Trabajo Fin de Máster
Máster en Ingeniería Industrial

Análisis y resolución exacta de problemas flowshop de permutación multi agente.

Autor: Gonzalo Ameneiros Martínez

Tutora: Paz Pérez González

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

Sevilla, 2020

Trabajo Fin de Máster
Máster en Ingeniería Industrial

Análisis y resolución exacta de problemas flowshop de permutación multi agente

Autor:

Gonzalo Ameneiros Martínez

Tutora:

Paz Pérez González

Dpto. de Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Máster: Análisis y resolución exacta de problemas flowshop de permutación multi agente.

Autor: Gonzalo Ameneiros Martínez
Tutora: Paz Pérez González

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

ÍNDICE GENERAL

CAPÍTULO 1. INTRODUCCIÓN 1

1.1. OBJETIVOS 2

1.2. JUSTIFICACIÓN 3

1.3. SUMARIO 3

CAPÍTULO 2. CARACTERIZACIÓN DEL PROBLEMA. 5

2.1. CONCEPTOS BÁSICOS DE PROGRAMACIÓN. 5

2.2. NOTACIÓN 8

2.3. TRATAMIENTO DE LAS FUNCIONES OBJETIVO 9

CAPÍTULO 3. REVISIÓN DE LA LITERATURA. 12

3.1. INTRODUCCIÓN 12

3.2. PROBLEMAS MULTI-AGENTES DE FLOWSHOP DE PERMUTACIÓN 12

3.2.1. LINEAR CONVEX COMBINATION 13

3.2.2. ÉPSILON-CONSTRAINT 17

3.3. CONCLUSIONES 21

CAPÍTULO 4. MODELOS DE PROGRAMACIÓN LINEAL ENTERA. 24

4.1. INTRODUCCIÓN 24

4.2. NOTACIÓN 27

4.2.1. DATOS DE ENTRADA 28

4.2.2. ÍNDICES 28

4.2.3. VARIABLES 29

4.3. MODELOS DE PROGRAMACIÓN LINEAL 29

4.3.1. FAMILIA WAGNER. MODELO TS2. 29

4.3.2. FAMILIA MANNE. MODELO SGST 32

4.3.3. NUEVOS MODELOS. MODELO TS3. 34

4.4. CÁLCULO DE ÉPSILON. 37

4.5. PROGRAMACIÓN EN LENGUAJE C. 40

4.6. ELABORACIÓN DE BATERÍAS Y RESOLUCIÓN DE MODELOS. 43

4.7. RESULTADOS EXPERIMENTALES. 46

4.7.1. TIEMPOS COMPUTACIONALES TOTALES 46

4.7.2. TIEMPOS COMPUTACIONALES EN FUNCIÓN DEL NÚMERO DE MÁQUINAS 48

4.7.3. TIEMPOS COMPUTACIONALES EN FUNCIÓN DEL NÚMERO DE TRABAJOS 50
4.7.4. TIEMPOS COMPUTACIONALES EN FUNCIÓN DE R Y T 52
4.7.5. CONCLUSIONES 54
CAPÍTULO 5. INFLUENCIA DEL PARÁMETRO ÉPSILON EN LA RESOLUCIÓN DEL MODELO SGST. 56
5.1. INTRODUCCIÓN 56
5.2. ESCENARIO 1. DELTA=0. 57
5.3. ESCENARIO 2. DELTA=0.1 58
5.4. ESCENARIO 3. DELTA=0.2 60
5.5. ESCENARIO 4. DELTA=0.3 62
5.6. ESCENARIO 5. DELTA=0.4 65
5.7. ESCENARIO 6. DELTA=0.5 67
5.8. RESUMEN DE LOS ESCENARIOS 69
CAPÍTULO 6. CONCLUSIONES. 72
ANEXO. 74
Modelo SGST formato LP. EJEMPLO. 74
CÓDIGO SGST 75
CÓGIDO CREAR INSTANCIAS. 82
CÓDIGO EXPORTAR RESULTADOS FUNCIÓN OBJETIVO. 83
BIBLIOGRAFÍA. 84

ÍNDICE DE TABLAS

Tabla 1. Modelos usados por cada artículo _____ *22*
*Tabla 2. Algoritmos usados por cada artículo.*_____ *23*
Tabla 3. Datos de partida de ejemplo de cáculo de Épsilon. _____ *38*
Tabla 4. Media y desviación de los tiempos de cómputo totales. _____ *47*
Tabla 5. Intervalos de confianza 95% de los tiempos de cómputo totales. _____ *47*
Tabla 6. Media y desviación de los tiempos de computación en función de M. _____ *49*
Tabla 7. Intervalos de confianza 95% de los tiempos de computación en función de M. _____ *49*
*Tabla 8. Media y desviación de los tiempos de computación en función de N.*_____ *50*
*Tabla 9. Intervalos de confianza 95% de los tiempos de computación en función de N.*_____ *51*
Tabla 10. Media y desviación de los tiempos de computación en función de R y T. _____ *53*
Tabla 11. Intervalos de confianza 95% de los tiempos de computación en función de R y T. __ *53*
*Tabla 12. Valores makespan, épsilon y CPU time para delta=0*_____ *57*
*Tabla 13. Número de instancias sin resolver para delta=0.1*_____ *59*
*Tabla 14. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0 y 0.1*_____ *59*
*Tabla 15. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.1 con respecto a situación inicial.*_____ *59*
*Tabla 16. Número de instancias sin resolver para delta=0.2*_____ *61*
Tabla 17. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1 y 0.2 _____ *61*
*Tabla 18. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.2 con respecto a situación inicial.*_____ *61*
*Tabla 19. Número de instancias sin resolver para delta=0.3*_____ *63*
*Tabla 20. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1, 0.2 y 0.3*_____ *64*
*Tabla 21. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.3 con respecto a situación inicial.*_____ *64*
*Tabla 22. Número de instancias sin resolver para delta=0.4*_____ *66*
Tabla 23. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1, 0.2, 0.3 y 0.4 _____ *66*
*Tabla 24. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.4 con respecto a situación inicial.*_____ *66*
*Tabla 25. Número de instancias sin resolver para delta=0.5*_____ *68*
Tabla 26. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1, 0.2, 0.3, 0.4 y 0.5. _____ *68*
*Tabla 27. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.1 con respecto a situación inicial.*_____ *69*
Tabla 28. Porcentaje de instancias resueltas en todos los escenarios. _____ *70*
*Tabla 29. Porcentaje de aumento del makespan y del CPU time de todos los escenarios con respecto a delta=0.*_____ *70*

ÍNDICE DE ILUSTRACIONES

Ilustración 1. Ejemplo Diagrama de Gant. Fuente: Elaboración propia _____ *6*
Ilustración 2. Modelo Épsilon-constraint _____ *10*
Ilustración 3. Ejemplo gráfico Frente de Pareto. _____ *10*
*Ilustración 4. Explicación gráfica mantenimiento preventivo. Fuente: (Khelifati & Benbouzid-Sitayeb, 2011)*_____ *14*
*Ilustración 5. Media de tiempos de resolución de los modelos Total Completion Time. Fuente: (Ameneiros, 2018).*_____ *26*
Ilustración 6.Media de tiempos de resolución de los modelos Total Completion Time Ponderado. Fuente: (Ameneiros, 2018). _____ *26*
*Ilustración 7. Media de tiempos de resolución de los modelos Tardanza Total. Fuente: (Ameneiros, 2018).*_____ *27*
Ilustración 8. Media de tiempos de resolución de los modelos Tardanza Total Ponderada. Fuente: (Ameneiros, 2018). _____ *27*
*Ilustración 9. Secuencia [T0,T1], EJEMPLO CÁCULO DE ÉPSILON*_____ *38*
Ilustración 10.Secuencia [T1,T0], EJEMPLO CÁCULO DE ÉPSILON _____ *39*
*Ilustración 11. Secuencia [T1,T0, T3, T2], EJEMPLO CÁCULO DE ÉPSILON*_____ *39*
*Ilustración 12. Secuencia [T1,T0, T2, T3], EJEMPLO CÁCULO DE ÉPSILON*_____ *39*
*Ilustración 13. Ejemplo de formato LP. Fuente: https://www.gurobi.com/documentation/9.0/refman/lp_format.html*_____ *41*
*Ilustración 14. Formato de datos de entrada. Tiempos de procesado. Ejemplo.*_____ *42*
Ilustración 15. Formato de datos de entrada. Fechas de entrega. Ejemplo. _____ *42*
Ilustración 16. Salida por pantalla. Ejemplo. _____ *43*
*Ilustración 17. Batería para la generación de modelos.*_____ *44*
Ilustración 18. Batería para la resolución de modelos. _____ *45*
Ilustración 19. Archivo .SOL de una instancia de un modelo SGST. _____ *45*
Ilustración 20. Diagrama de flujo de resolución de instancias. _____ *46*
*Ilustración 21. Tiempos de computación medios de los modelos.*_____ *48*
Ilustración 22. Media y desviación de los tiempos de computación en función de M. _____ *49*
Ilustración 23. Media y desviación de los tiempos de computación en función de N (N=5, N=10). _____ *51*
*Ilustración 24. Media y desviación de los tiempos de computación en función de N (N=15, N=20).*_____ *52*
Ilustración 25. Media y desviación de los tiempos de computación en función de R y T. _____ *54*
Ilustración 26. CPU time para delta=0 _____ *57*
Ilustración 27. Porcentaje de aumento del makespan de las instancias resueltas _____ *60*
Ilustración 28. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,2 con respecto a delta=0. _____ *62*
Ilustración 29. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,3 con respecto a delta=0. _____ *65*
Ilustración 30. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,4 con respecto a delta=0. _____ *67*

Ilustración 31. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,4 con respecto a delta=0. _____ *69*
Ilustración 32. Porcentaje de aumento del makespan de todos los escenarios con respecto a delta=0. _____ *71*

CAPÍTULO 1.

INTRODUCCIÓN

Este trabajo está inmerso dentro de la Programación de Operaciones, un campo de estudio que se ocupa de los problemas donde el objetivo principal es la asignación de una serie de recursos a unas tareas dadas durante períodos de tiempo determinados para optimizar uno o más criterios (Pinedo, 2016). Como resultado de este proceso se obtiene un programa, que incluye dichas tareas denominadas trabajos y su asignación temporal en los recursos mencionados anteriormente llamados máquinas. Cabe destacar que los recursos no tienen por qué ser máquinas como tal, sino que puede ser por ejemplo un operario que realiza las tareas manualmente. Sin embargo, todos estos recursos utilizados para realizar las tareas son denominados máquinas en el ámbito de la Programación de operaciones.

Programar una serie de actividades en un entorno productivo es una tarea que puede llegar a complicarse debido a la gran cantidad de restricciones que pueden tener, tales como las relaciones de dependencia entre ellas o las rutas que deben cumplir en las máquinas, por lo que toma especial relevancia los diferentes algoritmos existentes para llegar a una solución óptima o en caso de que no sea viable llegar a ella, poder obtener una solución cercana.

Existen una gran cantidad de entornos de programación, sin embargo, este proyecto se ha centrado en el estudio del flowshop de permutación, un escenario de máquinas dispuestas en serie donde todos los trabajos deben pasar por todas las máquinas y en el mismo orden de secuencia. Además, en cada máquina solo se puede estar realizando una sola tarea y dichas actividades son ininterrumpidas, por lo que, una vez empezado el procesado de un trabajo en una máquina, ésta no estará disponible hasta que finalice el tiempo de proceso de dicha tarea en ella. Dicho entorno es el más estudiado en la literatura (Fernandez-Viagas & Framinan, 2019).

Como se ha mencionado anteriormente, el programa es realizado para optimizar una o varias funciones objetivo. Normalmente, en la Programación de Operaciones se suelen estudiar problemas con un solo criterio, sin embargo, en la vida real abundan las situaciones donde puede haber varios conjuntos de trabajos los cuales pueden tener funciones objetivos diferentes. En la literatura se estudian bastante los problemas de programación de operaciones con más de un objetivo (V.T Kind & J. C Billaut, 2002).

Un conjunto de trabajos se puede definir como un conjunto de tareas con su propia función objetivo independientemente del resto de tareas del entorno, que formarán parte de otros agentes a pesar de que comparten la utilización de los mismos recursos (máquinas). En este proyecto se ha realizado el estudio de un entorno dos conjuntos de trabajos.

Hay varios criterios existentes a optimizar en el ámbito de la Programación de Operaciones y los más comunes se irán desarrollando a lo largo de este trabajo.

1.1. OBJETIVOS

El objetivo principal de este proyecto reside en la realización de un análisis y estudio de la situación en la literatura actual de los problemas de programación de entornos flowshop con varios agentes. Una vez realizado este estudio se han desarrollado diversos modelos de programación lineal entera, adaptados de problemas clásicos estudiados en la literatura para realizar un análisis de la eficiencia de dichos modelos al resolverlos de forma óptima relacionándose la eficiencia con los recursos computacionales usados para ello. Por lo tanto, se comparan los tiempos utilizados por cada modelo para obtener la solución.

Para realizar esta fase del proyecto se ha hecho uso de la herramienta Gurobi (www.gurobi.com), un programa basado en un optimizador computacional que devuelve la solución óptima a partir de unos modelos aportados. Dichos modelos pueden ser realizados de forma manual, pero carece de sentido cuando el número de variables a tratar es muy grande, por lo que se hace uso de una programación computacional de dichos modelos.

Para realizar ambas fases se ha utilizado el programa informático Codeblocks (www.codeblocks.org), un entorno de desarrollo integrado de código abierto para programas en lenguaje C y C++, siendo el primero de ellos el lenguaje usado para este proyecto. Para la realización de la comparación estadística entre algoritmos y métodos se ha utilizado Microsoft Excel con diversas funcionalidades que existen dentro del programa.

Por tanto, la realización de este proyecto persigue los siguientes objetivos:

1. Vista general de los problemas de Programación de Operaciones haciendo uso de un desarrollo de los principales conceptos de esta disciplina y en concreto de los problemas multi agentes.
2. Revisión de la literatura: Análisis y resumen de los diferentes estudios realizados acerca de los problemas de flowshop de permutación con varios conjuntos de trabajos.
3. Estudio de los modelos de programación lineal existentes en la literatura acerca de este tipo de problemas y su adaptación al problema estudiado.
4. Programación de dichos modelos adaptados en el lenguaje C haciendo uso de Codeblocks.
5. Optimización realizada por Gurobi y su posterior análisis estadístico para determinar el modelo más eficiente.
6. Análisis estadístico de distintos parámetros para conocer la influencia en los resultados.
7. Realización de una conclusión de todos los resultados y datos aportados.

Los resultados obtenidos se han obtenido a partir de un ordenador con las siguientes características:

Modelo: Lenovo Ideapad S145.
Sistema operativo: Windows 10.

Procesador: Procesador Intel Core i5-8265U, QuadCore, 1.6-3.9GHz, 6MB.
Memoria RAM: 8GB DDR4, 2400Mhz.

1.2. JUSTIFICACIÓN

La programación es un proceso de toma de decisiones que se utiliza regularmente en muchas industrias manufactureras y de servicios, por lo que su estudio, realización y evolución es fundamental para conseguir los objetivos de cualquier empresa. Una incorrecta programación de las tareas a realizar será crítica para su día a día, dando lugar a una gran cantidad de pérdidas económicas y materiales. Es una evidencia que el estudio de modelos de programación y su posible optimización es fundamental para mejorar estos procesos, por lo que es necesario encontrar la forma de resolver dichos modelos en el menor tiempo posible, ya que como es sabido, el tiempo es un indicador clave en las operaciones de una empresa.

Como se mencionó en el apartado anterior, el estudio sobre este ámbito se ha centrado fundamentalmente a lo largo de la historia en la optimización de una sola función objetivo, sin embargo, es habitual que varios conjuntos de trabajos con funciones objetivos independientes tengan que ser optimizados en un entorno donde comparten la utilización de los recursos (máquinas). Por tanto, el objeto de estudio de este proyecto abarcará problemas de este tipo.

1.3. SUMARIO

Este proyecto se estructura en seis grandes capítulos. En el primero de ellos se ha realizado una introducción al trabajo realizado, explicando el motivo y la justificación del por qué se ha escogido este tema de estudio.

En el capítulo 2 se presenta una introducción a los conceptos generales de la programación de operaciones y a la notación utilizada, para permitir una mejor lectura del proyecto.

El capítulo 3 consta de una revisión de la literatura de los problemas flowshop multi agente, haciéndose un repaso de los artículos e investigaciones realizados en los últimos años.

En el capítulo 4 se hace un estudio de los modelos más eficientes de programación lineal existentes en la literatura acerca de los entornos de flowshop, para ser posteriormente adaptados a los problemas estudiados en este proyecto. Para finalizar con este capítulo se han resuelto de forma exacta numerosas instancias para comparar dichos modelos en eficiencia, haciendo un análisis estadístico de los tiempos de resolución computacionales.

Posteriormente, en el capítulo 5 se analiza la influencia que tiene uno de los parámetros fijados en el capítulo anterior en los resultados, creando diversos escenarios que son estudiados estadísticamente para definir los cambios producidos en ellos.

Para finalizar, en el capítulo 6 se realizan las conclusiones obtenidas al realizar dicho proyecto.

CAPÍTULO 2.

CARACTERIZACIÓN DEL PROBLEMA.

2.1. CONCEPTOS BÁSICOS DE PROGRAMACIÓN.

Antes de definir cualquier tipo de problema de programación es esencial conocer unos conceptos básicos sobre dicho campo de estudio. Estos conceptos han sido extraídos de (Pinedo, 2016), (Ameneiros, 2018) y de los apuntes de la asignatura de Programación de Operaciones. Curso 2017/2018 impartida en la ETSI de Sevilla (Perez González, Paz & Fernández-Viagas, Victor). Dichos conceptos son los siguientes:

-Máquina: Recurso con capacidad para realizar operaciones de transformación o transporte de material. Como se mencionó en la Introducción, una máquina puede ser un operario, un conjunto de trabajadores, un torno, un vehículo AGV etc. El número de máquinas se denota por la letra M.

-Trabajo: Es la tarea que debe ser realizada en las máquinas. No tiene que ser un producto como tal, sino que es una abstracción del concepto de trabajo. Un ejemplo puede ser un barco que debe ser construido, una pared que hay que pintar etc. El número de trabajos se denota por la letra N.

-Programar: Asignar las tareas a los recursos y añadirle un marco temporal, es decir, asignar su fecha de inicio y finalización. Como resultado se obtiene un programa.

-Programa factible: Programa que cumple con todas las restricciones impuestas en el modelo.

-Programa óptimo: Programa factible con la función objetivo optimizada. Es importante destacar que un programa óptimo es siempre un programa factible pero no viceversa.

-Conjunto de trabajos: También conocido como agente, y se define como un conjunto de trabajos, los cuales están sujetos a una o varias funciones objetivos. En un entorno de programación pueden existir varios conjuntos de trabajos que deben ser programados teniendo en cuenta sus interferencias y analizadas en su conjunto a pesar de que cada uno tenga su propia función objetivo.

Hay dos tipos de conjuntos de trabajos según si comparten trabajos o no. Se les denomina conjuntos de trabajos disjuntos a aquellos agentes que no comparten ningún trabajo, objeto de estudio de este proyecto. Por el contrario, los conjuntos de trabajos no disjuntos tienen en común uno o más trabajos.

