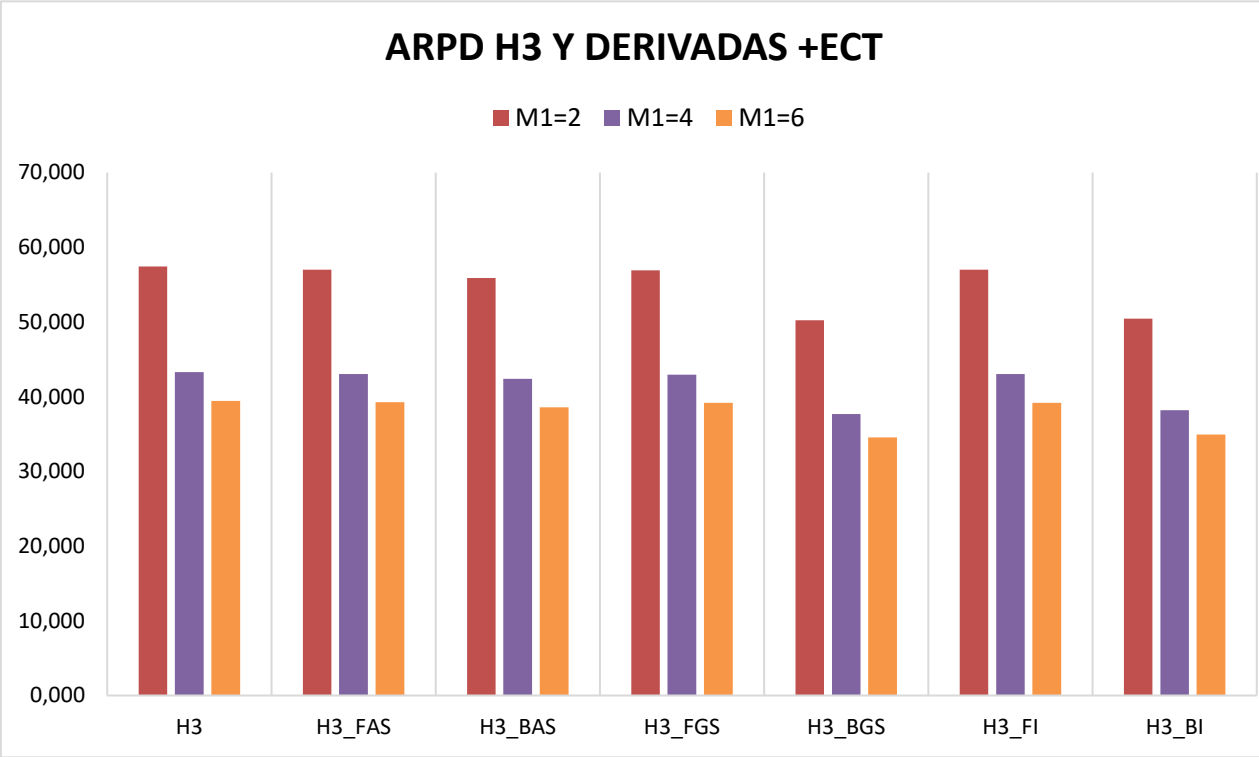

Figura 5-32. ARPD H3 y derivadas para valores de m_{ded} + ECT

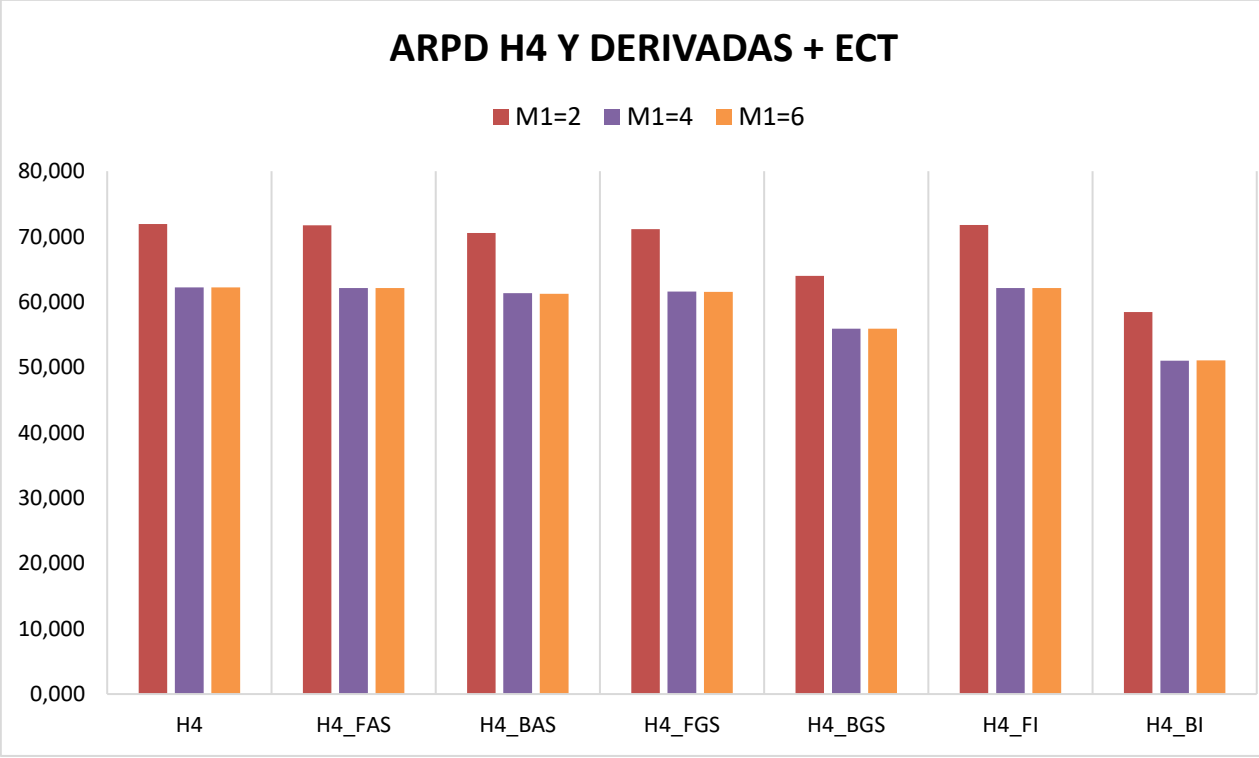

Figura 5-33. ARPD H4 y derivadas para valores de m_{ded} + ECT

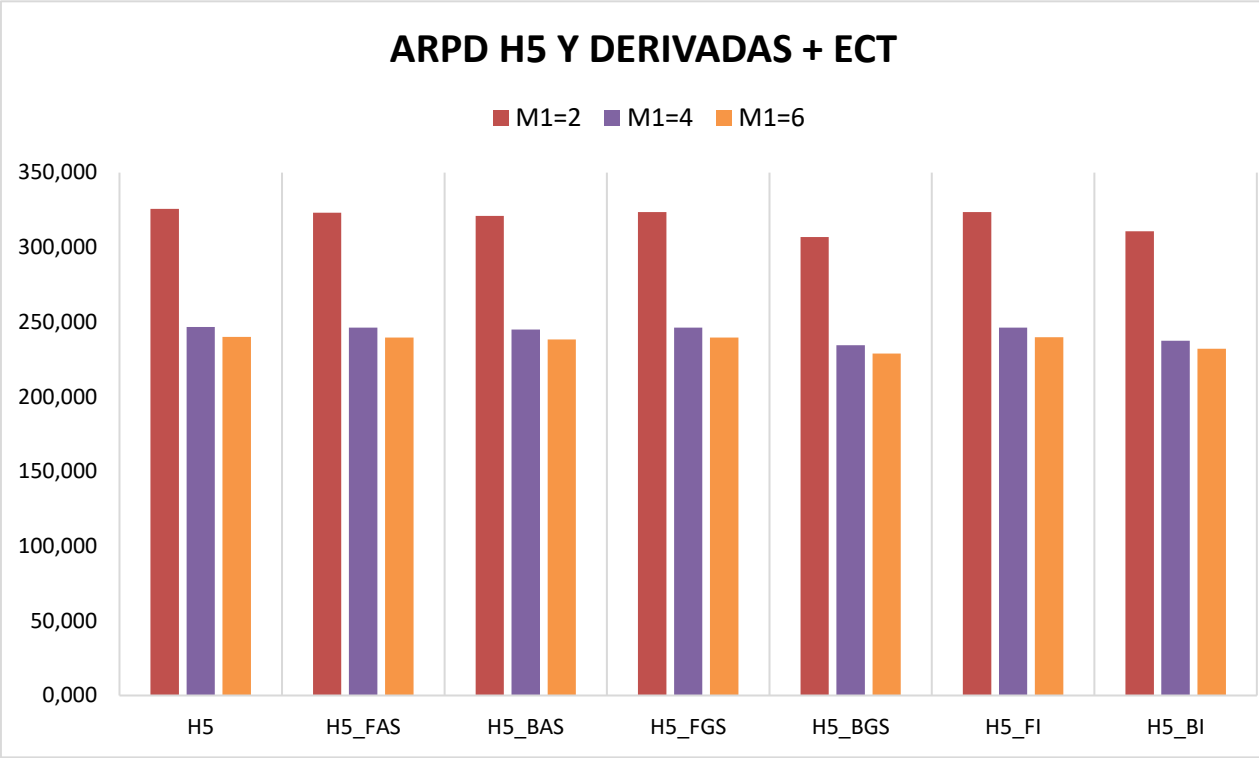

Figura 5-34. ARPD H5 y derivadas para valores de m_{ded} + ECT

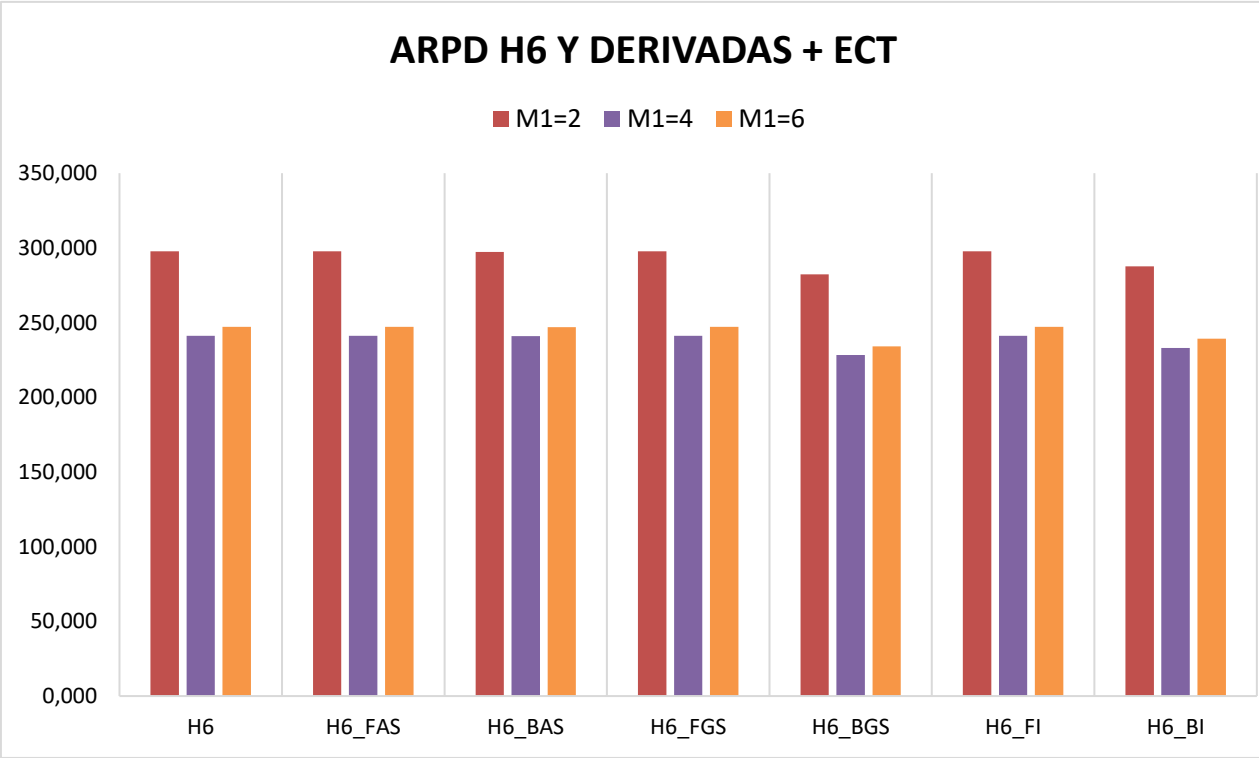

Figura 5-35. ARPD H6 y derivadas para valores de m_{ded} + ECT