-Ruta tecnológica: Camino que siguen los trabajos en las diferentes máquinas. Es importante destacar que no todos los trabajos deben tener la misma ruta, sin embargo, en el entorno

productivo analizado en este proyecto todos los trabajos pertenecientes a todos los agentes seguirán el mismo camino por las diferentes rutas.

-Diagrama de Gant: Representación gráfica temporal del programa.

-Flow shop (Fm): Entorno de producción donde existen varias máquinas dispuestas en serie. Todos los trabajos deben seguir la misma ruta, siendo procesados primero en la máquina uno, después en la dos y así sucesivamente hasta llegar a la última máquina.

-Permutación: Es una restricción que se le añade a un entorno Flow shop y que obliga a que las máquinas operen según la disciplina FIFO (First In First Out), esto es, el orden en que los trabajos son procesados en la primera máquina es mantenido en el resto.

A continuación, se muestra en la figura 1 un ejemplo gráfico de los tres conceptos anteriores: un diagrama de Gant de un Flow shop de tres máquinas y tres trabajos con permutación. Como se puede observar, la ruta de todos los trabajos es la misma, siendo procesados todos ellos primeramente en la máquina uno, para ser llevados posteriormente a la máquina dos y tres consecutivamente. Además, cada trabajo en las máquinas no puede empezar a ser procesado hasta que no ha finalizado su operación en la máquina anterior. La secuencia elegida ha sido (T1, T2, T3).

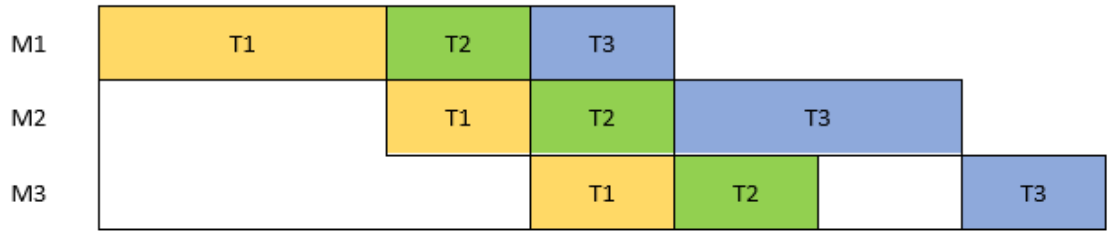

Ilustración 1. Ejemplo Diagrama de Gant. Fuente: Elaboración propia

Es importante destacar que éste no es el único programa factible, puesto que un entorno de Flow shop de permutación tiene n! posibles soluciones, por lo que a mayor número de trabajos el problema se complica mucho más.

Además, para añadir el concepto de conjuntos de trabajos, en el ejemplo anterior se puede considerar que el trabajo uno y dos forman parte de un agente y el trabajo tres está incluido en otro agente con otra función objetivo. El objetivo sería realizar un programa único como el de la figura 1 que optimice ambos objetivos.

Los datos principales de entrada en un problema de Programación de Operaciones son los siguientes:

- Número de trabajos. Se denomina con la letra N.

- Número de máquinas. Se denomina con la letra M.

- Tiempos de proceso: Duración que requiere la realización de un trabajo en una máquina.

- Fechas de entrega: Son los instantes de tiempo comprometidos para tener terminado el trabajo. Las fechas de entrega en general no son de obligatorio cumplimiento, por lo que los trabajos pueden acabar tarde.

- Pesos o prioridades: Indican la importancia de los trabajos. Este concepto aparecerá en el modelo en las funciones objetivos como un factor multiplicado a la variable estudiada.

Una vez explicados dichos conceptos el siguiente paso es definir como queda caracterizado un problema según la notación establecida por (Graham, Lawler, Lenstra & Rinnooy, 1979). En dicha notación se determina que un problema de programación queda definido por tres campos independientes:

1. Entorno α: Indica las características de las máquinas. Expresa la disposición de las máquinas. Existen diversos entornos y quedan definidos con un número o una letra. En el caso de este proyecto se estudia el Flow Shop (Fm).

2. Restricciones β: Indica las características de los trabajos. En este trabajo se estudian los trabajos con la restricción impuesta de permutación. (prmu).

3. Criterio ϒ: Función objetivo del problema. Como se ha visto anteriormente, puede haber varias y abarcar distintos conjuntos de trabajo. Los criterios siempre se minimizarán. Hay diversos criterios tradicionales y los más usados son los siguientes:

-Total Tardiness: Los trabajos tienen unas fechas de entrega y la función objetivo trata de minimizar la diferencia de todos los trabajos entre los tiempos de finalización en la última máquina y dichas fechas de entrega. En consecuencia de ello, si los trabajos terminan antes de su fecha de entrega, la tardanza de dichos trabajos será cero.

-Total Completion Time: La función objetivo consiste en minimizar la suma de todos los tiempos de finalización de todos los trabajos en la última máquina.

-Makespan: Es el criterio tradicional más conocido y estudiado y consiste en minimizar el tiempo de finalización en la última máquina del último trabajo de la secuencia, es decir, el tiempo en el que se habrán procesado todos los trabajos en todas las máquinas.

Además, es importante destacar que existen funciones objetivos ponderadas, en las cuales se le añade un peso a cada trabajo según la importancia que se le dé.

Por lo tanto, el problema puede quedar definido por:

$$\alpha \,|\beta|\, \Upsilon$$

Para el problema estudiado en este proyecto se tienen en cuenta varias hipótesis:

-Todos los trabajos están disponibles al inicio del cronograma.

-Los trabajos no se pueden interrumpir, por lo que una vez empezados a ser procesados deben terminar en la máquina en el tiempo de procesado dado.

-Cada máquina puede procesar un trabajo. Además, cada trabajo solo puede ser procesado en una máquina.

-Las máquinas, en general (excepto en modelos que lo consideren), no sufren averías ni tareas de mantenimiento preventivo, por lo que están en todo momento disponibles para procesar los trabajos.

-El buffer entre máquinas se supone infinito, es decir, el número de trabajos que pueden esperar antes de ser procesado en cada máquina es ilimitado.

-No se tienen en cuenta los tiempos de transporte de material entre máquinas, por lo que dicho tiempo se supone despreciable. Por lo tanto, una vez que un trabajo ha terminado de ser procesado en una máquina, está disponible para ser realizado en la siguiente máquina de la ruta.

-Los tiempos de preparación (set up) de las máquinas son independientes de la secuencia obtenida y están incluidos en los propios tiempos de procesado de los trabajos.

2.2. NOTACIÓN

La notación utilizada en este capítulo es la recomendada en (Perez-Gonzalez & Framinan, 2014), que unifica en un solo documento la notación utilizada en los diferentes artículos existentes dándoles un nombre común: MASPs “multi-agent scheduling problem”.

Cabe destacar que este capítulo se ha realizado analizando los diferentes artículos usando esta notación y no la original de dichos estudios. El objetivo de ello es facilitar una mejor comprensión de ellos. En el caso de que existiera una notación específica en algún artículo se añadirá a la general recomendada. En este apartado solo se recogerá la notación general común a todos los estudios realizados sobre este tipo de problemas. Dicha notación recomendada en (Perez-Gonzalez & Framinan, 2014) se recoge a continuación:

−J^k = Conjunto de trabajos k (k =1,2,… K), denominándose J al conjunto de todos ellos.

−Cada conjunto de trabajos tiene n^k trabajos.

−Se denomina N a la suma de todos los trabajos, cumpliéndose que:

$$N = n^1 + n^2 + \cdots n^k$$

−Cada conjunto de trabajos tiene una función objetivo f^k. La función objetivo global del modelo es denominada por f. En el caso de la existencia de un mismo criterio para todos los agentes, se nombra una función objetivo global común pasando el problema a denominarse “interfering jobs problem with global objective function”.

−Cada trabajo j ∈ J^k tiene que ser procesado en la máquina i con un tiempo de proceso de p_{ij}^k. En el hipotético caso de que existe solo una máquina, el tiempo de proceso se denota por p_j^k.

−Se define $p^k = \sum_{i=1}^{M} \sum_{j=1}^{n^k} p_{ij}^k$, es decir p^k es la suma de los tiempos de procesado en todas las máquinas del conjunto de trabajos k. Como consecuencia de esta definición surge el dato de entrada P, definido como la suma de todos los tiempos de proceso en todas las máquinas de todos los conjuntos de trabajo, es decir, $P = \sum_{i=1}^{M} \sum_{Jk=1}^{J} \sum_{j=1}^{n^k} p_{ij}^k$.

−La mayor parte de los problemas analizados en la actualidad tienen dos conjuntos de trabajos, por lo que en este proyecto se han denotado cada uno de los dos agentes como J^A y J^B, cada uno con n^A y n^B trabajos y con las funciones objetivo f^A y f^B respectivamente.

−σ= [$\sigma_{1,\ldots}\sigma_n$] es una secuencia de todos los trabajos pertenecientes a J, donde cada σ_j puede estar en cualquier J^k.

−Dada una secuencia σ, el completion time de un trabajo σ_j en una máquina i se denota por C_{ij}(σ) siendo C_j(σ) el completion time de un trabajo en la última máquina que debe ser procesada. $T_j(\sigma) = \max(C_j(\sigma) - d_j, 0)$ es la tardanza de un trabajo j con respecto a unas fechas de entrega dadas d_j. Se suele omitir σ si no es necesario.

2.3. TRATAMIENTO DE LAS FUNCIONES OBJETIVO

En los problemas multiobjetivo se consideran diferentes enfoques para abordar la optimización. Estos enfoques se extienden a los problemas MASPs.

1. Linear Convex Combination (LCC): El objetivo global común es una combinación lineal convexa de K criterios.

   $f = \sum_{i=1}^{K} \lambda_i f^i$ donde $\sum_{i=1}^{K} \lambda_i = 1$.

   Como se puede observar, esta forma se asemeja a la forma en la que se trata a la función objetivo en la programación tradicional de un solo criterio con ponderación de sus trabajos. Este tratamiento fue propuesto por primera vez en (Baker and Smith, 2003)

2. Épsilon-constraint (Ɛ constraint): Para los problemas MASPs, (Agnetis, Mirchandani, Pacciarelli & Pacifici, 2004) propusieron una optimización de una función objetivo siendo los otros criterios limitados por algunos factores. Fue la primera definición de este tipo de tratamiento y hasta día de hoy es la más utilizada. Consiste

fundamentalmente en minimizar una función objetivo de un conjunto de trabajos y establecer unos límites superiores que no pueden ser superados para el resto de funciones objetivos del resto de agentes.

Se denota por f=Ɛ (f^1/f^2, …, f^K) siendo el objetivo global minimizar f^1 sujeto a $f^k \leq \varepsilon^k$, siendo $\varepsilon^k \geq 0$ para k =2,…, K.

En resumen, el modelo base para un problema cualquiera con dos conjuntos de trabajos sería el siguiente:

$$\text{Min } f^A$$
$$\text{Sa:}$$
$$f^B \leq \varepsilon^B$$

Ilustración 2. Modelo Épsilon-constraint

3. Pareto: Es la menos utilizada y hace uso del frente de Pareto. Es decir, en una optimización multiobjetivo se define como Pareto-óptimo si se tiene una solución σ_1 y no existe otra solución σ_2 tal que mejore en un objetivo sin empeorar al menos en uno de los otros. Estos conceptos surgen de los estudios de Wilfried Pareto (1848-1923) que definió una situación como eficiente siempre que: "No existe un posible que beneficie a una persona sin perjudicar a otra". Por tanto, la frontera de Pareto se define como el conjunto de soluciones no dominadas en el espacio.

   Un Ejemplo gráfico se puede observar en la figura 3.

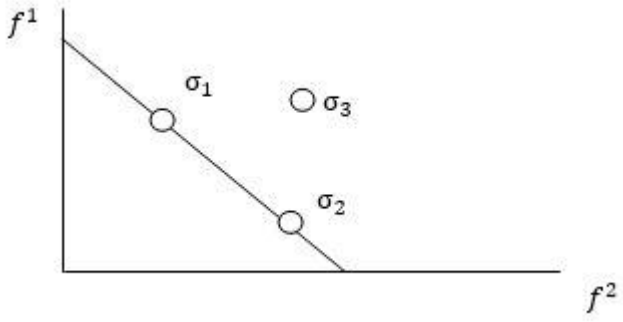

Ilustración 3. Ejemplo gráfico Frente de Pareto.

Las soluciones σ_1 y σ_2 son Pareto óptimas, sin embargo, σ_3 no es óptima, puesto que puede ser mejorada. Por lo tanto, se dice que σ_1 y σ_2 son soluciones no dominadas y σ_3 dominada.

La literatura existente sobre el problema estudiado en este proyecto denota a la función objetivo usada por Pareto como f=#(f^1, …, f^K).

Un programa σ es un óptimo de Pareto débil si y solo si ∄ σ' tal que para todo k, $f^k(\sigma') < f^k(\sigma)$ y es óptimo de Pareto estricto si y solo si ∄ σ' tal que para todo k, $f^k(\sigma') \leq f^k(\sigma)$.

Existen también otros métodos menos usados que no se verán en este trabajo. Éstos son los siguientes:

-Goal Programming approach: Se denota por GP(f^1, ..., f^K). Es el caso especial de Ɛ ($f/f^1, \dots, f^K$) con f constante. El objetivo aquí es encontrar una secuencia factible σ que satisfaga $f^k < Ɛ^k$ para k=1,…K.

-Lexicographical approach: Se denota por Lex(f^1, ..., f^K) y minimiza todos los criterios según el orden dado.

Este proyecto se centrará fundamentalmente en el estudio de un MASP tratado como épsilon-constraint en el cual se desea minimizar el makespan del primer agente, estando la tardanza total del segundo agente limitada por épsilon.

CAPÍTULO 3.

REVISIÓN DE LA LITERATURA.

3.1. INTRODUCCIÓN

Existen pocos estudios acerca del problema de programación con varios conjuntos de trabajos o problemas de programación de la producción multi-agente (MASPs). Las primeras menciones a este tipo de problemas aparecen en (Baker & Smith, 2003) y (Agnetis et al., 2004). A partir de ellos se han realizado varias investigaciones acerca de estos modelos, estando centradas en el estudio de modelos donde todos los trabajos tienen los mismos objetivos. Sin embargo, como se ha explicado en apartados anteriores, existe una gran cantidad de aplicaciones en el mundo industrial donde no todos los trabajos están sujetos a las mismas limitaciones y objetivos, por lo que recientemente se está elevando el número de estudios acerca de este tipo de problemas.

Se ha podido constatar una falta de unificación de criterios tanto en la forma de denotar a este tipo de problemas como en la notación y la manera de resolver dichos modelos, ya que cada investigación se ha realizado sin tener en cuenta los avances que se hayan podido tener en otras. En (Perez-Gonzalez & Framinan, 2014) se pretende crear un marco común a partir de las contribuciones existentes en torno a este tipo de problemas, así como una definición de las posibles vías de investigación futuras para mejorar los estudios realizados.

En los apartados posteriores de este proyecto se hará una revisión de los diferentes estudios realizados a lo largo de los últimos años sobre la programación multi-agente en entornos de Flow Shop de permutación, que es denominado tal y como se indicó en apartados anteriores como “multi-agent scheduling problem” (MASP). Se ha escogido esta denominación porque así es indicado en (Perez-Gonzalez & Framinan, 2014), donde se acuerda que es el nombre más usado y conveniente para este tipo de problemas, a pesar de que en la literatura aparecen otros nombres como “interfering jobs”, “multi-agent scheduling” o “mixed-criteria”.

3.2. PROBLEMAS MULTI-AGENTES DE FLOWSHOP DE PERMUTACIÓN

En este apartado se hará una revisión de la literatura del problema estudiado en este proyecto, haciendo un estudio de todos los artículos existentes hasta el día de hoy. Anteriormente se ha mencionado la contribución de (Perez-Gonzalez & Framinan, 2014), donde se hace una unificación de notación para los MASPs. Cabe destacar que en este

proyecto se usa el nombre y notación dados en (Perez-Gonzalez & Framinan, 2014) para contribuir a dicha unificación.

Para la realización de esta revisión se ha investigado en todos los artículos existentes sobre problemas de Flow shop de permutación multi agente. Dicha búsqueda no es tan simple como pueda parecer, puesto que este tipo de problemas ha sido poco estudiado en comparación con un problema tradicional de un solo criterio o problemas multicriterio pero con un solo conjunto de trabajos. En (Cheng, Ng, & Yuan, 2006) ya se estudió un problema con varios conjuntos de trabajos en una sola máquina donde se deseaba minimizar los trabajos tardíos de cada agente. Se tuvo que esperar varios años para que el estudio se extendiera a entornos con más de una máquina.

El problema de Flow shop es el más estudiado en este tipo de problemas, sin embargo, no todas las investigaciones tienen en cuenta la restricción de la permutación, por lo que este apartado se centrará especialmente en dichos artículos que sí lo hacen.

Este proyecto se centra en MASPs con dos conjuntos de trabajos disjuntos puesto que los agentes que tienen trabajos en común no han sido estudiados en profundidad en la literatura existente actual. Con respecto a este tipo de problemas solo se ha podido encontrar un estudio haciendo uso del frente de Pareto en (Ben Ltayef, Loukil, & Teghem, 2009), en donde se puede observar un estudio experimental sobre un Flow shop de permutación con dos máquinas intentando minimizar la tardanza máxima o el makespan. Los otros dos tipos de tratamientos de las funciones objetivo (Linear Convex Combination y Épsilon-constraint) no tienen ninguna aportación en la literatura con respecto a conjuntos de trabajos no disjuntos.

Con respecto a conjuntos de trabajos disjuntos, el objeto de estudio de este proyecto, se pueden encontrar algunos artículos que lo estudian en un entorno de Flow shop de permutación.

A continuación, se va a realizar un resumen de los diferentes artículos encontrados en la literatura haciendo uso de dos de los tres tipos de tratamiento de las funciones objetivos estudiados, ya que cabe destacar que no se ha encontrado ninguna investigación en la literatura actual que trate sobre el tratamiento de los distintos criterios de los agentes haciendo uso del frente de Pareto.

3.2.1. LINEAR CONVEX COMBINATION

Este tipo de tratamiento de las funciones objetivos se puede encontrar por primera vez en (Khelifati & Benbouzid-Sitayeb, 2011). Este artículo está destinado a los entornos de producción donde se debe realizar una programación de las tareas de producción junto a las de mantenimiento preventivo, algo muy común en un entorno de producción tradicional. En resumen, los trabajos de producción en las máquinas pueden verse interferidos por los trabajos de mantenimiento preventivo (y en menor medida las operaciones de mantenimiento correctivo) a la hora de programarlos, ya que no pueden ser realizados de

forma simultánea, es decir, para realizar una tarea de mantenimiento preventivo el recurso debe estar liberado.

Los dos conjuntos de trabajos tenidos en cuenta en este MASP serán por un lado el conjunto de tareas de producción y las tareas de mantenimiento preventivo por otro. Por tanto, el problema a tratar es de un entorno de flowshop de m máquinas con dos conjuntos de trabajos con la restricción de permutación.

Cada una de las m máquinas tienen que ser revisadas periódicamente según unos intervalos de tiempo, es decir, cada operación de mantenimiento j se realiza en cada máquina i con un tiempo dado cada T períodos. Además, esta periodicidad T tiene un intervalo de tolerancia $[Tmin_{ij}Tmax_{ij}]$, que se ve reflejado gráficamente en la figura 4.