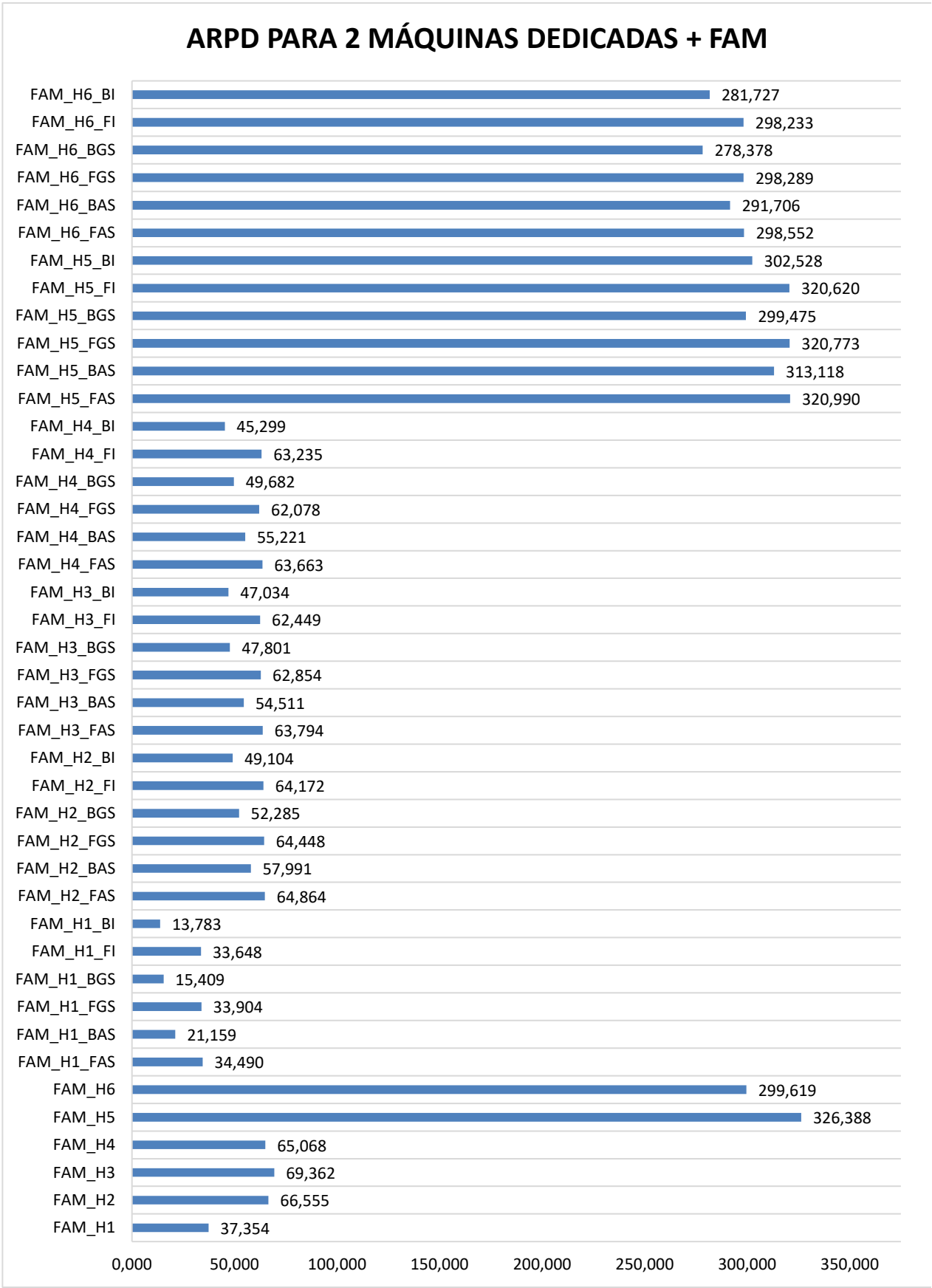
ARPD PARA 2 MÁQUINAS DEDICADAS + FAM
FAM_H6_BI 281,727
FAM_H6_FI 298,233
FAM_H6_BGS 278,378
FAM_H6_FGS 298,289
FAM_H6_BAS 291,706
FAM_H6_FAS 298,552
FAM_H5_BI 302,528
FAM_H5_FI 320,620
FAM_H5_BGS 299,475
FAM_H5_FGS 320,773
FAM_H5_BAS 313,118
FAM_H5_FAS 320,990
FAM_H4_BI 45,299
FAM_H4_FI 63,235
FAM_H4_BGS 49,682
FAM_H4_FGS 62,078
FAM_H4_BAS 55,221
FAM_H4_FAS 63,663
FAM_H3_BI 47,034
FAM_H3_FI 62,449
FAM_H3_BGS 47,801
FAM_H3_FGS 62,854
FAM_H3_BAS 54,511
FAM_H3_FAS 63,794
FAM_H2_BI 49,104
FAM_H2_FI 64,172
FAM_H2_BGS 52,285
FAM_H2_FGS 64,448
FAM_H2_BAS 57,991
FAM_H2_FAS 64,864
FAM_H1_BI 13,783
FAM_H1_FI 33,648
FAM_H1_BGS 15,409
FAM_H1_FGS 33,904
FAM_H1_BAS 21,159
FAM_H1_FAS 34,490
FAM_H6 299,619
FAM_H5 326,388
FAM_H4 65,068
FAM_H3 69,362
FAM_H2 66,555
FAM_H1 37,354
0,000
50,000
100,000
150,000
200,000
250,000
300,000
350,000

Figura 5-36. ARPD para m_{ded} = 2 + FAM

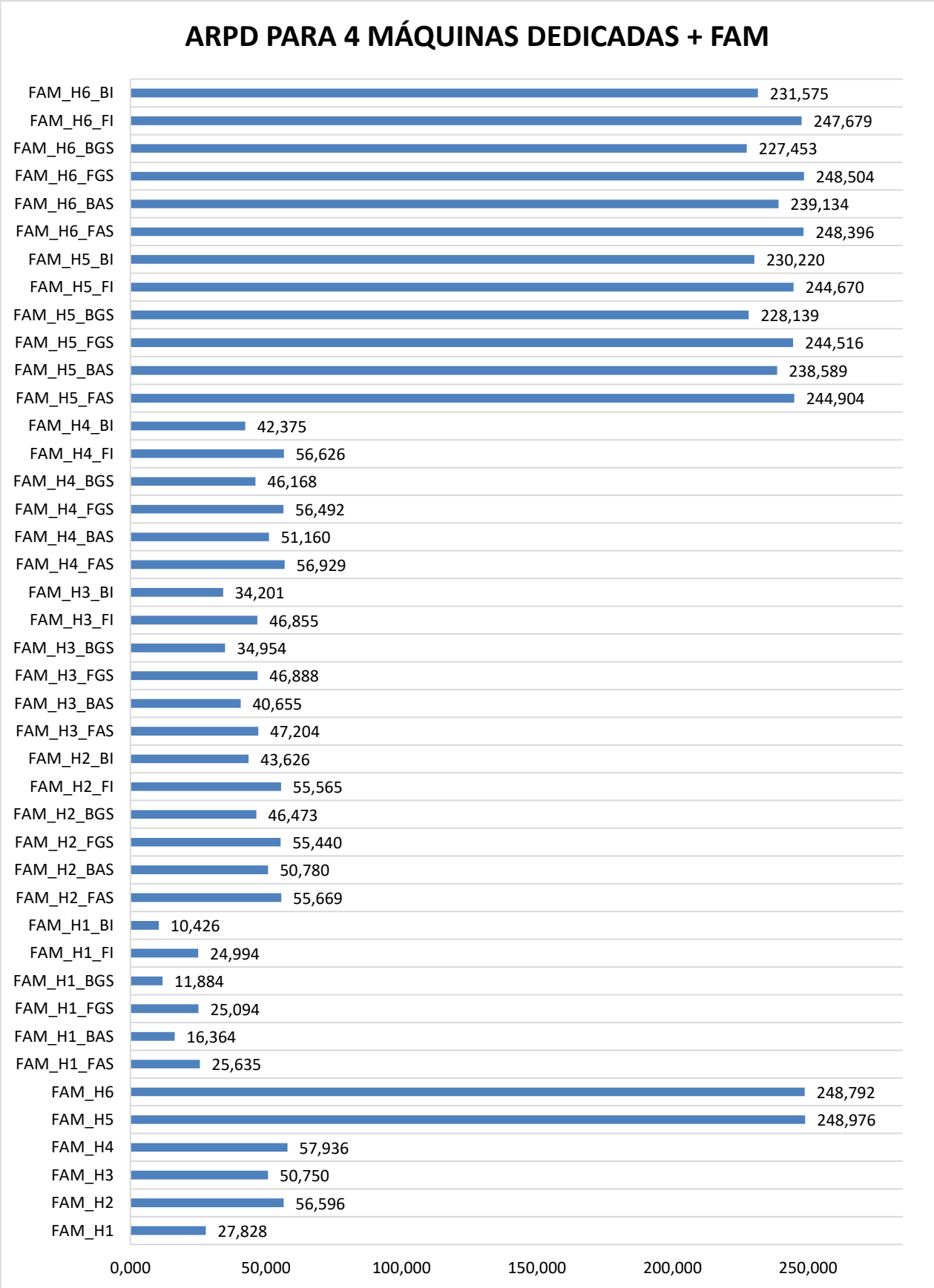

Figura 5-37. ARPD para $m_{ded} = 4 + FAM$

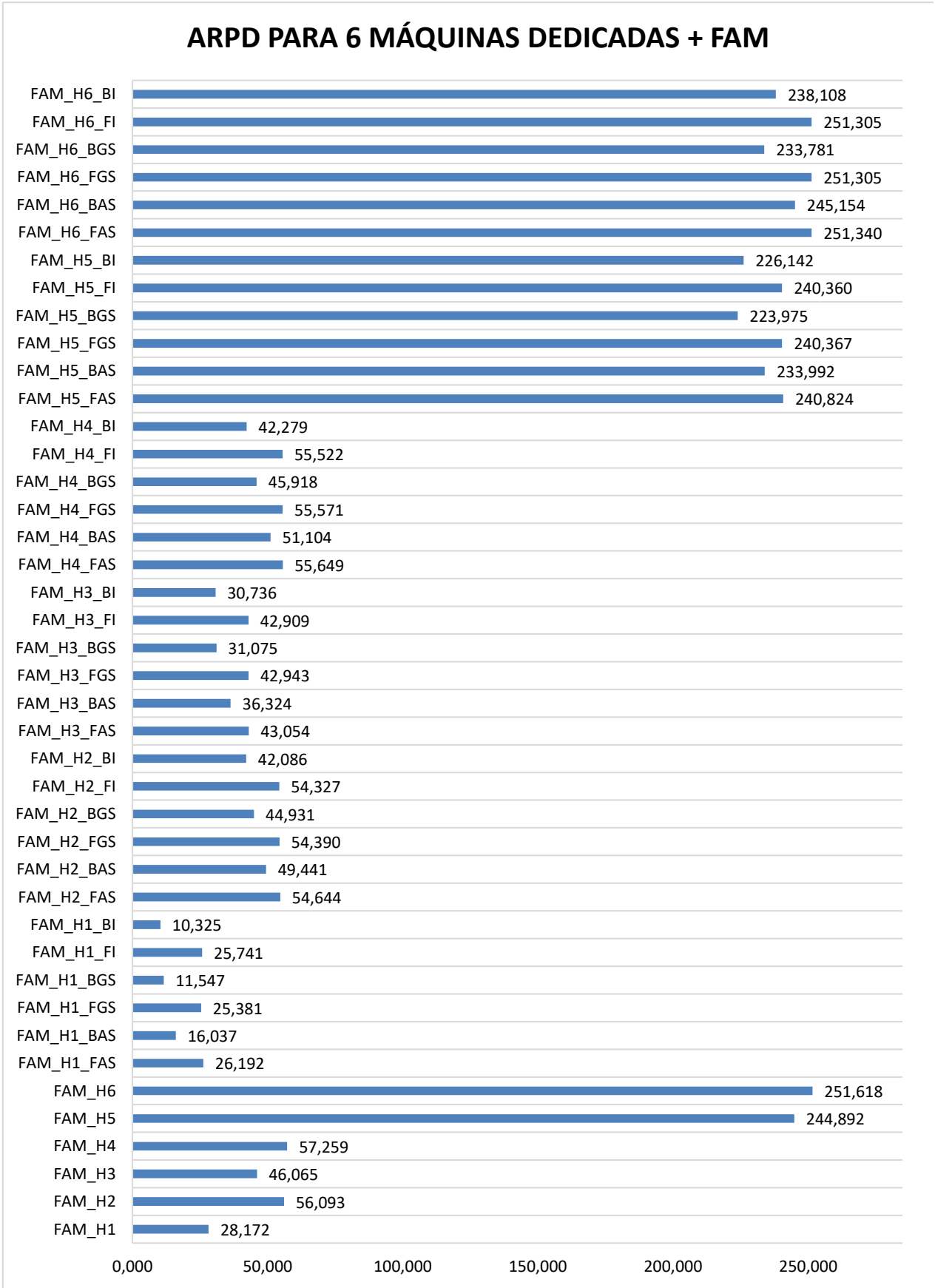

Figura 5-38. ARPD para m_{ded} = 6 + FAM

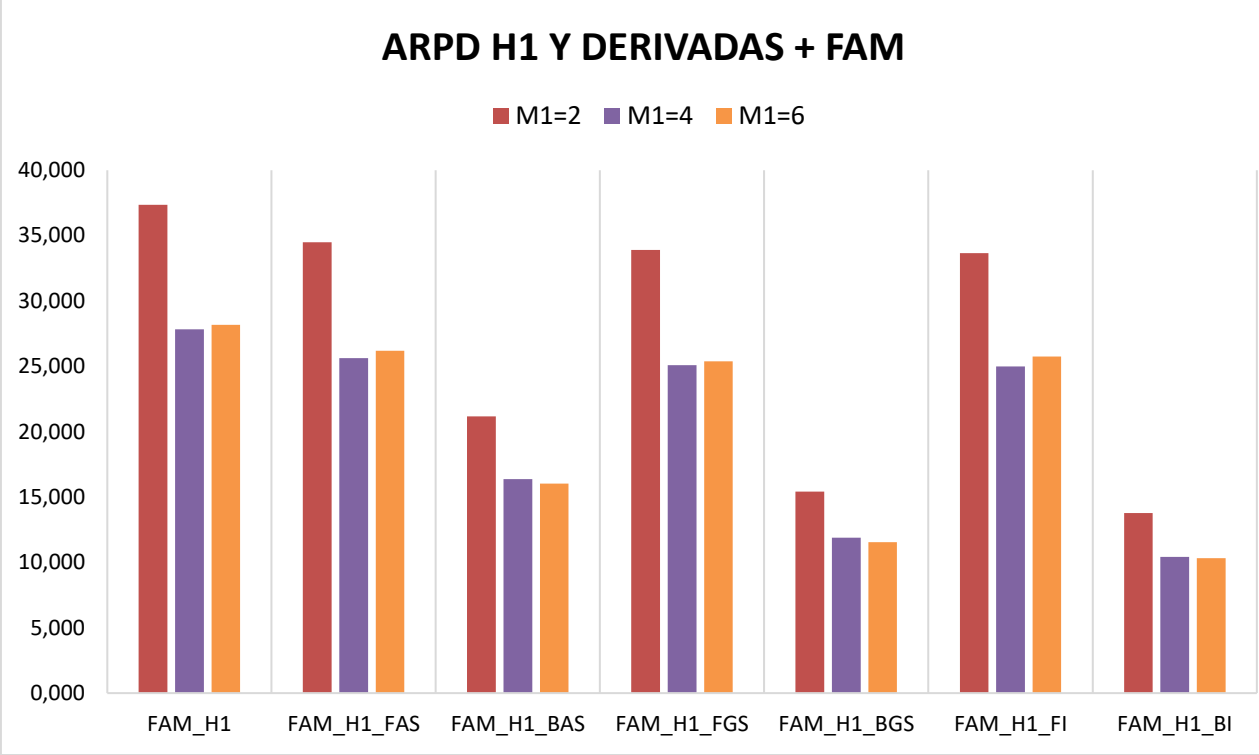

Figura 5-39. ARPD H1 y derivadas para valores de m_{ded} + FAM

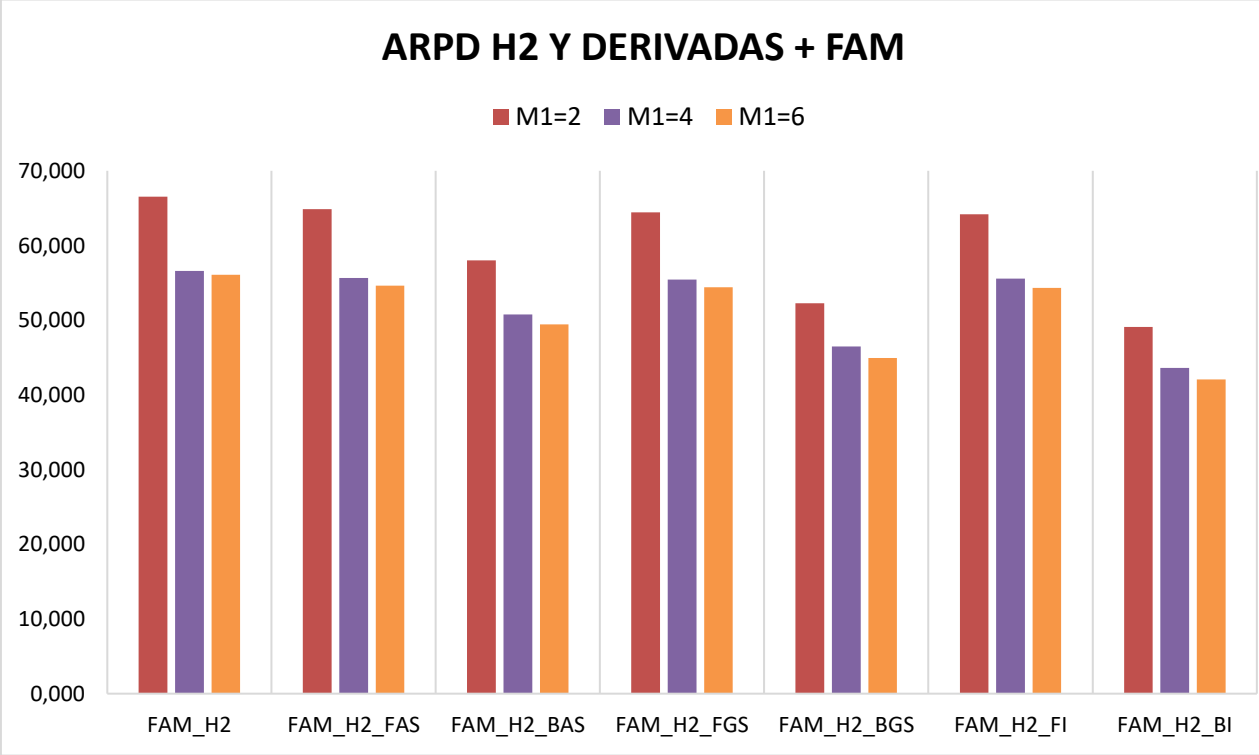

Figura 5-40. ARPD H2 y derivadas para valores de m_{ded} + FAM

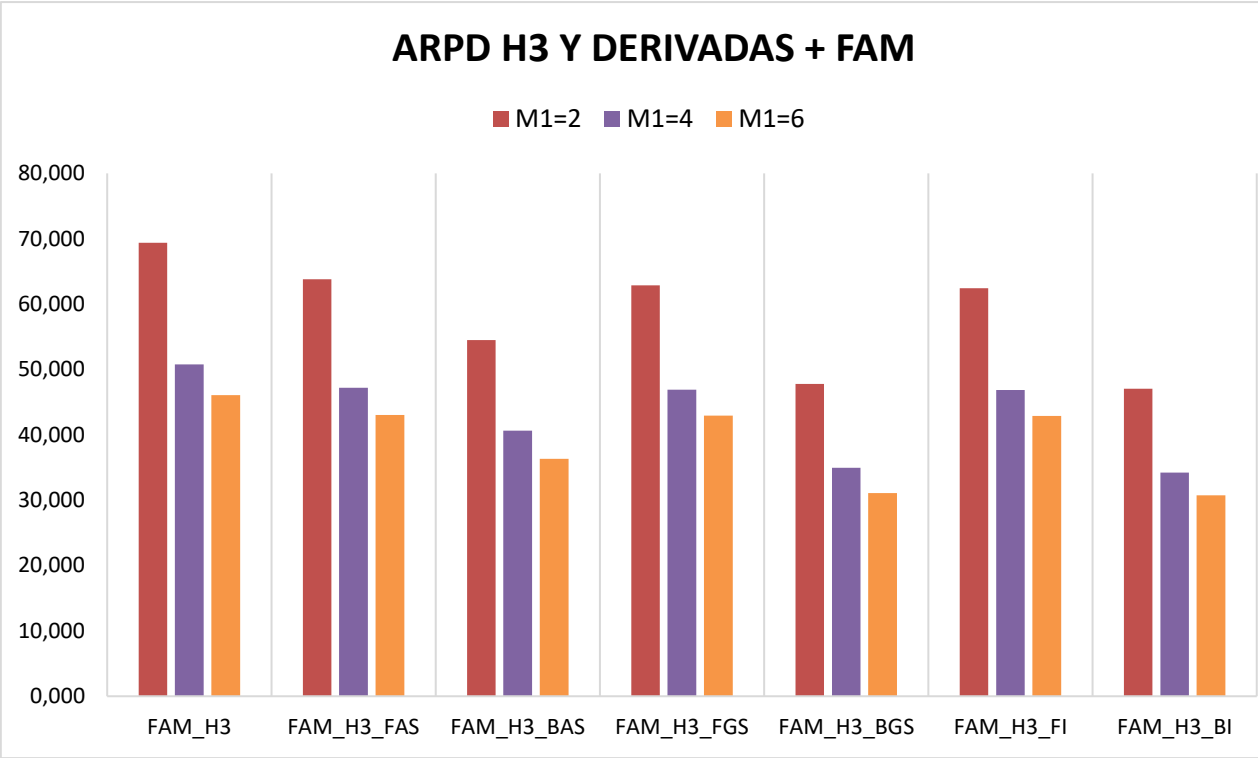

Figura 5-41. ARPD H3 y derivadas para valores de m_{ded} + FAM

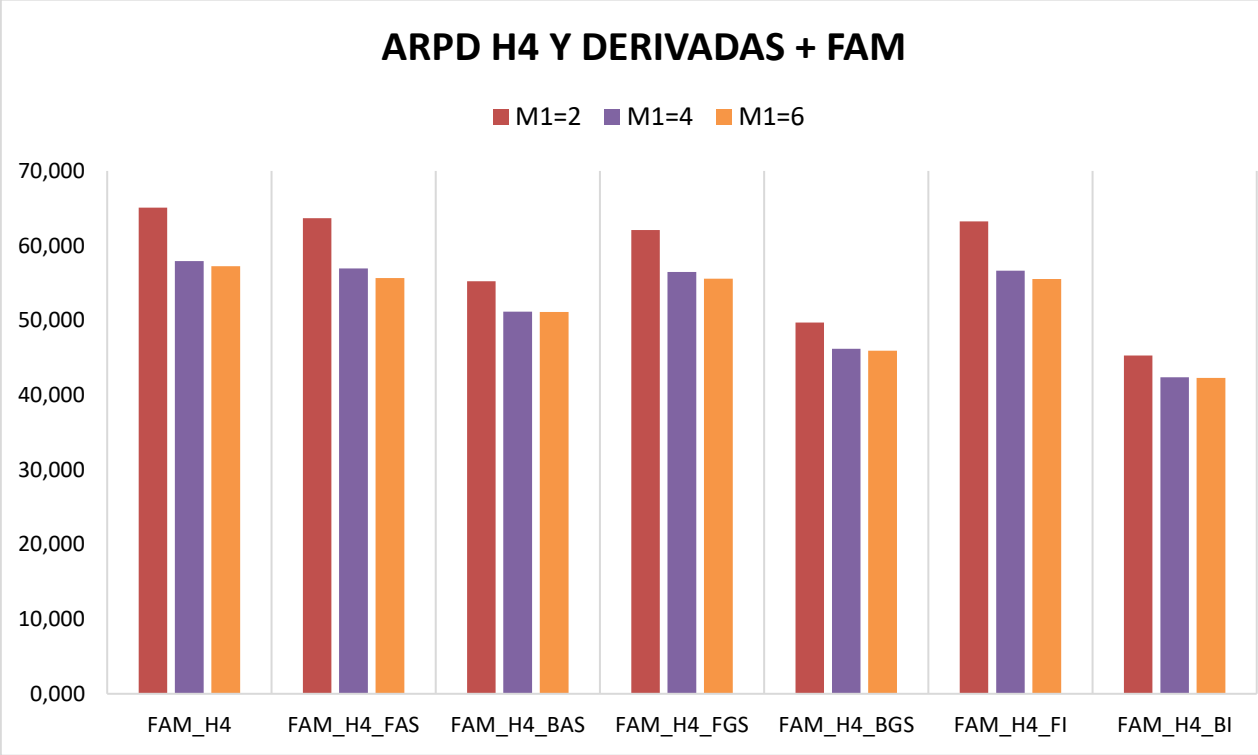

Figura 5-42. ARPD H4 y derivadas para valores de m_{ded} + FAM

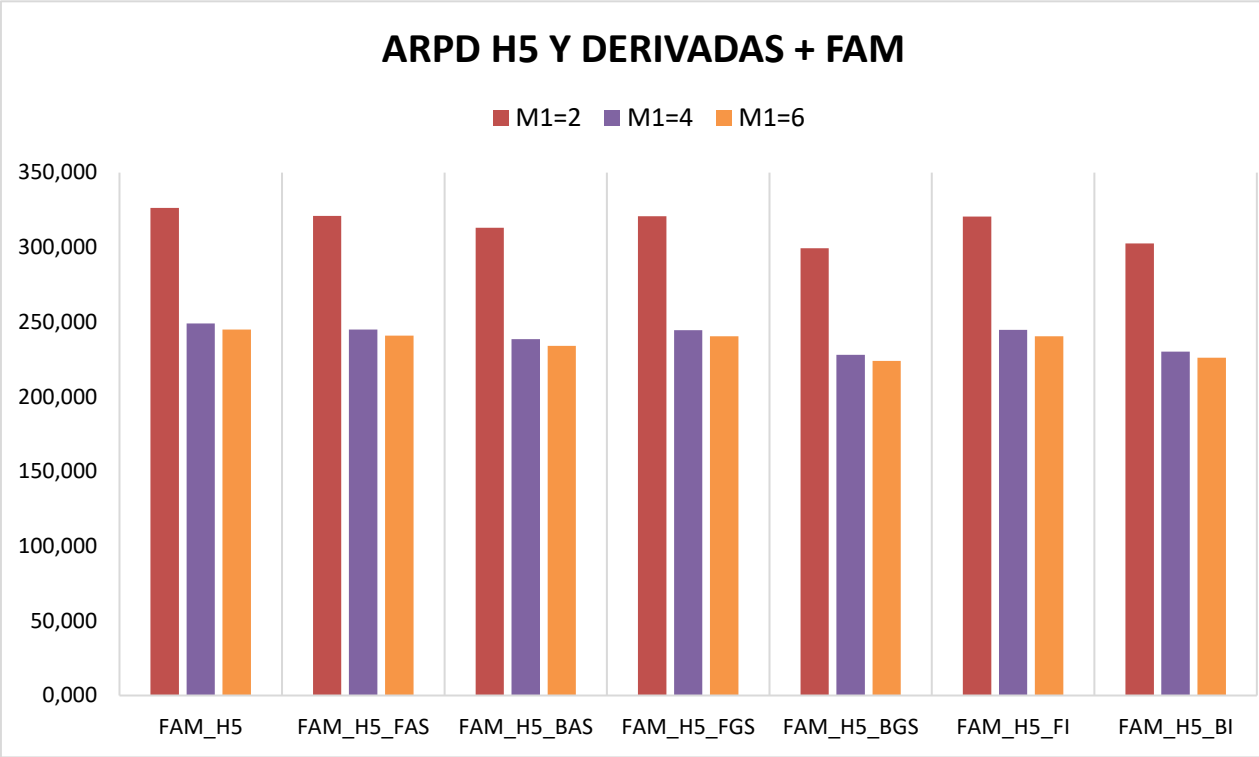

Figura 5-43. ARPD H5 y derivadas para valores de m_{ded} + FAM

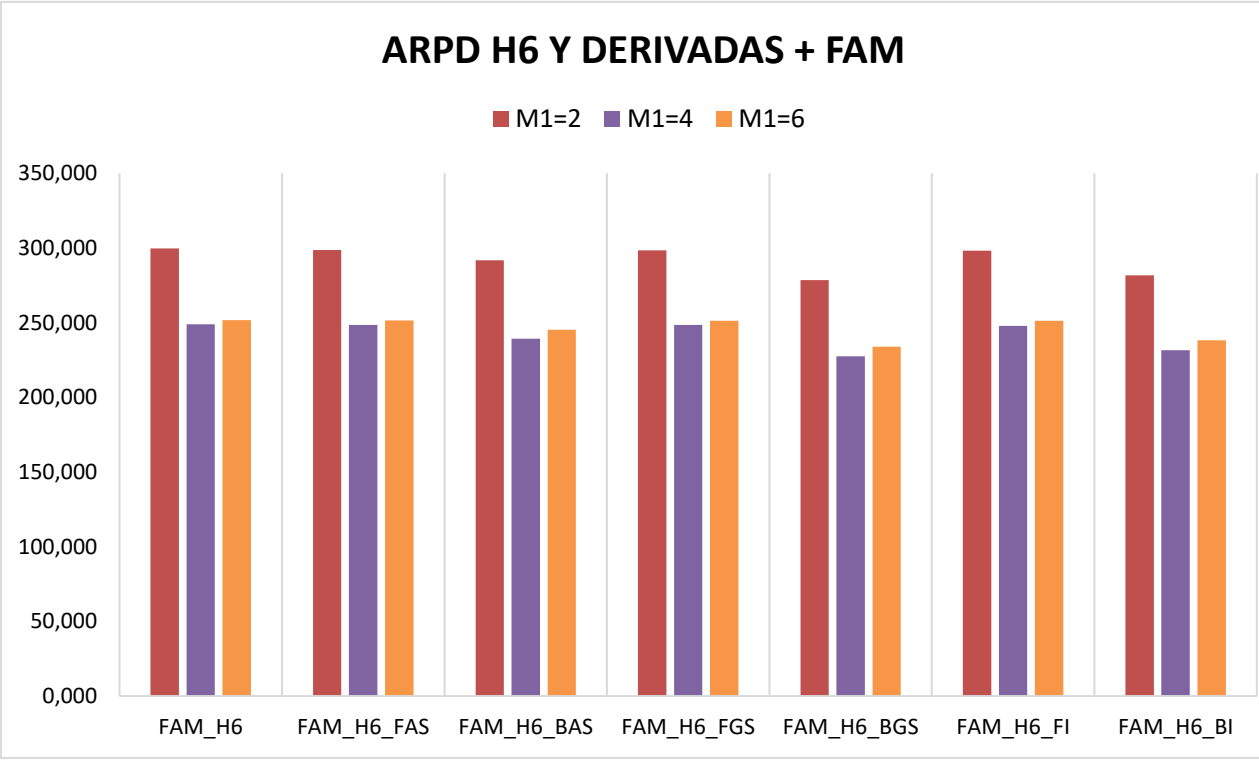

Figura 5-44. ARPD H6 y derivadas para valores de m_{ded} + FAM

6 CONCLUSIONES

Este proyecto se ha desarollado con el objetivo de resolver el problema de ensamblado en dos etapas con filosofía *just-in-time* haciendo uso de heurísticas constructivas y de mejora. Antes de comenzar a desarrollar los programas necesarios para tratar de dar solución al problema, se llevó a cabo una fase de documentación en la que se buscaron posibles soluciones a problemas de corte similar o que, al menos, hicieran