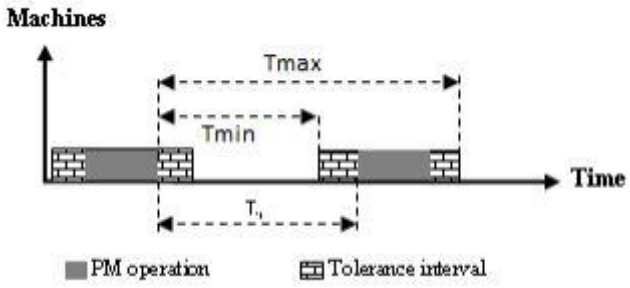

Ilustración 4. Explicación gráfica mantenimiento preventivo. Fuente: (Khelifati & Benbouzid-Sitayeb, 2011)

Las variables de mantenimiento son nombradas con una comilla. Lo ideal es que las tareas de mantenimiento se programen dentro del intervalo, sin embargo, es posible que se programen con adelanto o con retraso, originando dos nuevos tipos de variables (Adelantos y retrasos $E`_{ijk}$ y $L`_{ijk}$ respectivamente), donde el subíndice k indica la k^{th} ocurrencia de la operación de mantenimiento j en la máquina i. También hay operaciones de mantenimiento correctivo, que son realizadas cuando se produce una avería en una máquina. Estas operaciones se tratan de la misma manera que las de mantenimiento preventivo exceptuando que no tienen intervalo de tolerancia. Para darle una notación a las tareas de mantenimiento se sugiere la variable M_{ij} , que representa la operación de mantenimiento j en la máquina i.

Las funciones objetivos tratadas en este artículo son las siguientes:

1. Minimizar el makespan de las operaciones de producción (agente A):

   Minimizar f^{A}=max $\{C_j^{\mathrm{A}}\}$, jϵJ^{A}

2. Minimizar los adelantos y tardanzas de las operaciones de mantenimiento (agente B):

   Minimizar $f^{\mathrm{B}}=\sum_{i=1}^{m}\sum_{j=1}^{m_i}\sum_{k=!}^{m_{ij}} E`_{ijk} + L`_{ijk}$, j$\epsilon J^{\mathrm{B}}$ Donde m_i representa el número de operaciones de mantenimiento en la maquina i y m_{ij} representa el número de ocurrencias de la operación de mantenimiento M_{ij}.

En este artículo se hace un Linear Convex Combination puro, es decir, sin ponderar las funciones objetivos, dándole la misma importancia a ambos conjuntos de trabajos. Por tanto, la función objetivo global será la suma de ambos criterios explicados anteriormente:

$$f = f^1 + f^2$$

En definitiva, la descripción del problema estudiado es el siguiente:

$$F2\left|\ \text{prmu}\ \right| C^A_{max} + \sum_{i=1}^{m} \sum_{j=1}^{m_i} \sum_{k=!}^{m_{ij}} E`_{ijk} + L`_{ijk}$$

Para resolver dicho modelo se propone un proceso de programación dinámico basado en un sistema multiagente, que es definido como un sistema compuesto por múltiples agentes inteligentes que interactúan entre ellos. Hay 5 tipos de agentes:

– Agente supervisor de la producción (Prod-Sup-Agent);
– Agente supervisor del mantenimiento por máquina (Maint-Sup-Agent);
– Agente de máquinas por máquina. (Machine-Agent);
– Agentes de mantenimiento (Maint-Agent);
– Agentes de producción (Prod-Agent).

La siguiente aportación en forma de investigación se encuentra en (Luo, Chen & Zhang, 2012), en donde se hace uso de un entorno de Flow shop con dos máquinas y dos conjuntos de trabajos. El mismo problema con un solo agente puede ser resuelto de manera exacta en un tiempo polinomial, sin embargo, el modelo estudiado en este artículo se ha demostrado que es NP-hard y resuelto aproximadamente en un tiempo pseudo-polinomial.

Las funciones objetivos de ambos conjuntos de trabajos es el makespan y se propone un FPTAS (fully polynomial time approximation scheme) para resolverlo:

Minimize f^{A}=max $\{C_j^{\mathrm{A}}\}$, j$\in J^{\mathrm{A}}$
Minimize f^{B}= max $\{C_j^{\mathrm{B}}\}$, j$\in J^{\mathrm{B}}$

El modelo según la notación de (Graham et al., 1979) es el siguiente:

$$F2\left|\ \text{prmu}\ \right| C^A_{max} + \theta C^B_{max} \text{ siendo } \theta \geq 1$$

Sin embargo, en este artículo también hace uso de un modelo de optimización con restricciones (Épsilon-constraint) como se verá en el apartado siguiente.

La siguiente aportación a la optimización haciendo uso de la suma ponderada de los criterios se encuentra en (Lei, 2015). En este artículo se estudia el Flow shop con dos conjuntos de trabajos, pero no hace referencia a la restricción de permutación. Para el primer conjunto de trabajos se desea minimizar el makespan y para el segundo el total Tardiness:

Minimize f^{A}=max $\{C_j^{\mathrm{A}}\}$, j$\in J^{\mathrm{A}}$
Minimize f^{B}=$\sum_{j\in J^B} \max\{C_j^{\mathrm{B}} - d_j^{\mathrm{B}}, 0\}$, j$\in J^{\mathrm{B}}$

Por lo tanto, el modelo es el siguiente:

$$F2\left|\left|C_{max}^{A}+\theta\sum T_j^{B}\right.\right. \text{ siendo } \theta\geq 1$$

En este artículo se propone un algoritmo simple de búsqueda de vecindad variable (VNS, simple variable neighborhood search). VNS es un algoritmo de búsqueda local basado en las metaheurísticas y consiste en la búsqueda de una solución local óptima explorando los vecinos de una solución dada.

Al igual que en (Luo et al., 2012), en este artículo también utilizan el modelo de factibilidad con restricciones (Épsilon-constraint).

En 2016 se publica otro trabajo de investigación (Fan & Cheng, 2016) acerca de este tipo de problemas haciendo uso del Linear Convex Combination. En este artículo se estudia un entorno de Flowshop con dos máquinas y dos agentes sin tener en cuenta la restricción de la permutación y se tienen en cuenta dos problemas que ambos son tratados con Linear Convex Combination (LCC). Ambos son NP-hard:

1. Suma ponderada del makespan de ambos conjuntos de trabajos.

   Minimizar f^{A}=max $\{C_j^{\mathrm{A}}\}$, jϵJ^{A}
   Minimizar f^{B}= max $\{C_j^{\mathrm{B}}\}$, jϵJ^{B}

   El modelo según la notación de (Graham et al., 1979) es el siguiente:

   $$F2\left|\left|C_{max}^{A}+\theta C_{max}^{B}\right.\right. \text{ siendo } \theta\geq 1$$

   Para resolver este problema se propone un algoritmo de tiempo pseudopolinomial. Además, se menciona un algoritmo basado en la regla de Johnson, un algoritmo heurístico utilizado generalmente para la programación de tareas que tienen que ser procesadas en dos máquinas. Su principal objetivo es minimizar el makespan. La regla de Johnson basa su funcionamiento en el siguiente algoritmo:

   1. Se anota el tiempo de operación de cada trabajo en ambas máquinas.
   2. Se elige el tiempo más breve entre todos los tiempos.
   3. Si el tiempo breve del trabajo escogido es para la primera máquina, se asigna dicho trabajo en la primera posición de la secuencia y si es para la segunda máquina, se asigna dicho trabajo en la última posición de la secuencia. En caso de empate se hace el trabajo en la primera máquina.
   4. Repetir los pasos 2 y 3 con los restantes trabajos hasta completar la programación.

2. Suma ponderada del total completion time de un conjunto de trabajos y el makespan del otro:

   Minimizar f^{A}=$\sum C_{ij}^{A}$, jϵJ^{A}
   Minimizar f^{B}= max $\{C_j^{\mathrm{B}}\}$, jϵJ^{B}

El modelo según la notación de (Graham et al., 1979) es el siguiente:

$$F2\left|\right| \sum C_{ij}^{A} + \theta C_{max}^{B} \text{ siendo } \theta \geq 1$$

Para resolver dicho modelo se propone un algoritmo de aproximación basado en una relajación lineal de la programación.

Además, se aportan unos algoritmos simples que son capaces de resolver algunas instancias de ambos problemas.

3.2.2. ÉPSILON-CONSTRAINT

Como se ha explicado en apartados anteriores, Épsilon-Constraint es una forma de tratamiento de las funciones objetivo en los MASPs basada en minimizar una función objetivo de un conjunto de trabajos y establecer unos límites superiores que no pueden ser superados para el resto de funciones objetivos del resto de agentes. Precisamente esta relajación del problema hace que sea más fácil de realizar, por lo que este tipo de tratamiento es el más utilizado en la literatura actual.

La primera aportación a este tipo de problema lo encontramos en (Lee, Chen, Chen & Wu 2011), donde se estudia el Flow shop con dos conjuntos de trabajos en un entorno con dos máquinas sin la restricción de la permutación.

En este artículo se estudian dos problemas con un criterio diferente a minimizar en la función objetivo global. El primer conjunto de trabajos aparecerá en la función objetivo del modelo general y se desea minimizar el total completion time para un problema y la tardanza en otro. Sin embargo, el segundo conjunto de trabajos aparece en las restricciones en forma de límite superior que no debe ser superado. Para este segundo criterio y para los dos problemas estudiados en este artículo se desea no tener trabajos con tardanza, por lo que el número de trabajos que son procesados más tarde de sus fechas de entrega no debe superar un cierto límite superior.

Por tanto, aparece una nueva variable no definida hasta ahora, U_j, que se define como:

$$U_j = 1, \text{ si } T_j > 0 \text{ y } U_j = 0 \text{ en otro caso.}$$

Por tanto, los dos modelos estudiados se describen como a continuación:

$$-F2\left|\right|\varepsilon(\sum C_j^A / \sum U_j^B)$$

$$-F2\left|\right| \varepsilon(\sum T_j^A, \sum U_j^B)$$

Para ambos problemas se desarrolla un branch and bound (con tiempos coherentes para instancias de hasta 20 trabajos). Además, se aportan algoritmos heurísticos de recocido simulado, en donde en cada iteración se evalúan algunos vecinos del estado actual y se decide si efectuar una transición a un nuevo estado o no. Dicho nombre proviene del proceso

de recocido del acero, ya que éste se calienta para posteriormente enfriar el material y variar sus propiedades físicas, ya que el calor provoca que los átomos aumenten su energía y se desplacen hacia un nuevo estado con unas posiciones diferentes a las iniciales.

Posteriormente, en (Luo et al., 2012), se desarrolla un modelo de optimización con restricciones basado en un entorno de flowshop de dos máquinas y dos conjuntos de trabajos con la restricción de permutación. Este artículo ya apareció en el apartado anterior de Linear Convex Combination y es que en este artículo se hace una doble visión del MASP, es decir, aporta dos tipos de modelos para ser optimizados:

-Modelo Linear Convex Combination: F2 | prmu | C^A_{max}+θC^B_{max} siendo θ≥1, que ya fue explicado en el apartado anterior.

-Modelo de optimización con restricciones (Épsilon-Constraint): En este modelo se intenta optimizar (minimizar) el makespan del primer conjunto de trabajos, mientras que tiempo total de finalización del segundo agente está sujeto a un límite superior que no puede ser superado, por lo que en la función objetivo global solo aparecerá el primer agente y el makespan del segundo conjunto de trabajos estará presente en las restricciones del modelo. El problema es NP-hard:

Minimizar $f^A = C^A_{max}$ sujeto a $f^B = C^B_{max} \leq$Q siendo Q>0

Por lo tanto, el modelo según la notación de (Graham et al., 1979) es el siguiente:

$$F2 \mid prmu \mid \varepsilon\,(C^A_{max}/C^B_{max}) \text{ siendo } \varepsilon^B = Q$$

Para resolver este modelo, al igual que en el caso del Linear Convex Combination, se propone un FPTAS (fully polynomial time approximation scheme). Sin embargo, este algoritmo es capaz de resolver dicho problema siempre y cuando se relaje la restricción del segundo agente en un factor Ɛ, es decir:

$$f^B = C^B_{max} \leq (1+\varepsilon)Q$$

El diseño de un algoritmo para resolver dicho modelo sin relajar la restricción es todavía un desafío.

En (Mor & Mosheiov, 2014) se estudia un flowshop de dos agentes y m máquinas resueltos con algoritmos de tiempo polinomial. Además, es un "proporcionate flowshop", esto es, se asume que el tiempo de proceso de cada trabajo j es el mismo para todas las máquinas:

$$p^k_{ij} = p^k_j \quad i=1,\ldots,.\, m.$$

En este artículo se estudian tres problemas diferentes con diferentes funciones objetivo para el primer agente. Además, se demuestra que los problemas "proporcionate flowshop" analizados en este artículo pueden ser resueltos en un tiempo polinomial, a pesar de que el problema básico de dos agentes en un entorno de flowshop para las funciones objetivos del makespan para ambos agentes se demostró ser NP-hard (Agnetis et al., 2004).

-El primer problema analizado incluye una nueva variable no descrita hasta el momento en la cual se tiene en cuenta el coste de finalización de los trabajos. Dicha función objetivo se denomina "máximo coste de los trabajos". El coste de cada trabajo depende de una función de coste específica f_j^k que depende de su completion time. Para el segundo agente se establece un límite superior para el coste máximo. Por tanto, el modelo será el siguiente:

$$Fm\left|\right|\Sigma(f_{max}^A / f_{max}^B)$$

-En el segundo problema se desea minimizar el total completion time del primer agente, apareciendo el segundo criterio en las restricciones en forma de límite superior para el máximo coste permitido de finalización de los trabajos:

$$Fm\left|\right|\Sigma\ (\sum C_j^A / f_{max}^B)$$

-Por último, el tercer problema analizado minimiza en la función objetivo global del modelo el número de trabajos tardíos del primer agente y al igual que en los dos casos anteriores, el segundo agente aparece en las restricciones con un límite superior del máximo coste de finalización de los trabajos:

$Fm\left|\right|\Sigma(\sum U_j^A / f_{max}^Y)$ siendo $U_j^k = 0\ si\ C_j^k \leq d_j^k$ y $U_j^k = 1$ en otro caso.

Al igual que en el caso anterior, se encuentra otro artículo donde se hace una visión doble del MASP, por lo que en (Lei, 2015) se pueden encontrar dos tipos de modelos (Linear Convex Combination y Épsilon-constraint). Para ambos se propone un algoritmo simple de búsqueda de vecindad variable (VNS, simple variable neighborhood search) y no se tiene en cuenta la restricción de permutación:

-Modelo Linear Convex Combination: $F2\left|\right| C_{max}^A + \theta \sum T_j^B$ siendo $\theta \geq 1$.

-Modelo Épsilon-constraint: Se optimiza el makespan del primer agente $f^A = C_{max}^A$, mientras que la Tardanza Total del segundo conjunto de trabajos $f^B = \sum T_j^B$ está sujeto a una restricción basada en un límite superior que no puede ser superada:

Minimizar $f^A = C_{max}^A$ sujeto a $f^B = \sum T_j^B \leq Q$ siendo Q>0.

El modelo según la notación de (Graham et al., 1979) es el siguiente:

$$F2\left|\right| \Sigma\ (C_{max}^A / \sum T_j^B\ ) \text{ siendo } \Sigma^B = Q$$

También en 2015 se lanza otro artículo de un MASP (Shiau, Tsai, Lee & Cheng, 2015). En él, se estudia un entorno de flowshop con permutación con dos máquinas. Además, surge un nuevo concepto no tenido en cuenta hasta entonces, los efectos de aprendizaje. En los problemas clásicos de programación se asume que todos los tiempos de procesos son fijados desde el inicio del proceso hasta el final, sin embargo, en muchas situaciones realistas los empleados pueden procesar las tareas más eficientemente a lo largo del tiempo debido a que

van acumulando experiencia. Esta situación es conocida como efecto del aprendizaje en la literatura de la Programación de Operaciones y tiene cabida cuando el recurso que procesa los trabajos son empleados y no máquinas como tal.

En este artículo se denomina a_{jr} al tiempo de procesado del trabajo j situado en la posición r^{th} en la primera máquina. Es decir, su expresión matemática es la siguiente:

a_{jr}=$a_j \prod_{k=0}^{r-1} l_k$, siendo l_0=1 y 0<l_k≤1 para k=1,…,n.

Como se puede observar en la expresión anterior, se tienen unos datos de partida a_j para todos los trabajos y unos coeficientes l_k para todas las posiciones en la secuencia, que irán multiplicándose a los tiempos de procesados iniciales a medida que van siendo procesados los trabajos, permitiendo que se reduzca el tiempo de cada trabajo en cada recurso debido a la acumulación de experiencia de los operarios. Para el trabajo procesado en la primera posición de la secuencia el coeficiente a multiplicar es igual a 1, debido a que no se ha acumulado aprendizaje.

Para la segunda máquina la explicación es la misma, y se define como b_{jr} al tiempo de procesado del trabajo j situado en la posición r^{th} en dicha máquina. Su expresión matemática es la siguiente:

b_{jr}=$b_j \prod_{k=0}^{r-1} l_k$, siendo l_0=1 y 0<l_k≤1 para k=1,…,n.

El problema NP-hard estudiado consiste en minimizar el total completion time de los trabajos de un conjunto de trabajos mientras que la tardanza máxima del otro no puede exceder un límite, es decir:

F2 $\left|$prmu $\right|$ Ɛ ($\sum C_j^A / T_{max}^B$) siendo Ɛ^B=Q

Para resolver dicho problema se aporta un Branch and Bound cuyas pruebas experimentales han permitido saber que puede resolver instancias de hasta veinte trabajos en menos de una hora. Además, se proponen algoritmos genéticos para obtener soluciones aproximadas a la óptima.

Por último, en (Ahmadi-Darani, Moslehi & Reisi-Nafchi, 2018) se encuentra la última contribución de la literatura acerca de problemas MASPs. En dicho artículo se hace un estudio de un entorno de Flowshop con dos máquinas y dos agentes sin la restricción de la permutación.

En dicho modelo NP-hard se desea minimizar la tardanza total del primer agente y que el makespan del segundo conjunto de trabajos no supere un límite superior. Por tanto, el modelo según la notación de (Graham et al., 1979) es la siguiente:

F2$\left|\right|$ Ɛ($\sum T_j^A / C_{max}^B$), es decir C_{max}^B< Ɛ

Para resolver algunas instancias de este problema se propone el algoritmo de Branch and Bound, pero se comprobó que no era eficiente y que solo podía resolver instancias de hasta 16 trabajos, por lo que se optó por un algoritmo de búsqueda tabú, un algoritmo metaheurístico, que mediante el uso de las estructuras de memoria (a corto y a largo plazo)

puede determinar una solución como visitada (se le marca como tabú), de forma que el algoritmo no vuelva a visitar dicha solución.

3.3. CONCLUSIONES

Una vez desarrollado los diferentes artículos existentes sobre el problema MASP se observa un mismo patrón en todos ellos.

Se ha podido constatar durante el estudio de estas investigaciones la importancia de este tipo de problemas y su gran aplicación práctica que tiene. El tratamiento con talleres de Flow Shop y varios conjuntos de trabajos con diferentes funciones objetivos es muy común en el día a día de las empresas de diferentes sectores, como se vio en el apartado 1.2, por lo que dicha afirmación justifica un estudio a fondo de este tipo de problemas.

Se ha observado que en la mayoría de los artículos encontrados se estudia un entorno de Flow shop debido a que es posiblemente el tipo de entorno más usado en la programación de Operaciones y como consecuencia, el que ha sufrido más investigaciones para su posible desarrollo. Sin embargo, en problemas multi criterio con varios conjuntos de trabajos la mayoría de las aportaciones tienen en cuenta un taller con dos máquinas y dos conjuntos de trabajos, siendo el tiempo total de finalización el criterio más usado, como se puede ver en la tabla 2. Además, en algunos de los artículos no se tiene en cuenta la restricción de permutación. Todas estas simplezas son debido a que todavía las investigaciones están en fase de experimentación y se han encontrado pocos avances en este tipo de problemas, por lo que es muy posible que en un futuro habrá avances con respecto a estos modelos debido a la gran carga práctica que tienen en la vida real de las empresas.

Artículo	Modelos
Khelifati & Benbouzid-Sitayeb, 2011	F2| prmu $\left\|C^A_{max}+\sum_{i=1}^{m}\sum_{j=1}^{m_i}\sum_{k=!}^{m_{ij}} E`_{ijk} + L`_{ijk}\right.$
Lee et al., 2011	F2$\left\|\right\|\varepsilon(\sum C^A_j / \sum U^B_j)$
	F2$\left\|\right\|\varepsilon(\sum T^A_j, \sum U^B_j)$
Luo et al., 2012	F2 $\left\|\right.$ prmu $\left\|C^A_{max}+\theta C^B_{max}\right.$
	F2 $\left\|\right.$prmu $\left\|\right.$ ε (C^A_{max}/C^B_{max})
Mor & Mosheiov, 2014	Fm$\left\|\right\|$ ε(f^A_{max} / f^B_{max})
	Fm $\left\|\right\|$ ε$(\sum C^A_j / f^B_{max})$
	Fm$\left\|\right\|$ε $(\sum U^A_j / f^Y_{max})$
Lei, 2015	F2$\left\|\right\|$ $C^A_{max}+\theta\sum T^B_j$
	F2$\left\|\right\|$ε $(C^A_{max}/\sum T^B_j)$
Shiau et al., 2015	F2$\left\|\right.$prmu $\left\|\right.$ε $(\sum C^A_j / T^B_{max})$
Fan & Cheng, 2016	F2$\left\|\right\|$ $C^A_{max}+\theta C^B_{max}$
	F2$\left\|\right\|\sum C^A_{ij}+\theta C^B_{max}$
Ahmadi-Darani et al., 2018	F2$\left\|\right\|\varepsilon(\sum T^A_j / C^B_{max})$

Tabla 1. Modelos usados por cada artículo

Como se mencionó en el apartado 2.2, se ha constatado una falta de unificación de criterio a la hora de desarrollar este tipo de problemas, y es que, a pesar de que existen algunas similitudes en la notación, en la mayoría de ellos se usa un lenguaje diferente, haciendo muy complicado crear un punto de unión entre ellos. Este aspecto se ve claramente reflejado en el nombre que recibe este tipo de trabajos en los diferentes artículos, siendo diferentes en la mayoría de ellos. Por ejemplo, podemos observar que en (Agnetis et al., 2004) el nombre que recibe el problema estudiado en este proyecto es "Competing users" o simplemente "Customers" en el caso de (Baker and Smith, 2003).

La estructura de todos los artículos encontrados es similar:

1. Breve descripción de la investigación realizada seguido de una introducción del tema de estudio (se ha encontrado un mismo patrón en todos los artículos, en el cuál estas introducciones son muy similares entre ellas, dando un repaso histórico de cómo ha evolucionado estos estudios a lo largo de los últimos años).

2. Explicación de la notación utilizada.
3. Desarrollo de los modelos propuestos.
4. Algoritmo propuesto para resolver dichos modelos.
5. Resultados experimentales y conclusiones.

Además, el estudio hasta el día de hoy de dicho problema se ha centrado fundamentalmente en dos maneras de tratar las funciones objetivos:

-Épsilon-Constraint (La más utilizada).
-Linear Convex Combination.

Cada uno de los artículos centra gran parte de su desarrollo en explicar los algoritmos propuestos por cada uno de ellos para resolver dichos problemas. Un resumen de ello se puede observar en la tabla 2.

Artículo	**Criterios**	**Algoritmos**
Khelifati & Benbouzid-Sitayeb, 2011	Linear Convex Combination	Programación dinámica en sistema multiagente
Lee et al., 2011	Épsilon-constraint	Branch and Bound
		Algoritmo de recocido simulado
Luo et al., 2012	Linear Convex Combination	FPTAS
	Épsilon-constraint	FPTAS
Mor & Mosheiov, 2014	Épsilon-constraint	Algoritmo de tiempo polinomial
Lei, 2015	Linear Convex Combination	VNS
	Épsilon-constraint	VNS
Shiau et al., 2015	Épsilon-constraint	Branch and Bound
Fan & Cheng, 2016	Linear Convex Combination	Algoritmo de tiempo pseupolinomial
		Algoritmo heurístico basado en Regla Johnson
		Algoritmos simples
Ahmadi-Darani et al., 2018	Épsilon-constraint	Algoritmo de búsqueda tabú

Tabla 2. Algoritmos usados por cada artículo.

En definitiva, se puede concluir que en un futuro seguirán desarrollándose nuevos estudios y artículos sobre este tipo de problemas, pues las líneas de investigación siguen abiertas y expandiéndose hacia nuevos algoritmos y formas de tratamiento de los diferentes criterios existentes.

CAPÍTULO 4

MODELOS DE PROGRAMACIÓN LINEAL ENTERA.

4.1. INTRODUCCIÓN

El objetivo de este capítulo es el de programar diversos modelos en lenguaje C para su posterior resolución de forma exacta con el solver de optimización Gurobi. El problema elegido es el de minimizar el makespan del primer conjunto de trabajos y la tardanza total del segundo conjunto.

Este proyecto ha analizado la situación de los estudios realizados en la literatura acerca de MASPs y se ha encontrado que en ninguno de ellos se hace una resolución de forma exacta de las instancias propuestas, tal y como se puede observar en el capítulo 3. Esto es porque generalmente este tipo de problemas necesitan de altos tiempos de resolución cuando las instancias son grandes, sin embargo, para instancias pequeñas, que también son tratadas en algunas empresas, puede ser útil resolverlas mediante algún método exacto sin incurrir en grandes cantidades de tiempo, tal y como se analizará en este capítulo. Otra aportación de interés es determinar hasta qué tamaño (número de trabajos totales) el solver devuelve la solución óptima en un tiempo de cómputo razonable. Por lo tanto, esta línea es de interés en este trabajo.

Para la realización de este capítulo se ha hecho uso de diversas referencias y modelos propuestos en diferentes artículos de la literatura: (Stafford, Tseng, & Gupta, 2005) y (Tseng & Stafford, 2008). En el primero de ellos se proponen varios modelos pertenecientes a dos familias diferentes:

-Familia Wagner: Son los modelos WST, Wilson y TS2.

-Familia Manne: A ella pertenecen los modelos denominados SGST y LYeq.

En (Tseng & Stafford, 2008) se proponen dos modelos nuevos haciendo uso de una combinación de las ecuaciones propuestas en los modelos anteriores para buscar una mejor eficiencia a la hora de resolver el problema. Los dos modelos propuestos en este artículo son llamados TBA y TS3. En este proyecto, para tener una visión amplia de los posibles resultados se han escogido los modelos más eficientes teniendo en cuenta los estudios realizados en (Ameneiros, 2018). En dicho trabajo se realizó una adaptación de dichos modelos a cuatro problemas tradicionales (un solo criterio) diferentes: Total completion time, total completion time ponderado, tardanza total y tardanza total ponderada. Posteriormente se hizo un análisis estadístico para evaluar la eficiencia de dichos modelos. En las figuras 5, 6, 7 y 8 extraídas de dicho estudio, se representan la media de tiempos de resolución de los modelos Total Completion Time, Total Completion Time Ponderado, Tardanza Total y Tardanza Total Ponderada respectivamente.

Como se puede observar en estas ilustraciones, el modelo TS2 es el más eficiente de la familia Wagner para todos los problemas estudiados, por lo que será el modelo de dicha familia escogido en este Proyecto para su programación. Con respecto a la familia Manne se puede realizar el mismo razonamiento, ya que se puede observar que los tiempos de cómputo son menores en el modelo SGST si son comparados con LYeq. Por último, el modelo TS3 es el escogido al ser más eficiente que el modelo TBA en la mayoría de los casos estudiados.

En definitiva, en este proyecto se ha realizado la adaptación y programación de los modelos TS2, SGST y TS3 debido a su mayor eficiencia con respecto al resto.

Los modelos clásicos encontrados en (Stafford et al., 2005) y (Tseng & Stafford, 2008) son usados para minimizar el makespan en un problema tradicional de programación con una sola función objetivo en un entorno de flowshop de permutación con N trabajos y M máquinas, por lo que se ha realizado una adaptación de dichos modelos para que representen un entorno con dos conjuntos de trabajos. Cabe destacar que la mayoría de las restricciones y variables usadas en dichos artículos son factibles para el trabajo estudiado en este proyecto.

El método escogido de tratamiento de las funciones objetivo es el de Épsilon-Constraint, por lo que el principal problema en este caso es el de encontrar el valor de épsilon para cada una de las instancias aportadas.

En conclusión, la estructura de los modelos en este proyecto buscará minimizar el makespan de los trabajos pertenecientes al primer agente sujeto a una serie de restricciones, incluida la cual que impide que la suma de tardanzas de los trabajos pertenecientes al segundo agente no pueda superar un cierto límite superior épsilon.

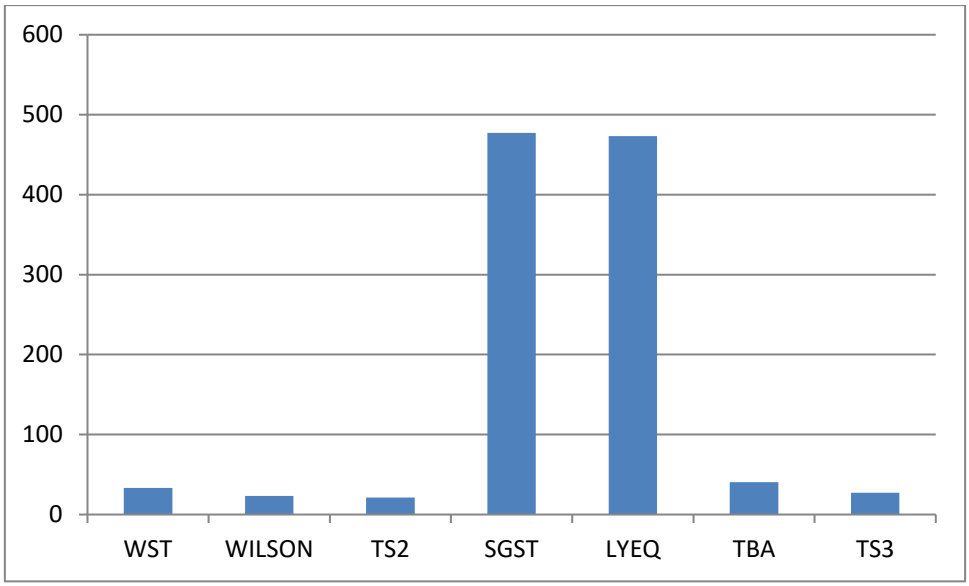

Ilustración 5. Media de tiempos de resolución de los modelos Total Completion Time. Fuente: (Ameneiros, 2018).

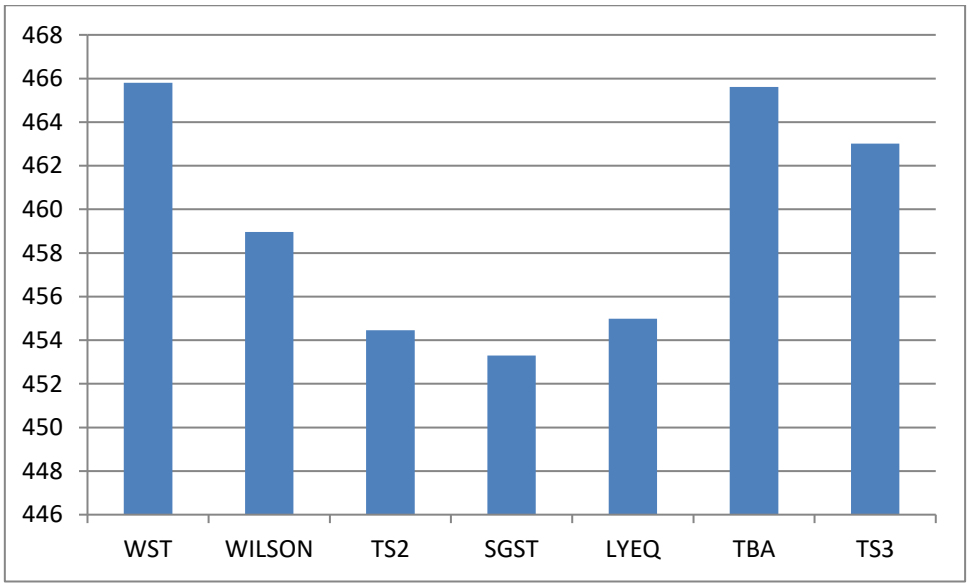

Ilustración 6.Media de tiempos de resolución de los modelos Total Completion Time Ponderado. Fuente: (Ameneiros, 2018).

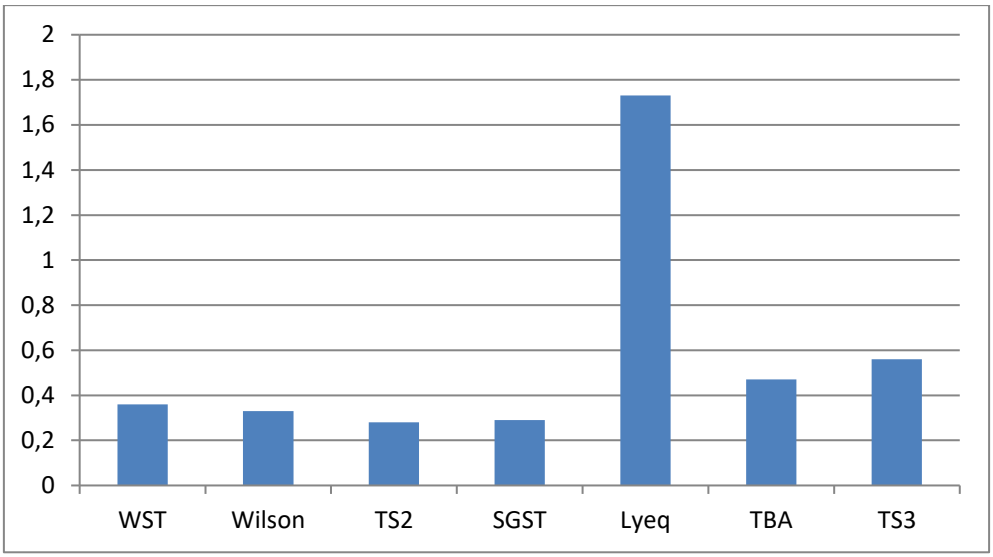

Ilustración 7. Media de tiempos de resolución de los modelos Tardanza Total. Fuente: (Ameneiros, 2018).

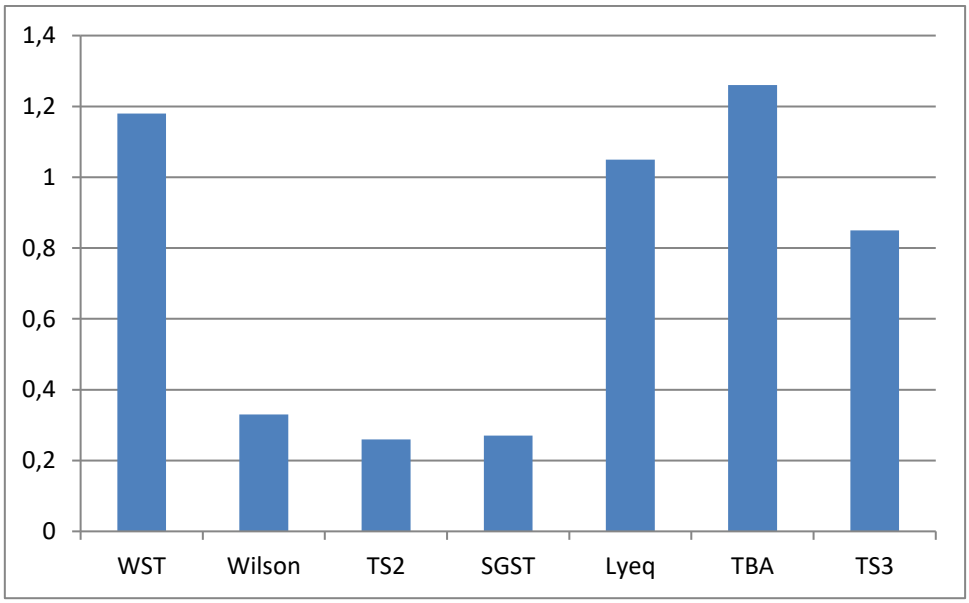

Ilustración 8. Media de tiempos de resolución de los modelos Tardanza Total Ponderada. Fuente: (Ameneiros, 2018).

4.2. NOTACIÓN

Para la realización de este capítulo, se ha escogido una notación ligeramente diferente a la notación utilizada en la literatura de este tipo de problemas multi agente reflejada en el apartado 2.4. Esto es debido a que se ha mantenido la notación utilizada en (Stafford et al., 2005) y (Tseng & Stafford, 2008), ya que sus modelos no estaban destinados a ser estudiados con varias funciones objetivo.

La principal diferencia radica en el uso de los índices para máquinas y trabajos ya que, a la hora de modelar, algunos de los modelos propuestos hacen una diferenciación de índices entre el número del trabajo y la posición dentro de la secuencia, tal y como se puede observar a continuación.

4.2.1. DATOS DE ENTRADA

Son los datos que se recibirán para realizar los modelos para cada instancia aportada. Son los siguientes:

M, número de máquinas.

N, número de trabajos.

n^A, número de trabajos del agente A.

n^B, número de trabajos del agente B.

J^A = Conjunto de trabajos A.

J^B = Conjunto de trabajos B.

T_{ri}=Tiempo de proceso del trabajo i en la máquina r. iϵJ^A, iϵJ^B

D_i=Fecha de entrega del trabajo i. iϵJ^B

P=Es un valor grande y se define como la suma de tiempos de proceso de todos los trabajos en todas las máquinas. P=$\sum_{r=1}^{M} \sum_{i=1}^{N} T_{ri.}$ iϵJ^A, iϵJ^B

$\mathcal{E}^B$= Épsilon. Es el valor límite el cual la función objetivo del segundo agente no puede superar.

4.2.2. ÍNDICES

Cómo se ha explicado anteriormente, los índices usados en este capítulo de modelaje cambian con respecto a la notación propuesta por (Perez-Gonzalez & Framinan, 2014). Los índices propuestos son los siguientes:

r,q para las máquinas. (1≤r≤ M) (1≤q≤ M)

i,k para los trabajos. (1≤i<k≤ N)

j,p para la posición en la secuencia. (1≤j≤ N) (1≤p≤ N)

4.2.3. VARIABLES

C_{ri}=Tiempo de finalización del trabajo i en la máquina r. (1≤r≤ M) (1≤i≤ N)

E_{rj}=Tiempo de finalización del trabajo en la posición j en la máquina r. (1≤r≤ M) (1≤j≤ N)

TT_{rj} = Tiempo de finalización del trabajo en la posición j en la máquina r. (1≤r≤ M) (1≤j≤ N)

F_j= Flowtime del trabajo en la posición j.