uso de heurísticas constructivas. En esta búsqueda, se encontró un artículo (Perez-Gonzalez y Framiñán, 2017) en el que se ofrecía una heurística contructiva de mejora con el objetivo de minimizar los retrasos. Esta heurística, ha sido el principal punto de partida y modelo para el desarrollo de las heurísticas que a lo largo de este documento se han presentado.

Posteriormente, se llevó a cabo la resolución a mano de un caso concreto de este problema a pequeña escala para facilitar la labor de encontrar los puntos clave en la resolución de este. Gracias a ello, se identificó que lo más importante para la resolución de este problema es disponer de una buena secuencia de asignación y hacer uso del mejor criterio de asignación posible para la segunda etapa. A partir de este instante se desarrolló el programa necesario para la evaluación de una secuencia en base a la función objetivo del problema.

Con el programa ya desarrollado, se pudo dar comienzo al desarrollo de nuevas heurísticas contructivas con el objetivo de encontrar la mejor secuencia posible. De este proceso resultaron 6 heurísticas diferentes que se combinaron con 6 búsquedas locales, dando lugar a 36 nuevas heurísticas que, junto con las 6 primeras, formaron un total de 42. Posterior a este proceso, llegaría el momento de evaluar las secuencias ofrecidas por cada una de las heurísticas. Siendo este instante en el que se decidió hacer uso de 2 criterios de asignación diferentes: ECT y FAM.

Una vez definidos estos pasos, solamente hubo que generar 675 instancias diferentes con las que poner a prueba las heurísticas y los criterios y analizar los resultados. De este análisis se obtuvieron una serie de conclusiones:

1. Para problemas cuyos objetivos estén relacionados con el instante en el que cada trabajo termina de ser procesado, el criterio ECT es mejor que FAM, ya que la asignación se realiza en base a ese instante de terminación, mientras que FAM realiza la asignación en base a que la máquina minimice los tiempos de parada.
2. Para la resolución de este problema, la heurística que mejores resultados ofrece es H1_BI, independientemente del tamaño de la instancia. A pesar de esto, todas las heurísticas derivadas de H1 ofrecen valores notablemente buenos y, en algunos casos, con tiempos de ejecución menores.
3. De las heurísticas H3, H4 y sus derivadas, se puede intuir que las fechas de entrega son muy holgadas, ya que se producen muchos adelantos. Esto abre la puerta a un nuevo proyecto dedicado a la generación de fechas de entrega, de forma que no sean muy holgadas ni muy ajustadas.