X_{rj}=Tiempo ocioso en la máquina r antes del comienzo del procesamiento del trabajo en la posición j de la secuencia. (1≤r≤ M) (1≤j≤ N)

Y_{rj}=Tiempo de espera del trabajo en la posición j de la secuencia después de ser procesado en la máquina r. (1≤r≤ M) (1≤j≤ N)

$D_{i,k}$= 1. El trabajo i es procesado antes que el trabajo k.
0. En otro caso.

Z_{ij} = 1. El trabajo i es procesado en la posición j.
0. En otro caso.

$Tard_j$=Tardanza del trabajo en la posición j de la secuencia. jϵJ^B

T_i=Tardanza del trabajo i. iϵJ^B

Como se puede observar, las variables asociadas a la tardanza de los trabajos están solo definidas para los trabajos pertenecientes al segundo agente ya que la función objetivo a minimizar con respecto al primer conjunto de trabajos es el makespan.

4.3. MODELOS DE PROGRAMACIÓN LINEAL

4.3.1. FAMILIA WAGNER. MODELO TS2.

Para comenzar, se ha desarrollado el modelo TS2 de la familia Wagner. Los modelos pertenecientes a estas familias utilizan variables con el índice j, es decir, los trabajos están referenciados según la posición que ocupan en la secuencia usada. Por tanto, la variable más importante en los modelos pertenecientes a esta familia es Z_{ij}, ya que permite pasar del índice j al índice i, permitiendo que el modelo pueda identificar el trabajo que está tratando.

Esta situación afecta a la hora de adaptar los modelos originales al problema multi agente que se está estudiando en este proyecto, como se podrá ver a continuación.

Modelo original

Este modelo usa las variables relacionadas con el tiempo de terminación de los trabajos en las máquinas, es decir, E_{rj}.

La función objetivo del modelo original es el makespan y, por tanto, con el uso de las variables mencionadas anteriormente, se define de manera trivial, puesto que la función objetivo será la fecha de finalización del último trabajo secuenciado en la última máquina, definida como E_{MN}.

El conjunto de restricciones (4.1) permite que cada trabajo i sea asignado a una sola posición en la secuencia, mientras que (4.2) prohíbe que cada posición sea ocupada por más de un trabajo. El conjunto de restricciones (4.3) propone que un trabajo no puede terminar en una máquina hasta que no se ha procesado el trabajo anterior y ha concluido el tiempo de proceso perteneciente a dicha máquina.

En (4.4) se indica que el procesado de un trabajo no pueda terminar en una máquina r+1 hasta que dicho trabajo no haya terminado en la máquina anterior r y se haya producido su tiempo de procesado en r+1. Por último, en (4.5) se puede observar la restricción que modela la situación en la cual el primer trabajo en la primera máquina no pueda terminar antes que su tiempo de proceso en dicha máquina.

Min E_{MN}

sa

$$1=\sum_{j=1}^{N} Z_{ij} \quad (1\leq i\leq N) \qquad (4.1)$$

$$1=\sum_{i=1}^{N} Z_{ij} \quad (1\leq j\leq N) \qquad (4.2)$$

$$E_{rj}+\sum_{i=1}^{N} T_{ri} Z_{i,j+1} \leq E_{r,j+1}$$

$$(1\leq j\leq N\text{-}1)\ (1\leq r\leq M) \qquad (4.3)$$

$$E_{rj}+\sum_{i=1}^{N} T_{r+1,i} Z_{i,j} \leq E_{r+1,j}$$

$$(1\leq j\leq N)\ (1\leq r\leq M\text{-}1) \qquad (4.4)$$

$$E_{11}\geq\sum_{i=1}^{N} T_{1,i} Z_{i,1} \qquad (4.5)$$

$$E_{rj} \geq 0\ (1\leq j\leq N)\quad (1\leq r\leq M)$$

$$Z_{ij}\in\{0,1\}\ (1\leq i\leq N)\ (1\leq j\leq N)$$

Modelo adaptado

Para el modelo adaptado al problema estudiado en este proyecto, se añaden una serie de restricciones, además de cambiar la función objetivo, que buscará minimizar una variable Cmax que representará el makespan del primer conjunto de trabajos.

Para la función objetivo del segundo agente, hay que incluir variables relacionadas con la tardanza, por lo que aparece la variable $Tard_j$, incluida en algunas restricciones como (4.6) y (4.7). La primera de ellas define cada una de las tardanzas de cada trabajo en todas las posiciones como la diferencia entre sus fechas de finalización en la última máquina y sus fechas de entrega. En (4.7) se obliga a que dicha variable sea mayor que cero, puesto que la tardanza se define siempre como un valor positivo, es decir, como el máximo entre cero y la diferencia entre la fecha de finalización en la última máquina y su fecha de entrega. En definitiva, si esta diferencia es negativa y el trabajo es finalizado con adelanto, la variable tardanza valdrá cero.

Como se explicó anteriormente, al estar el modelo original referenciado con variables con el índice j, hay que buscar una manera de identificar qué trabajos son los que están en cada posición de la secuencia. Para ello, se añaden nuevas restricciones de conexión-desconexión los conjuntos (4.10) y (4.11).

En (4.10) se calcula el valor de las tardanzas de cada trabajo i perteneciente al segundo agente (ya que solo interesan las tardanzas de este conjunto de trabajos), por lo que entra en juego la variable T_i. Para ello, se hace uso del valor P, una cantidad grande que activa y desactiva una porción de restricción en caso de que Z_{ij} sea 1 o 0. Es decir, si el trabajo i está en la posición j, dicha variable tiene el valor 1 y T_i será mayor o igual que $Tard_j$. El procedimiento a seguir con las fechas de finalización de los trabajos del primer agente en la última máquina es el mismo, como se puede observar en el conjunto de restricciones (4.11). Finalmente, en (4.9) se define el makespan del primer conjunto de trabajos, que debe ser superior a cualquier fecha de finalización de todos los trabajos de dicho agente en la última máquina.

Para finalizar, y común a todos los modelos épsilon-constraint, hay que añadir la restricción que haga referencia a que la función objetivo del agente B no puede superar un cierto valor superior, como se puede observar en (4.8).

Min C_{Max}

sa

$$\sum_{i=1}^{N} T_i \leq \mathcal{E}^B \ (i \epsilon J^B) \quad (4.8)$$

$$C_{Max} \geq C_{Mi} \ (i \epsilon J^A) \quad (4.9)$$

$$T_i + P(1 - Z_{ij}) \geq Tard_j \ (i \epsilon J^B) \ (1 \leq j \leq N) \quad (4.10)$$

$$C_{Mi} + P(1 - Z_{ij}) \geq E_{Mj} \ (i \epsilon J^A) \ (1 \leq j \leq N) \quad (4.11)$$

$$1 = \sum_{j=1}^{N} Z_{ij} \ (1 \leq i \leq N) \quad (4.1)$$

$$1 = \sum_{i=1}^{N} Z_{ij} \ (1 \leq j \leq N) \quad (4.2)$$

$$E_{rj} + \sum_{i=1}^{N} T_{ri} Z_{i,j+1} \leq E_{r,j+1}$$

$$(1 \leq j \leq N\text{-}1) \ (1 \leq r \leq M) \quad (4.3)$$

$$E_{rj} + \sum_{i=1}^{N} T_{r+1,i} Z_{i,j} \leq E_{r+1,j}$$

$$(1 \leq j \leq N) \ (1 \leq r \leq M\text{-}1) \quad (4.4)$$

$$E_{11} \geq \sum_{i=1}^{N} T_{1,i} Z_{i,1} \quad (4.5)$$

$$Tard_j \geq E_{Mj} - \sum_{i=1}^{N} D_i Z_{ij} \ (1 \leq j \leq N) \quad (4.6)$$

$$Tard_j \geq 0 \ (1 \leq j \leq N) \quad (4.7)$$

$$E_{rj} \geq 0 \ (1 \leq j \leq N) \ \ (1 \leq r \leq M)$$

$$Z_{ij} \epsilon \{0,1\} \ (1 \leq i \leq N) \ (1 \leq j \leq N)$$

$$T_i \geq 0 \ (i \epsilon J^B)$$

$$C_{Mi} \geq 0 \ (i \epsilon J^A)$$

$$C_{Max} \geq 0$$

4.3.2. FAMILIA MANNE. MODELO SGST

Esta familia de modelos presente en (Stafford et al., 2005) provoca un cambio a la hora de manejar las variables, puesto que dejan de estar referenciadas con el índice j y se empieza utilizar el subíndice i para los trabajos, es decir, éstos ya no están referidos a su posición en la secuencia, sino que se identifica directamente qué número de trabajo usa cada variable. En consecuencia, se deja usar Z_{ij}, que permitía pasar del índice j al i y en su lugar se utilizan pares de restricciones dicotómicas.

Modelo original

En el modelo original de SGST aparece la variable C_{Max} utilizada en el modelo adaptado TS2, que representa directamente el valor del makespan de todos los trabajos, que deberá ser

superior a cada una de las fechas de finalización de todos los trabajos en la última máquina, representadas por la variable C_{Mi}, tal y como se puede observar en el conjunto de restricciones (4.16). La función objetivo de este modelo original es minimizar el makespan de todos los trabajos, por lo que es trivial su modelaje con el uso de la variable C_{Max}.

Para completar el modelo hace falta añadir algunas restricciones más, como por ejemplo el conjunto de restricciones (4.12), que representa que ningún trabajo puede terminar en la primera máquina antes que su tiempo de procesado, es decir, una vez que se ha cumplido el tiempo necesario para su procesamiento, es posible que pueda terminar. Por su parte, el conjunto de restricciones (4.13) indica que ningún trabajo pueda terminar en una máquina si no ha sido totalmente procesado en la máquina anterior y haya cumplido su tiempo de procesado en la máquina actual.

Por último y como se indicaba anteriormente, este modelo añade un par de restricciones dicotómicas ((4.14) y (4.15)), en las que a partir de la variable D_{ik} se permite que cualquier trabajo i pueda proceder o seguir a un trabajo k, pero nunca ambos simultáneamente. Esto se permite ya que dicha variable indica 1 o 0 si el trabajo i es procesado en la secuencia antes que el k o viceversa, respectivamente.

Min C_{Max}

sa

$$C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \qquad (4.12)$$

$$C_{r+1,i} - C_{r,i} \geq T_{r+1,i} \quad (1 \leq r \leq M-1)\ (1 \leq i \leq N) \qquad (4.13)$$

$$C_{r,i} - C_{r,k} + P D_{ik} \geq T_{r,i} \quad (1 \leq r \leq M)\ (1 \leq i < k \leq N) \qquad (4.14)$$

$$C_{r,k} - C_{r,i} + P(1 - D_{ik}) \geq T_{r,k}$$

$$(1 \leq r \leq M)\ (1 \leq i < k \leq N) \qquad (4.15)$$

$$C_{Max} \geq C_{Mi} \quad (1 \leq i \leq N) \qquad (4.16)$$

$$C_{ri}, C_{Max} \geq 0 \quad (1 \leq i \leq N) \quad (1 \leq r \leq M)$$

$$D_{ik} \in \{0,1\}\ (1 \leq i \leq N)\ (1 \leq j \leq N)$$

Modelo adaptado

A diferencia del modelo TS2, el modelaje del problema multi agente es más sencillo y directo debido a esa utilización del índice i, que permite una fácil relación de las variables makespan y tardanza para los trabajos de cada agente.

En el caso del primer conjunto de trabajos, la restricción (4.9) es la misma que la (4.16) del modelo original, con el simple cambio de que solo afectará a los trabajos del primer conjunto,

los cuales se desea minimizar el makespan, tal y como aparece en la función objetivo del modelo, que queda sin alterar.

Para el segundo agente se añaden las restricciones (4.17) y (4.9). La primera de ellas es un conjunto de restricciones que permite calcular la tardanza de los trabajos pertenecientes al segundo conjunto como la diferencia entre sus fechas de finalización en la última máquina y sus fechas de entrega, siempre que dicha diferencia sea positiva. La restricción (4.9) representa el límite superior que no puede superar la función objetivo del segundo agente, siendo la suma de las tardanzas inferior a un valor ε^B.

Min C_{Max}

sa

$$\sum_{i=1}^{N} T_i \leq \varepsilon^B \ (i \in J^B) \quad (4.8)$$

$$C_{Max} \geq C_{Mi} \ (i \in J^A) \quad (4.9)$$

$$T_i \geq C_{Mi} - D_i \ (i \in J^B) \quad (4.17)$$

$$C_{1i} \geq T_{1i} \ (1 \leq i \leq N) \quad (4.12)$$

$$C_{r+1,i} - C_{r,i} \geq T_{r+1,i} \ (1 \leq r \leq M\text{-}1) \ (1 \leq i \leq N) \quad (4.13)$$

$$C_{r,i} - C_{r,k} + P D_{ik} \geq T_{r,i} \ (1 \leq r \leq M) \ (1 \leq i < k \leq N) \quad (4.14)$$

$$C_{r,k} - C_{r,i} + P(1 - D_{ik}) \geq T_{r,k}$$

$$(1 \leq r \leq M) \ (1 \leq i < k \leq N) \quad (4.15)$$

$$C_{ri} \geq 0 \ (1 \leq i \leq N) \quad (1 \leq r \leq M)$$

$$D_{ik} \in \{0,1\} \ (1 \leq i \leq N) \ (1 \leq j \leq N)$$

$$T_i \geq 0 \ (i \in J^B)$$

$$C_{Mi} \geq 0 \ (i \in J^A)$$

$$C_{Max} \geq 0$$

4.3.3. NUEVOS MODELOS. MODELO TS3.

Estos modelos son extraídos de (Tseng & Stafford, 2008), que surgieron para mejorar la eficiencia de los utilizados hasta ese momento, permitiendo que las instancias pudieran ser resueltas en tiempos computacionales menores para el objetivo tradicional del makespan. La técnica que se utilizó para la creación de dichos modelos fue la realización de una combinación de las ecuaciones usadas en (Stafford et al., 2005), reduciendo el número de variables presentes en ellos.

La formulación de las restricciones se realizó de una forma diferente a como se había hecho hasta entonces, ya que se utilizó el diagrama JAML, de esta manera, y apoyándose en él, se crearon las restricciones. En dicho diagrama y a diferencia de un diagrama de GANTT, además de mostrarse las actividades principales y sus tiempos de finalización y comienzo, se marcan los tiempos ociosos y los tiempos de espera de los trabajos en las máquinas.

En (Tseng & Stafford, 2008) se presentan dos modelos, denominados TBA y TS3. Como se demostró en el apartado 4.1, el modelo TS3 presenta mejores resultados ante instancias cada vez mayores, por lo que en este proyecto se adaptará dicho modelo original a los requerimientos de un entorno multi agente con dos conjuntos de trabajos.

Una de las principales particularidades de este modelo es que utiliza el subíndice j para referenciar los trabajos, como ocurre en los modelos de la familia Wagner. Además, debido a la combinación de ecuaciones de modelos anteriores, surgen restricciones muy largas y tediosas, por lo que, para simplificar la notación, se propone el uso de una nueva variable TT_{rj}=$\sum_{i=1}^{N} T_{ri}Z_{ij}$, que define el tiempo de procesado del trabajo en la posición j en la máquina r. Por último y como novedad, surge el concepto del Flowtime, representado por la variable F_j, definido como el tiempo total que un trabajo desarrolla en el sistema de producción, incluyendo los tiempos de espera. Como en este proyecto se ha supuesto que todos los trabajos se encuentran disponibles al inicio del horizonte temporal, se llega a la conclusión de que F_j=E_{Mj}, por lo que el flowtime en este caso será igual al tiempo de finalización de los trabajos en la última máquina.

Modelo original

Debido a la utilización del subíndice j, se vuelve a utilizar la variable Z_{ij} para identificar el trabajo que ocupa la posición j de la secuencia. Como ocurre en el modelo TS2, aparecen el conjunto de restricciones (4.1) y (4.2) relacionadas con la variable Z_{ij}, además de una nueva restricción (4.18) con la variable Y_{rj}, ya que el modelo TS3 basa sus restricciones en los tiempos de espera de los trabajos en cada máquina.

Por último, el conjunto de restricciones (4.19) es originado a partir de una compleja combinación de ecuaciones que define una inecuación relacionando las variables Z_{ij}, T_{ri} y Y_{qj}.

Min $\sum_{i=1}^{N} T_{1i}$+$\sum_{\mathrm{r}=2}^{\mathrm{M}} \sum_{i=1}^{N} (T_{ri}\, Z_{i,N})$+ $\sum_{r=1}^{M-1} Y_{rN}$

sa

$1=\sum_{j=1}^{N} Z_{ij}$ $(1\leq i\leq N)$ (4.1)

$1=\sum_{i=1}^{N} Z_{ij}$ $(1\leq j\leq N)$ (4.2)

Y_{r1}=0 $(1\leq r\leq M\text{-}1)$ (4.18)

$$\sum_{i=1}^{N} T_{1i}\, Z_{i,j-1} + \sum_{q=1}^{r-1} \sum_{i=1}^{N} (Z_{ij} - Z_{i,j-1})\, T_{qi} -$$
$$\sum_{i=1}^{N} T_{ri}\, Z_{i,j-1} + \sum_{q=1}^{r-1} (Y_{qj} - Y_{q,j-1}) \geq 0$$
$$(2 \leq r \leq M)\ (2 \leq j \leq N) \qquad (4.19)$$
$$Y_{r,j}\ \ (1 \leq r \leq M-1)\ (1 \leq j \leq N)$$
$$Z_{ij} \in \{0,1\}\ (1 \leq i \leq N)\ (1 \leq j \leq N)$$

Modelo adaptado

Para adaptar el modelo original a un problema con varios conjuntos de trabajos se hace uso de la variable TT_{rj} mencionada anteriormente, que aparece en el conjunto de restricciones (4.22). Además, se calculan los tiempos de finalización de los trabajos en (4.23), mediante el uso de sumatorios de los tiempos de procesado y los tiempos de espera de los diferentes trabajos.

Al igual que en el caso del modelo TS2 se añaden las restricciones (4.7), (4.8), (4.9), (4.10) y (4.20) y (4.21), cambiando la variable E_{Mj} por F_j.

Min C_{Max}

sa

$$\sum_{i=1}^{N} T_i \leq \mathcal{E}^B\ (i \in J^B) \qquad (4.8)$$
$$C_{Max} \geq C_{Mi}\ \ (i \in J^A) \qquad (4.9)$$
$$T_i + P(1 - Z_{ij}) \geq Tard_j\ (i \in J^B)\ (1 \leq j \leq N) \qquad (4.10)$$
$$C_{Mi} + P(1 - Z_{ij}) \geq F_j\ (i \in J^A)\ (1 \leq j \leq N) \qquad (4.20)$$
$$Tard_j \geq F_j - \sum_{i=1}^{N} D_i Z_{ij}\ (1 \leq j \leq N) \qquad (4.21)$$
$$Tard_j \geq 0\ (1 \leq j \leq N) \qquad (4.7)$$
$$TT_{rj} = \sum_{i=1}^{N} T_{ri} Z_{ij}\ (1 \leq j \leq N)\ (1 \leq r \leq M) \qquad (4.22)$$
$$F_j = \sum_{p=1}^{j} TT_{1p} + \sum_{q=2}^{M} TT_{qj} + \sum_{q=1}^{M-1} Y_{qj}\ (1 \leq j \leq N) \qquad (4.23)$$
$$1 = \sum_{j=1}^{N} Z_{ij}\ \ (1 \leq i \leq N) \qquad (4.1)$$
$$1 = \sum_{i=1}^{N} Z_{ij}\ \ (1 \leq j \leq N) \qquad (4.2)$$
$$Y_{r1} = 0\ (1 \leq r \leq M\text{-}1) \qquad (4.18)$$
$$\sum_{i=1}^{N} T_{1i}\, Z_{i,j-1} + \sum_{q=1}^{r-1} \sum_{i=1}^{N} (Z_{ij} - Z_{i,j-1})\, T_{qi} -$$
$$\sum_{i=1}^{N} T_{ri}\, Z_{i,j-1} + \sum_{q=1}^{r-1} (Y_{qj} - Y_{q,j-1}) \geq 0$$
$$(2 \leq r \leq M)\ (2 \leq j \leq N) \qquad (4.19)$$
$$F_j, TT_{rj}\ \ (1 \leq r \leq M)\ (1 \leq j \leq N)$$

$$Y_{r,j}\ (1 \leq r \leq M-1)$$

$$Z_{ij} \epsilon \{0,1\}\ (1 \leq i \leq N)\ (1 \leq j \leq N)$$

$$T_i \geq 0\ (i \epsilon J^B)$$

$$C_{Mi} \geq 0\ (i \epsilon J^A)$$

$$C_{Max} \geq 0$$

4.4. CÁLCULO DE ÉPSILON.

Como se ha comentado en los apartados anteriores, la manera utilizada en este proyecto para resolver los MASPs es el tratamiento de las funciones objetivos mediante el método de épsilon-constraint. La principal dificultad que tiene dicho método se basa en la búsqueda del valor de épsilon, un valor límite superior que la función objetivo del segundo agente no debe superar. Durante la historia del estudio de este tipo de problemas se han empleado algunos métodos para su cálculo y en este proyecto se propone el algoritmo NEH para su resolución.

El algoritmo NEH es una heurística creada por (Nawaz, Enscore & Ham, 1923) y es utilizada en multitud de algoritmos evolutivos, principalmente para encontrar una buena primera solución factible.

Esta heurística consigue crear una secuencia insertando trabajos consecutivamente (previamente ordenados de forma descendente según la suma de sus tiempos de proceso en todas las máquinas) y evaluando las diferentes secuencias parciales que se van originando. El algoritmo NEH se ha utilizado tradicionalmente para resolver un problema de flowshop de permutación donde se desea minimizar el makespan.

El algoritmo NEH en su forma tradicional se basa en los siguientes pasos:

1. Se calcula la suma de todos los tiempos de proceso en todas las máquinas para cada trabajo y se ordenan de forma descendente.
2. Se escogen los dos primeros trabajos de la lista y se forman las dos secuencias parciales posibles con dicho par de trabajos.
3. Se evalúa el makespan de ambas secuencias y se escoge la de menos makespan, siendo ésta la secuencia incumbente o provisional que se toma como solución inicial de partida para la siguiente iteración del algoritmo.
4. Se elige el siguiente trabajo de la lista del paso 1 y se coloca en todas las posiciones posibles de la secuencia del paso anterior, respetando la posición relativa de los trabajos ya colocados.
5. Se repite el paso 4 hasta secuenciar todos los trabajos.

Para el cálculo de épsilon se realiza un procedimiento que utiliza el algoritmo NEH como base, como se puede observar a continuación:

1. Se dividen los trabajos en dos según su pertenencia a un conjunto de trabajos u otro.
2. Se aplica el algoritmo NEH a los trabajos del primer agente.

3. Una vez secuenciado los trabajos del primer agente, se van insertando a continuación del último trabajo secuenciado, los trabajos del segundo agente aplicándose el algoritmo NEH.
4. Se evalúa la función objetivo del segundo agente para la secuencia obtenida del paso 3. Dicho valor obtenido es el valor de épsilon.

Para observar de forma ejemplificada el proceso descrito anteriormente se describe a continuación el cálculo de épsilon en un problema simple resuelto por este método:

Se desea calcular épsilon de un problema Épsilon-constraint en un entorno de flowshop de permutación donde se desea minimizar el makespan del primer conjunto de trabajos y que la tardanza total del segundo agente no supere un cierto valor épsilon.

Los tiempos de proceso de los trabajos pertenecientes a J^A y J^B y las fechas de entrega de los trabajos de J^B se muestran en la tabla 3.

		TRABAJOS			
		J^A		J^B	
		T0	T1	T2	T3
MÁQUINAS	M0	5	2	1	3
	M1	2	4	4	3
		Fechas de entrega		2	6

Tabla 3. Datos de partida de ejemplo de cáculo de Épsilon.

Para empezar con el algoritmo se aplica el algoritmo NEH a los dos trabajos del primer agente, ordenándose de forma descendente según sus tiempos totales de procesado:

-Tiempo total de procesado de T0= 7 u.t.

-Tiempo total de procesado de T1= 6 u.t.

Por tanto, la lista de trabajos ordenados de forma descendente es [T0, T1], y se prueban las dos posibles secuencias con dichos trabajos.

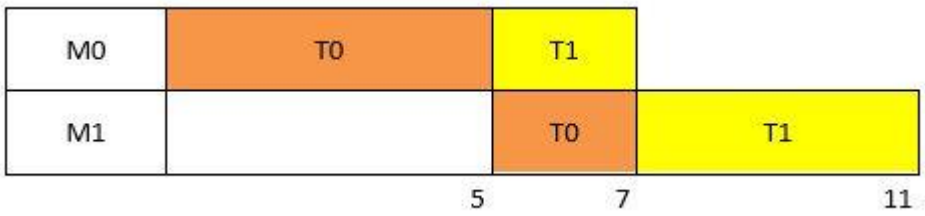

Ilustración 9. Secuencia [T0,T1], EJEMPLO CÁCULO DE ÉPSILON

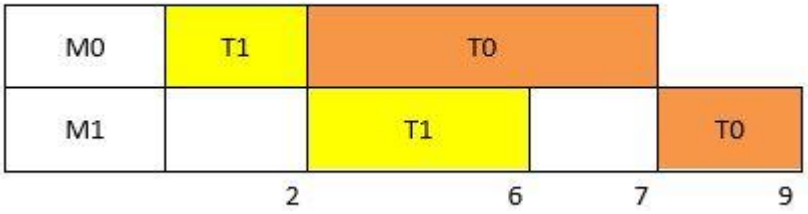

Ilustración 10.Secuencia [T1,T0], EJEMPLO CÁCULO DE ÉPSILON

En este caso, al haber solo dos trabajos no se produciría ninguna inserción más en ninguna iteración. Sin embargo, en el caso de que hubieran más de dos trabajos habría que realizar sucesivas iteraciones hasta terminar con todos los trabajos.

Como se puede observar en las ilustraciones 9 y 10, la secuencia parcial que produce un menor makespan es [T1,T0], por lo que será la secuencia base para la siguiente etapa del método.

A continuación, se aplica el algoritmo NEH para los trabajos del segundo agente.

-Tiempo total de procesado de T2= 5 u.t.

-Tiempo total de procesado de T3= 6 u.t.

Por tanto, la lista de trabajos ordenados de forma descendente es [T3, T2], y se prueban las dos posibles secuencias con dichos trabajos, insertados a continuación de la secuencia parcial formada por trabajos del primer agente obtenida del paso anterior.

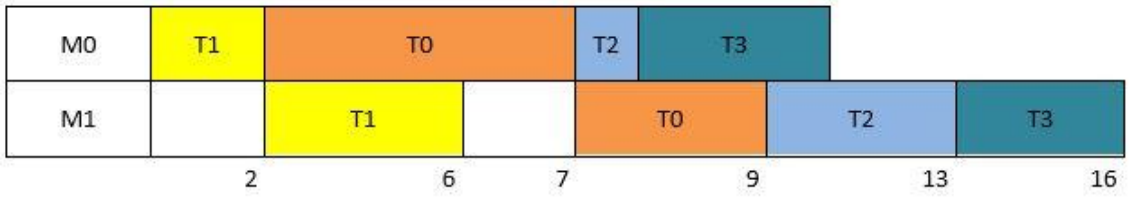

Ilustración 12. Secuencia [T1,T0, T2, T3], EJEMPLO CÁCULO DE ÉPSILON

Ilustración 11. Secuencia [T1,T0, T3, T2], EJEMPLO CÁCULO DE ÉPSILON

La secuencia que provoca un menor makespan es [T1, T0, T2, T3], por lo que es la escogida para el cálculo de épsilon. Por tanto, el último paso es evaluar la función objetivo del segundo conjunto de trabajos, que en el caso estudiado es la tardanza total de los trabajos T3 y T2.

Tardanza de T2= 13-2= 11 u.t.

Tardanza de T3=16-6=10 u.t.

Por tanto, la tardanza total del segundo agente es igual a 21 u.t, por lo que éste será el valor de épsilon escogido para este problema.

4.5. PROGRAMACIÓN EN LENGUAJE C.

La programación de los modelos para este proyecto se ha realizado con el programa informático CodeBlocks, un entorno de desarrollo integrado de código abierto que permite el desarrollo de programas en lenguaje C. Dicha programación está orientada a la realización de los modelos en formato LP para su resolución en Gurobi. Este tipo de formato es el más intuitivo debido a su facilidad de lectura y están estructurados en varias secciones según el tipo de modelo a realizar. Las secciones utilizadas en los modelos de este proyecto son las siguientes:

1. Sección objetivo: Esta sección empieza con la palabra clave “minimize” seguido de la expresión que se desee optimizar.
2. Sección de restricciones: Esta sección empieza con la expresión “subject to:” seguido del conjunto de restricciones del modelo. Para ello, se utilizan las diferentes variables separadas por espacios y con los signos =, <=, <,> = o >. Para la separación de las restricciones se usan los saltos de línea.
3. Sección de tipo de variables: Las variables pueden ser designadas como binarias (Bin), enteras (general) o continuas. Gurobi las asume como continuas. En este proyecto los modelos son lineales con todos los datos enteros, por lo que provoca una situación en la que las soluciones serán también enteras y debido a ello, se definirán todas las variables como enteras.
4. Sección final: En esta sección se indica el final de la sección anterior y está reservada para la palabra clave “end”.

Cabe destacar que existe otra sección para indicar los límites de las variables, como se puede observar en la ilustración 13, sin embargo, todas las variables por defecto son positivas, por lo que dicha sección no tiene cabida en este proyecto.

```
Maximize
  x + y + z
Subject To
  c0: x + y = 1
  c1: x + 5 y + 2 z <= 10
  qc0: x + y + [ x ^ 2 - 2 x * y + 3 y ^ 2 ] <= 5
Bounds
  0 <= x <= 5
  z >= 2
Generals
  x y z
End
```

Ilustración 13. Ejemplo de formato LP. Fuente: https://www.gurobi.com/documentation/9.0/refman/lp_format.html

El ejemplo del apartado (4.4) se puede observar en el anexo de este proyecto.

Para la implementación de los códigos se ha utilizado la librería Schedule, creada por Jose Manuel Framiñán (2018). En ella, se incluye una colección de clases y funciones escritas en el núcleo del lenguaje. Para su uso se incluye en la parte superior del código la instrucción #include < schedule_lib.h>. Algunas de las funciones que permite esta librería se recogen a continuación.

La principal ventaja que aporta la librería es la facilidad de uso y manipulación de vectores y matrices, tanto a la hora de definirlos como al utilizarlos. Además, permite la lectura sencilla de los datos de entrada de cada instancia mediante la función (MAT_INT) loadPTimes_nrows (char * nombre_fichero, &num_trabajos, &num_maq, mostrar), donde los datos recogidos en un documento de texto de entrada son volcados en una matriz, ya sea de tiempos de procesado o fechas de entrega. Para la recepción correcta de dichos datos, éstos tienen que estar en un formato determinado, como se puede observar en la ilustración 14, donde se representan los datos de entrada del ejemplo del apartado (4.4). En este archivo de entrada se deben definir en las dos primeras líneas, el número de máquinas y trabajos para, posteriormente expresar una matriz (en este caso con los tiempos de proceso de cada trabajo en todas las máquinas) con un número de filas igual al número de trabajos y un número de columnas igual al número de máquinas. Para las fechas de entrega se introduce un documento de texto con la estructura similar, suponiendo que el número de máquinas es igual a uno, como se observa en la ilustración 15. Es importante tener en cuenta que CodeBlocks transpone la matriz una vez la lee, como se puede comprobar en la ilustración 16.

Ilustración 14. Formato de datos de entrada. Tiempos de procesado. Ejemplo.

Ilustración 15. Formato de datos de entrada. Fechas de entrega. Ejemplo.

Algunas funciones de la librería Schedule utilizadas en este proyecto son las siguientes:

- (int) iMax(numero1, numero2): Devuelve el máximo entre dos números enteros.

-(void) insertIVector(vector, tamaño, valor, posicion): Inserta el valor "valor" en la posición "posición" del vector "vector", siendo "tamaño" el del nuevo vector que se genera.

- (int) extractIVector (vector, tamaño, posicion): Elimina el elemento en la posición "posicion" del vector "vector". Devuelve el nuevo tamaño del vector, siendo "tamaño" el de antes de extraer el elemento.
- (void) copyIVector(original, destino, tamaño): Se crea un vector y se copia otro en él.

- (void) sortLVector(vector, vector_orden, numero_elementos, orden): Ordena un vector según un índice ascendente o descedente, siendo "vector_orden" los índices de dicho vector ordenado.

Cabe destacar que esta librería permite el cálculo de la función objetivo de un problema de Flow shop de permutación mediante unas funciones predeterminadas, sin embargo, en este proyecto se han creado funciones propias para su cálculo. La programación del modelo SGST se puede consultar en el anexo de este proyecto, quedando el resto de modelo disponibles en la versión digital del trabajo.

En definitiva, el procedimiento realizado para la programación de los modelos es el siguiente:

1. Recogida de los datos de entrada (tiempos de procesado de todos los trabajos y fechas de entrega del segundo agente) mediante la función loadPTimes_nrows.
2. Cálculo de épsilon mediante el algoritmo descrito en el apartado (4.4). Para ello, se hace uso de algunas funciones creadas, como por ejemplo:

   - int FSmakespan (int n, int m, VECTOR_INT secuencia, MAT_INT T): Se calcula el makespan de una determinada secuencia.

   -int tardanzatotalsegundogrupo (int n, int m, int na, VECTOR_INT secuencia, MAT_INT T, MAT_INT D): Se calcula la tardanza total de los trabajos del segundo agente.

   int makespanprimergrupo (int n, int m, int na, VECTOR_INT secuencia, MAT_INT T): Se calcula el makespan de los trabajos del primer agente.
3. Escritura del modelo en un archivo de texto de salida en formato LP.

En la ilustración 16 se puede observar la salida que se recibe por pantalla al realizar el modelo del ejemplo del apartado (4.4).

```
Instance Datos2.txt - n: 4    m: 1
    6    3    2    6
Instance Datos.txt - n: 4    m: 2
    5    2    1    3
    2    4    4    3

El vector de suma de tiempos del primer grupo de trabajo es:    7    6
La secuencia ordenada ascendente del primer grupo de trabajo es:    0    1

El makespan de la secuencia de trabajos NEHAes 9
La secuencia solucion del algoritmo NEHA es:    1    0

El vector de suma de tiempos del segundo grupo de trabajo es:    5    6
La secuencia ordenada descendente del segundo grupo de trabajo es:    1    0
La secuencia ordenada descendente del segundo grupo de trabajo en variables glob
ales es:    3    2


El makespan de la secuencia de trabajos es 16
La secuencia solucion del algoritmo NEHB es: 2 3

La secuencia TOTAL obtenida de NEHA Y NEHB es:    1    0    2    3

El epsilon es 21
Process returned 4 (0x4)   execution time : 0.052 s
```

Ilustración 16. Salida por pantalla. Ejemplo.

4.6. ELABORACIÓN DE BATERÍAS Y RESOLUCIÓN DE MODELOS.

Una vez realizada la programación de los modelos, el siguiente paso es la escritura de cada uno de ellos para cada instancia generada y la resolución mediante un solver, en este caso Gurobi, para comparar cuál de los tres modelos desarrollados es más eficiente para el problema multi agente estudiado. Para ello, se han creado una serie de instancias con datos de partida diferentes para su posterior modelado y resolución. En total consta de 640 instancias que dan lugar a problemas con los siguientes datos de partida. Se han realizado 10 instancias por cruce:

-Números de trabajos: 5,10, 15, 20. En el caso de las instancias pares, la primera mitad de los trabajos forman parte del primer conjunto de trabajos y la segunda mitad está incluida en el segundo conjunto. Por otro lado, en las instancias impares, los primeros trabajos hasta alcanzar el trabajo cuyo número es la mitad del número de trabajos totales redondeado al primer entero inferior es el número de trabajos del primer conjunto, siendo el resto de trabajos los que forman el primer conjunto. Por ejemplo, en caso de una instancia con 5 trabajos, los primeros dos trabajos introducidos como dato de partida, forman parte del primer conjunto y los otros tres restantes se integran en el segundo conjunto.

-Número de máquinas: 2, 3, 4, 5.

-Tiempos de proceso con distribución uniforme. U∼ [1,99].

-Fechas de entrega para los trabajos del segundo agente creadas a partir de una distribución uniforme discreta entre dos valores U∼ [p(1-T-R/2), p(1-T+R/2)]. Dichas fechas de entrega son generadas en función de tres valores:

-P=Es un valor grande y se define como la suma de tiempos de proceso de todos los trabajos en todas las máquinas. P=$\sum_{r=1}^{M} \sum_{i=1}^{N} T_{ri.}$ iϵJ^{A}, iϵJ^{B}.

-R y T= Son dos parámetros que toman el valor de 0,2 o 0,6 y se generan para crear diferentes escenarios de holgura, lo que provocará que existan problemas con fechas de entregas más cercanas a las fechas de finalización de los trabajos y otras instancias con fechas más holgadas.

La escritura y resolución de los modelos podría ser manual, sin embargo, como el objetivo es resolver una gran cantidad de instancias, sería inviable, por lo que es necesario la utilización de las baterías, que son simples archivos de texto con la extensión de archivo .BAT, que al ejecutarlas, sus instrucciones se realizan consecutivamente en la consola de CMD, es decir, la aplicación usada por Windows para ejecutar los comandos. De este modo, basta con crear un archivo de texto con las instrucciones necesarias para ejecutar el programa para todas las instancias creadas. En la ilustración 17 se puede observar un fragmento de la batería utilizada para la creación de los modelos SGST.