4. La solución a problemas de retrasos en entornos de fabricación reales no siempre pasa por la implementación de nuevas máquinas, si no por la optimización del tiempo disponible por las ya existentes mediante criterios de asignación adecuados.
5. La heurística H3 y sus derivadas pueden llegar a ser una potencial solución a un problema en un entorno similar al desarrollado en este problema con el objetivo de minimizar los retrasos.

A partir de las heurísticas propuestas, se pueden abrir nuevas líneas de investigación para el desarrollo de soluciones a otros problemas de diferentes entornos con filosofía *just-in-time.*

REFERENCIAS

[1] Pérez, P. et al. (2021). *Programación de Operaciones.* Escuela Técnica Superior de Ingeniería Universidad de Sevilla, Sevilla.

[2] Pérez, P. & Framiñán, J. M. (2017). "Order Scheduling with tardiness objective: Improved approximate solutions" *European Journal of Operational Research*. Disponible en: https://idus.us.es/bitstream/handle/11441/68459/TROI-2016-02_v04.pdf?sequence=1

[3] Herrmann, J. (2006). *A history of production scheduling.* Springer US.

[4] Framinan, J. M., Leisten, R., & Ruiz, R. (2014). *Manufacturing scheduling systems: An integrated view on models, methods and tools*. Springer-Verlag London Ltd. Disponible en: https://doi.org/10.1007/978-1-4471-6272-8