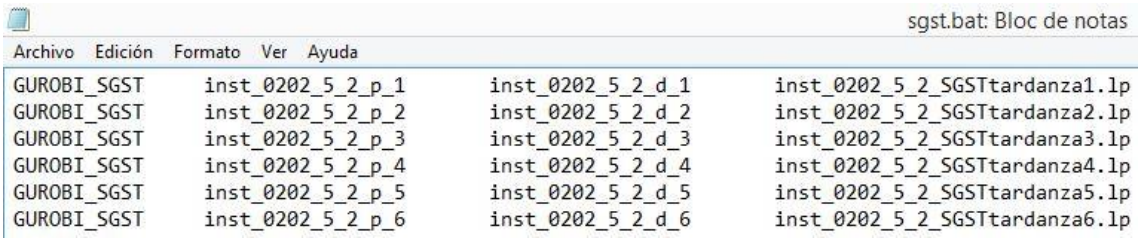

Ilustración 17. Batería para la generación de modelos.

La estructura utilizada para cada instrucción (separada de las demás por saltos de línea) consta de cuatro columnas en las cuales la primera de ellas se indica el nombre del ejecutable

en CodeBlocks que realiza la escritura del modelo. La segunda y tercera columna está reservada para las instancias, tanto los tiempos de procesado como las fechas de entrega, siendo la última columna el archivo.lp donde se genera el modelo. Es importante destacar que para que la batería sea ejecutada con éxito, tanto ésta como las instancias y el ejecutable de CodeBlocks deben estar en la misma carpeta.

Una vez realizada la creación de todos los modelos para SGST, TS2 y TS3, el siguiente paso es la resolución de ellos mediante Gurobi. Para ello se recurre de nuevo a la creación de baterías, que deben estar en la misma carpeta que el archivo .dll y el ejecutable de Gurobi, además de los modelos originados del paso anterior. El formato de estas baterías se puede observar en la ilustración 18, donde la primera columna está reservada para la expresión gurobi_cl para conectar con Gurobi. Además, se limita el tiempo de resolución a 900 segundos, por lo que, si en este tiempo, el modelo no ha sido capaz de resolver la instancia, se procede a resolver la siguiente. Los archivos de entrada son los modelos en formato.lp creados del paso anterior, siendo archivos en formato .SOL los documentos creados como salida, donde se recogen la resolución de dichos modelos. En la ilustración 19 se puede observar un ejemplo de uno de estos archivos para una instancia del modelo SGST.

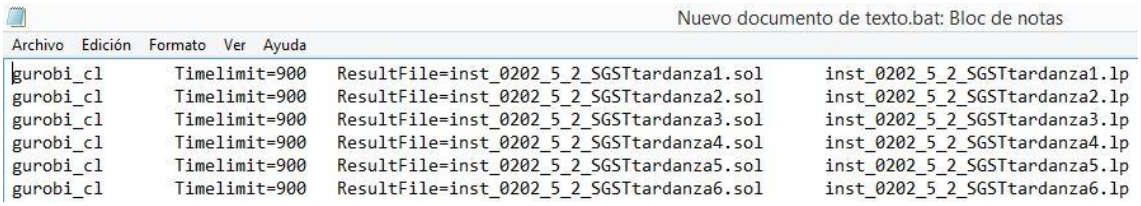

Ilustración 18. Batería para la resolución de modelos.

```
# Objective value = 116
CMAX 116
T2 0
T3 0
T4 0
C10 85
C11 116
C12 323
C13 286
C14 229
C00 3
C01 70
C02 275
C03 204
C04 136
D01 1
D02 1
D03 1
D04 1
D12 1
D13 1
D14 1
D23 0
D24 0
D34 0
```

Ilustración 19. Archivo .SOL de una instancia de un modelo SGST.

El último paso es la obtención de los tiempos de resolución de cada instancia. Para ello, se ha hecho uso de un ejecutable que toma como entrada el archivo .log creado por Gurobi, donde recoge todos los datos que han sido sacados por pantalla en la resolución de los modelos. Como salida se obtiene un archivo Excel en el que se recogen los tiempos de resolución de cada instancia para su posterior análisis estadístico.

En resumen, el procedimiento empleado en este apartado se resume en la ilustración 20.

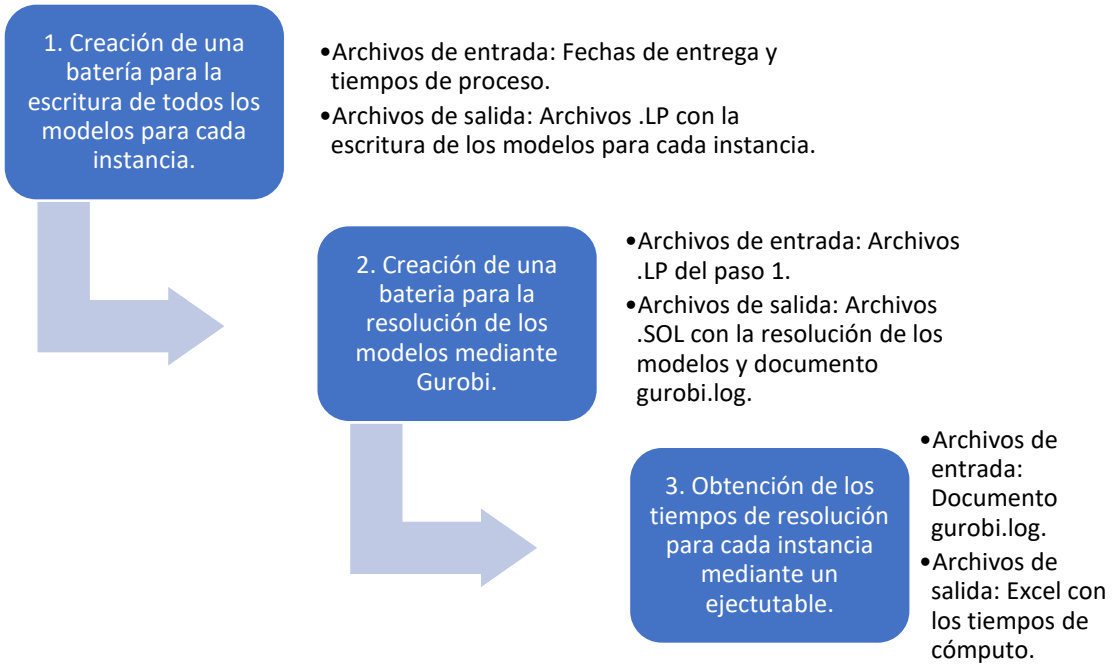

Ilustración 20. Diagrama de flujo de resolución de instancias.

4.7. RESULTADOS EXPERIMENTALES.

4.7.1. TIEMPOS COMPUTACIONALES TOTALES

En este apartado se realiza un análisis de los CPU time totales al resolver los tres modelos elegidos, teniendo en cuenta las 640 instancias aportadas. Para ello, se calculan la media de dichos tiempos, así como su desviación estándar y un intervalo de confianza al 95%. Todas estas medidas se han realizado con la herramienta Excel.

En rasgos generales se puede sentenciar que el modelo SGST es el mejor para este tipo de problemas, pues presenta una media y desviación inferiores a los modelos TS2 y TS3, como se puede observar en la tabla 4 y de forma gráfica en la ilustración 21. Además, en la tabla 5 se

indica que al 95%, Gurobi es capaz de resolver los modelos SGST en un intervalo entre 3 y casi 5 segundos, un tiempo bajo teniendo en cuenta los tiempos recogidos en (Ameneiros, 2018).

Esta mejora puede ser debido a que el modelo SGST utiliza el índice i para referirse a los trabajos y no el j como el resto de los modelos estudiados. Esto provoca que el número de variables usadas se reduzca, disminuyendo el tamaño de este, ya que las variables están referenciadas con el mismo índice de las que son aportadas como entrada, omitiéndose de esta forma variables como Z_{ij}.

Comparándose los dos modelos restantes TS2 y TS3, se puede observar una clara diferencia entre ambos, siendo el modelo TS3 el menos eficiente en términos generales, además de tener una dispersión muy alta. Esto es debido a que como se verá a continuación, todos los modelos resuelven las instancias pequeñas en tiempos muy bajos, sin embargo, una vez que dichas instancias incluyen mayor número de trabajos, el modelo TS3 se comporta peor en términos de eficiencia que SGST y TS2.

En definitiva, la diferencia de eficiencia entre los modelos está marcada por su comportamiento ante instancias grandes, ya que instancias con un número bajo de trabajos y máquinas (N=5, M=2) pueden ser resueltas en tiempo prácticamente nulos. Por lo tanto, el modelo SGST es el que mejor se comporta ante dicha situación.

Por último, cabe destacar que, aunque se aportó a Gurobi una instrucción para limitar el tiempo de resolución a 900 segundos, todas las instancias han sido resueltas en un tiempo menor a este, a diferencia de lo que sucedía en (Ameneiros, 2018), en el cual, un gran número de instancias de un problema tradicional con un solo agente no eran resueltas en dicho tiempo límite. Esta situación es originada probablemente por una gran holgura de las fechas de entrega.

MODELO	MEDIA	DESVIACIÓN ESTÁNDAR
SGST	3,95	10,52
TS2	9,25	24,64
TS3	16,08	53,28

Tabla 4. Media y desviación de los tiempos de cómputo totales.

MODELO	LÍMITE INFERIOR	LIMITE SUPERIOR
SGST	3,13	4,77
TS2	7,34	11,15
TS3	11,95	20,21

Tabla 5. Intervalos de confianza 95% de los tiempos de cómputo totales.

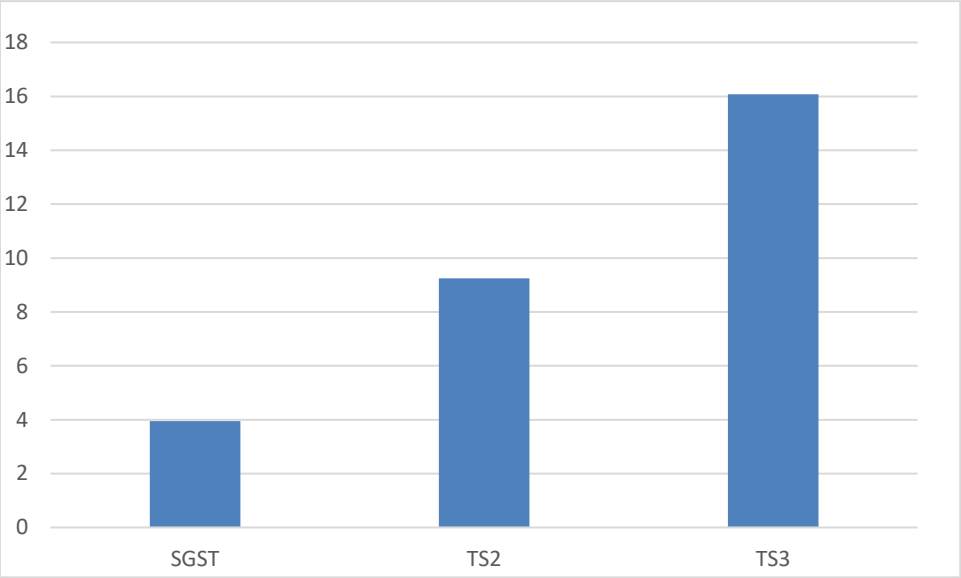

Ilustración 21. Tiempos de computación medios de los modelos.

4.7.2. TIEMPOS COMPUTACIONALES EN FUNCIÓN DEL NÚMERO DE MÁQUINAS

En el apartado 4.7.1 se ha realizado un análisis de la eficiencia de los modelos a rasgos generales, teniendo en cuenta todas las instancias, sin distinguir entre número de máquinas ni trabajos. En este apartado se realiza el estudio del comportamiento de los tres modelos según el número de máquinas que presente la instancia.

Se han resuelto 160 instancias para cada número de máquinas, es decir, 160 instancias con dos máquinas, y otras 160 para M=3, M=4 y M=5, por lo que el estudio se puede definir como representativo.

En la tabla 6 se puede observar un aumento de los tiempos de resolución cuando el número de máquinas aumenta, aumentándose también su dispersión. Esta tendencia no la sigue el modelo SGST, que presenta la mayor media de todas para instancias con M=2. Sin embargo, la situación se revierte una vez que el número de máquinas aumenta, siendo como se indicó en el apartado 4.7.1, el modelo más eficiente.

En la ilustración 22 se puede observar que la dispersión del modelo TS3 para M=5 es muy alta, y esto es debido a que como se verá en el apartado 4.7.3, dicho modelo resuelve de forma muy eficiente las instancias con un número de trabajos bajo (N=5), a pesar de que el número de máquinas sea más alto, pero sin embargo, no es eficiente en instancias con un gran número de trabajos (N=20, en este proyecto), por tanto, esto provoca que su desviación estándar aumente. Esta situación se ve reflejada también en los otros dos modelos, pero en menor medida.

Por tanto, se puede llegar a la conclusión de que el número de máquinas no es la variable más representativa a la hora de escoger qué modelo es más eficiente, puesto que parece claro que el número de trabajos influye mucho más en el tiempo de resolución por parte de Gurobi.

	MEDIA				DESVIACIÓN ESTANDAR			
	M=2	M=3	M=4	M=5	M=2	M=3	M=4	M=5
SGST	4,88	2,72	2,99	5,21	9,81	5,98	8,26	15,48
TS2	2,94	5,55	8,94	19,56	6,16	11,55	17,05	42,61
TS3	4,61	9,12	15,42	35,18	8,56	18,44	31,88	97,10

Tabla 6. Media y desviación de los tiempos de computación en función de M.

	M=2		M=3		M=4		M=5	
	INF	SUP	INF	SUP	INF	SUP	INF	SUP
SGST	3,36	6,40	1,79	3,65	1,71	4,27	2,81	7,61
TS2	1,99	3,90	3,76	7,34	6,29	11,58	12,95	26,16
TS3	3,28	5,93	6,26	11,98	10,48	20,36	20,14	50,23

Tabla 7. Intervalos de confianza 95% de los tiempos de computación en función de M.

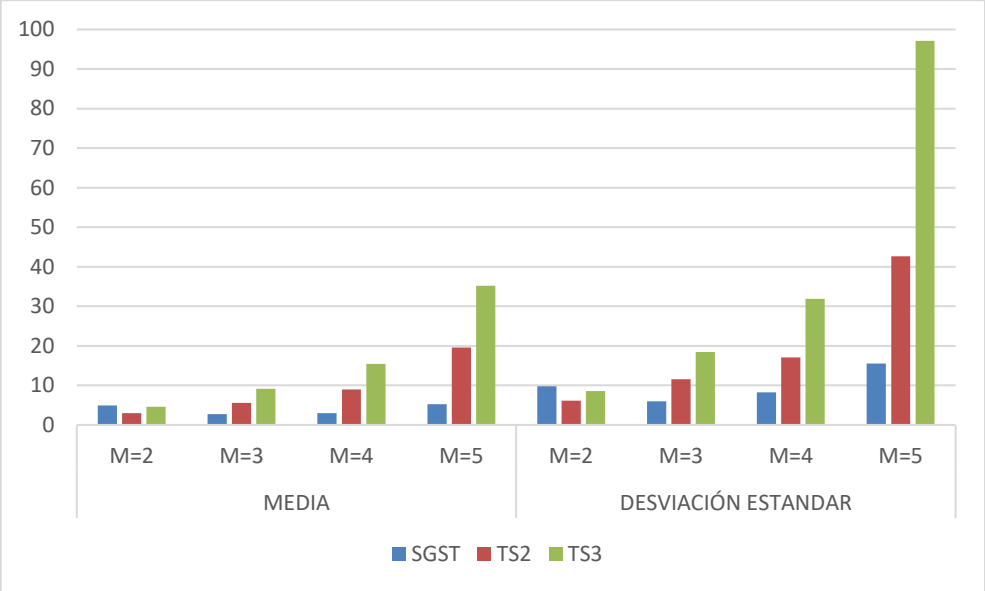

Ilustración 22. Media y desviación de los tiempos de computación en función de M.

4.7.3. TIEMPOS COMPUTACIONALES EN FUNCIÓN DEL NÚMERO DE TRABAJOS

Una vez realizado el análisis de los tiempos computacionales en función del número de máquinas, se procede a realizar el estudio en función de otro importante parámetro, el número de trabajos. Como se indicó en el apartado anterior, este parámetro influye más en la eficiencia del modelo, puesto que los tiempos de resolución aumentan de forma exponencial con un aumento del número de trabajos. Este aspecto se ve reflejado tanto en la media como en la dispersión de los datos, como se puede observar en la tabla 8.

Se puede confirmar que el modelo SGST es el más eficiente para cualquier número de trabajos, apareciendo una gran diferencia cuando N=15, ya que, en instancias pequeñas, todos los modelos se comportan de forma muy eficiente con tiempos de resolución prácticamente nulos para todas ellas, como se puede observar en las desviaciones de dichas instancias en la tabla 8. Por tanto, en instancias con un número bajo de trabajos sería indiferente la elección de un modelo u otro. Sin embargo, para modelos con N>10, el modelo SGST debe ser escogido para su resolución, ya que es capaz de resolver instancias grandes en tiempos relativamente bajos en comparación con el tamaño de los modelos.

Es importante destacar que la dispersión del modelo TS3 para N=20 es muy grande, y esto es debido a que es capaz de resolver algunas instancias grandes en poco tiempo computacional, pero, sin embargo, la mayoría de ellas necesita de un tiempo superior a los modelos TS2 y SGST. Para observar de forma gráfica los resultados, se han realizado las gráficas 23 y 24. Se ha dividido en dos para que se puedan observar en la escala correcta debido a la diferencia de tiempos entre instancias grandes y pequeñas.

En definitiva, el parámetro N es más influyente que el número de máquinas y extrapolándose los datos, los modelos difícilmente podrían encontrar una solución en tiempos viables para instancias con más de 20 trabajos. Se puede sentenciar que, a igualdad del resto de parámetros, el modelo SGST es el más eficiente con la variación del número de trabajos.

	MEDIA				DESVIACIÓN ESTANDAR			
	N=5	N=10	N=15	N=20	N=5	N=10	N=15	N=20
SGST	0,02	0,14	0,72	14,93	0,01	0,71	0,62	16,79
TS2	0,03	0,46	4,14	32,36	0,02	0,23	4,70	41,11
TS3	0,03	0,60	5,53	58,17	0,01	0,34	3,64	94,86

Tabla 8. Media y desviación de los tiempos de computación en función de N.

	N=5		N=10		N=15		N=20	
	INF	SUP	INF	SUP	INF	SUP	INF	SUP
SGST	0,01	0,02	0,03	0,25	0,62	0,81	12,33	17,53
TS2	0,03	0,03	0,42	0,49	3,41	4,86	25,99	38,73
TS3	0,03	0,04	0,54	0,65	4,97	6,09	43,47	72,87

Tabla 9. Intervalos de confianza 95% de los tiempos de computación en función de N.