[5] Ballesteros, P.P., Ballesteros, D.P., & Bravo, J.E. (2013). "Aplicación de una heurística constructiva en programación secuencial para asignación de varios trabajos a varias máquinas en paralelo". *Scientia et Technica Año XVIII, Vol. 18, No 1.* Disponible en: https://www.redalyc.org/pdf/849/84927487018.pdf

[6] Piersma, N. & Van Dijk, W. (1996). "A local search heuristic for Unrelated Parallel Machine Scheduling with Efficient Neighborhood Search". *Mathl. Comput. Modelling, Vol. 24, No. 9, pp. 11-19.* Disponible en: https://reader.elsevier.com/reader/sd/pii/0895717796001501?token=C5BE55CC7D90118C6642B3100F04253892A16A57EAB2ECE0F9BC9D359FB706C0F94B44B85F3FFD283B47F350492541E8&originRegion=eu-west-1&originCreation=20220708214748

[7] Graham et al. (1979). " Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey" *Annals of Discrete Mathematics, Vol. 5.* Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S016750600870356X

[8] Potts, C.N. & Van Wassenhove, L.N. (1982). " A decomposition algorithm for the single machine total tardiness problem" *Operations Research Letters, Vol. 1, Issue 5.* Disponible en: https://www.sciencedirect.com/science/article/abs/pii/0167637782900359