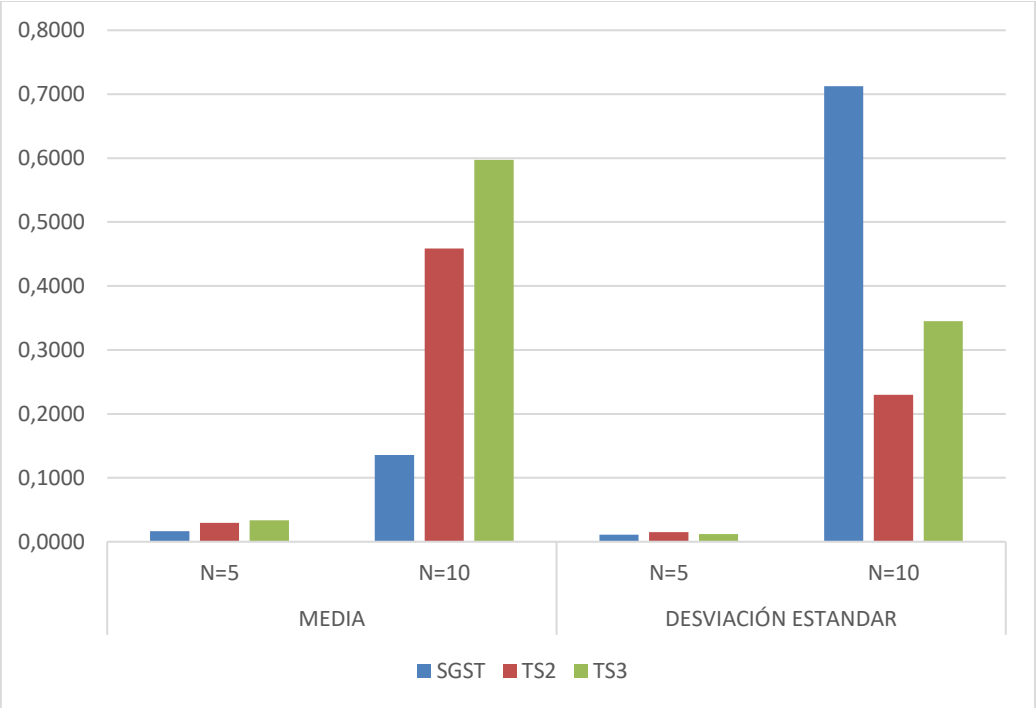

Ilustración 23. Media y desviación de los tiempos de computación en función de N (N=5, N=10).

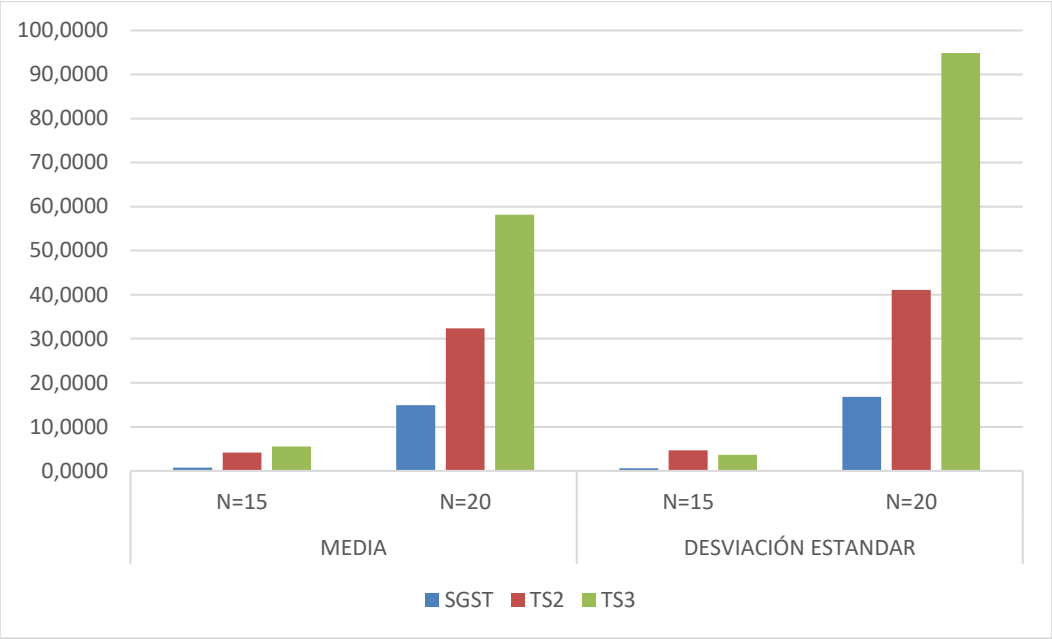

Ilustración 24. Media y desviación de los tiempos de computación en función de N (N=15, N=20).

4.7.4. TIEMPOS COMPUTACIONALES EN FUNCIÓN DE R Y T

Los dos últimos parámetros para tener en cuenta a la hora de analizar las instancias aportadas son R y T. Estos parámetros se generan para crear diferentes escenarios de holgura, lo que provoca que existan problemas con fechas de entregas ajustadas y otras instancias con fechas más holgadas, lo que influye en los tiempos computacionales de resolución, ya que es obvio que, al tener fechas de entrega ajustadas a los tiempos de finalización de los trabajos, se requerirá de un mayor tiempo computacional para obtener la solución óptima.

La situación mencionada anteriormente ocurre cuando R y T aumentan, es decir, un aumento de los parámetros provoca que las instancias no sean tan holgadas, dando lugar a un aumento de los tiempos de resolución en los modelos SGST, TS2 y TS3. Sin embargo, se puede observar que el aumento provocado no es muy alto a excepción del modelo TS3, ya que recibe un aumento de 7 segundos de media al cambiar el valor de ambos parámetros. Sin embargo, tal y como se puede observar en la tabla 10, los modelos SGST y TS2 reciben un aumento inferior al segundo, lo que resulta un incremento en el tiempo casi insignificante.

Además, se puede observar en la ilustración 25 que el aumento es similar tanto para R como para el parámetro T, es decir, al pasar de 0,2 a 0,6, el CPU time recibe un aumento en las mismas proporciones para ambos parámetros.

Esta tendencia se puede observar también en la dispersión de los datos, que se mantiene prácticamente constante en todas las casuísticas de SGST y TS2. Además, se produce un

aumento considerable en el caso del modelo TS3, tal y como se puede observar en la tabla 10.

Es importante destacar que los resultados vuelven a considerar el modelo SGST como el más eficiente de ellos, tal y como se pudo concluir con el estudio del resto de parámetros analizados, siendo el modelo TS3 el menos eficiente.

En definitiva, se puede confirmar que los parámetros R y T influyen de forma casi insignificante en el caso de los modelos SGST y TS2, aumentando ligeramente los tiempos de resolución, no sucediendo lo mismo para el caso del tercer modelo, TS3, que recibe un aumento tanto en la media como en la desviación estándar de los resultados obtenidos.

	MEDIA				DESVIACIÓN ESTANDAR			
	R=0,2	R=0,6	T=0,2	T=0,6	R=0,2	R=0,6	T=0,2	T=0,6
SGST	3,80	4,10	3,89	4,01	10,13	10,92	11,00	10,04
TS2	9,11	9,38	8,50	9,99	24,96	24,35	23,29	25,93
TS3	12,39	19,77	11,82	20,35	31,80	68,17	34,01	67,03

Tabla 10. Media y desviación de los tiempos de computación en función de R y T.

	R=0,2		R=0,6		T=0,2		T=0,6	
	I NF	SUP	I NF	SUP	I NF	SUP	I NF	SUP
SGST	2,69	4,91	2,91	5,30	2,68	5,09	2,91	5,11
TS2	6,38	11,85	6,71	12,05	5,95	11,05	7,15	12,83
TS3	8,91	15,87	12,31	27,24	8,09	15,54	13,00	27,69

Tabla 11. Intervalos de confianza 95% de los tiempos de computación en función de R y T.

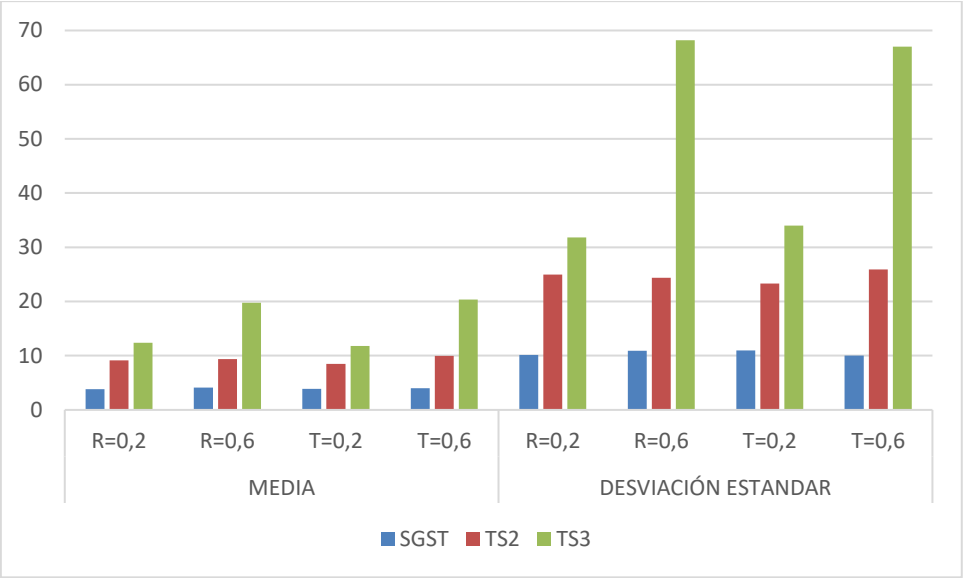

Ilustración 25. Media y desviación de los tiempos de computación en función de R y T.

4.7.5. CONCLUSIONES

Una vez realizado el análisis de los tiempos computacionales de resolución de los modelos en función de los distintos parámetros presentes en las instancias aportadas, se pueden obtener una serie de conclusiones en torno a este estudio.

En primer lugar, el modelo SGST ha obtenido los mejores resultados en términos generales, presentando además una desviación estándar baja en casi todos los estudios realizados, permitiendo una mayor fiabilidad a la hora de valorar los resultados. Esto puede deberse a la forma en la cual están referenciados los trabajos en este modelo, que como se indicó anteriormente, utiliza el subíndice “i” y no “j” como TS2 y TS3, permitiendo una concordancia entre las variables de salida y las de entrada, reduciéndose así el número de variables empleadas, lo que puede dar lugar a una reducción en los tiempos de resolución.

Por otra parte, se ha podido constatar que el número de trabajos es el parámetro más influyente en los tiempos de resolución, aumentando de forma exponencial con un incremento de ellos, tanto en media como en dispersión de los datos de salida. Cuando la instancia presenta una cantidad baja de trabajos, todos los modelos pueden resolverlas de forma muy eficiente, presentando tiempos muy bajos, por lo que se puede concluir que, en estos casos, se pueden resolver con cualquier modelo de forma prácticamente similar. No ocurre esto en el caso de N>10, a partir del cual el modelo SGST se destaca como el más eficiente de ellos.

Con respecto al número de máquinas, se observa que los modelos TS2 y TS3 presentan un aumento en los tiempos de resolución si el número de máquinas asciende. No obstante, en el caso de SGST se observa una tendencia irregular de subida, por lo que se concluye que no es un parámetro importante en el comportamiento de los modelos en su resolución.

Por último, los parámetros R y T tienen influencias diferentes en los tres modelos. Las instancias de TS3 presentan subidas superiores que TS2 y SGST, que apenas se ven influidas por un cambio de estos dos parámetros, aumentando ligeramente los tiempos de resolución con un aumento de ellos. Por lo tanto, se puede concluir, que al igual que M, no influyen de manera significativa en los modelos.

En definitiva, SGST es el modelo estudiado más eficiente para el caso de la resolución de forma exacta en Gurobi de un MASP con makespan y tardanza total, siendo N el parámetro más influyente en los tiempos de resolución de cada instancia.

CAPÍTULO 5.

INFLUENCIA DEL PARÁMETRO ÉPSILON EN LA RESOLUCIÓN DEL MODELO SGST.

5.1. INTRODUCCIÓN

En este capítulo se ha elaborado un análisis de la influencia del parámetro épsilon en el modelo más eficiente estudiado en este proyecto, que como se puede comprobar en el capítulo 4, es el modelo SGST. En dicho capítulo se resuelven los tres modelos analizados fijando épsilon en el valor obtenido al hacer el algoritmo propuesto en el apartado 4.4. Sin embargo, el siguiente paso a analizar es, qué ocurre al disminuir dicho valor, obligando a que la función objetivo del segundo agente (Tardanza total) esté restringida superiormente en un límite más bajo que los casos estudiados en el capítulo anterior. Esta situación provocará que los tiempos computacionales aumenten y se produzca una disminución en el valor de la función objetivo del primer agente, debido a que para producir una mejora en el criterio del agente B, se debe realizar generalmente una nueva secuencia que provoque un aumento en el makespan del primer conjunto de trabajos.

Para comprobar de forma más exacta el comportamiento de los modelos al disminuir el valor de épsilon, se han elaborado nuevas instancias con fechas de entrega más ajustadas. Dichas fechas de entrega, al igual que ocurre en el capítulo 4, se han elaborado siguiendo la distribución U~ [p(1-T-R/2), p(1-T+R/2)]. Sin embargo, el valor de p se define ahora como la suma de tiempos de proceso de todos los trabajos del segundo agente en todas las máquinas. P=$\sum_{r=1}^{M} \sum_{i=1}^{N} T_{ri.}$, i$\epsilon J^{B}$.

Para realizar este estudio, se han resuelto las instancias con R=0,2 y T=0,6, es decir, un total de 160 instancias para cada escenario de épsilon analizado. En total se han generado 6 situaciones distintas con diferentes valores de épsilon, generados a partir de la siguiente formula:

$$\text{épsilon}^{'} = épsilon\ (1 - \delta)$$

Los valores de delta serán 0, 0.1, 0.2, 0.3, 0.4 y 0.5, es decir, se disminuirá épsilon un 10, 20, 30, 40 y 50%.

5.2. ESCENARIO 1. DELTA=0.

En este caso, la situación es similar al capítulo 4, pero con la diferencia de que las instancias tienen fechas de entrega más ajustadas. Los resultados obtenidos para las 160 instancias con R=0.2 y T=0.6 se reflejan en la tabla 12 y de forma gráfica en la ilustración 26.

	Makespan		Épsilon		CPU time	
	Media	Desviación	Media	Desviación	Media	Desviación
TOTAL	438,01	174,50	1703,67	1365,53	8,74	32,10
N=5	230,58	77,37	464,43	184,81	0,04	0,02
N=10	393,60	91,09	1184,38	397,09	0,20	0,09
N=15	492,20	101,97	1906,00	806,39	1,88	1,10
N=20	635,68	101,13	3259,88	1543,54	32,83	58,35
M=2	346,48	163,32	2218,70	1696,38	4,48	8,57
M=3	398,33	161,04	1841,43	1306,08	6,57	17,42
M=4	465,70	148,06	1476,13	1172,97	7,79	19,83
M=5	541,55	166,21	1278,43	1054,83	16,12	57,89

Tabla 12. Valores makespan, épsilon y CPU time para delta=0

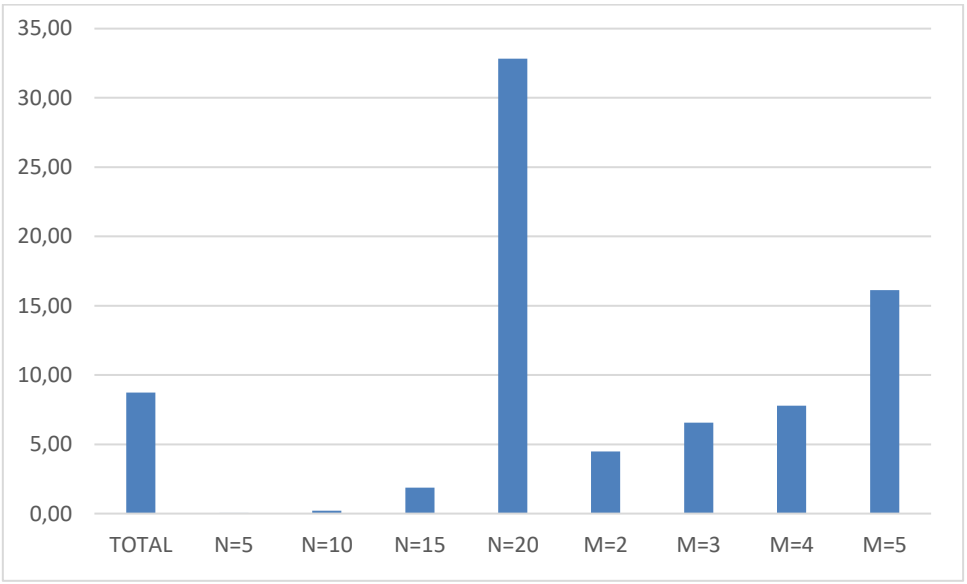

Ilustración 26. CPU time para delta=0

Como se puede observar en la ilustración 26, Gurobi es capaz de resolver todas las instancias en tiempos viables, siendo los casos de N=20 aquellas en las que más tiempo computacional se ha consumido. Para los casos de N=5 y N=10, las instancias son resueltas de forma casi

inmediata. Además, se observa una tendencia al alza del tiempo de resolución con un aumento del número de máquinas. Sin embargo, se confirman las conclusiones obtenidas en el capítulo 4, ya que el parámetro más importante vuelve a ser N.

En la tabla 12 se refleja que la función objetivo del agente B es mayor que el makespan del primer agente, debido a que es esta última la función objetivo a minimizar en el modelo. La tardanza total del segundo agente es de media aproximadamente 4 veces el valor del makespan del primer conjunto de trabajos, aumentando la diferencia con un aumento del número de trabajos.

5.3. ESCENARIO 2. DELTA=0.1

En este escenario los valores de épsilon se reducen un 10% en todas las instancias. Al reducir el límite superior, el modelo se convierte en más restrictivo, lo que provocará un aumento en los tiempos de resolución de las distintas instancias. Esta situación se ve reflejada en la tabla 13, donde se muestran el número de instancias que Gurobi no es capaz de resolver en un tiempo menor a 900 segundos, que es considerado en este proyecto como el máximo para que sea viable. En dicha tabla se presentan las instancias mencionadas según el número de máquinas y trabajos, además del total de todas ellas.

Gurobi es capaz de resolver todas las instancias de N=5 y N=10 en tiempos similares a las instancias con δ =0. Estos tiempos son casi nulos, ya que las instancias con un número bajo de trabajos son resueltas de forma prácticamente inmediata, por lo que una disminución de épsilon no influye de manera general en los tiempos de resolución de Gurobi en este tipo de instancias. Sin embargo, la situación es diferente cuando los modelos incluyen un mayor número de trabajos, ya que los tiempos de resolución aumentan, con una diferencia de casi 50 segundos de media en los casos de N=15 y N=20. Sin embargo, la situación es al contrario si el estudio se centra en el número de máquinas, disminuyendo los tiempos de resolución con respecto aumenta el número de ellas. Estos datos se pueden observar en la tabla 14.

El aumento del valor del makespan de los trabajos del agente A debido a una disminución de la función objetivo del segundo agente se ve reflejado en la gráfica 27. Como se puede observar, el porcentaje de aumento del makespan disminuye con un aumento del número de trabajos, alcanzando casi un 18% en el caso de N=5, mientras que la tendencia es prácticamente estable en el caso del estudio del número de máquinas.

Es importante destacar que tanto la tabla 14 como la ilustración 27 reflejan los datos sin tener en cuenta aquellas instancias que Gurobi no ha sido capaz de resolver en los 900 segundos fijados con anterioridad, puesto que dichas instancias no entran en este estudio.

	delta=0.1	
	Suma	Porcentaje
TOTAL	21/160	13,13%
N=5	0/40	0,00%
N=10	0/40	0,00%
N=15	3/40	7,50%
N=20	18/40	45,00%
M=2	10/40	25,00%
M=3	5/40	12,50%
M=4	3/40	7,50%
M=5	3/40	7,50%

Tabla 13. Número de instancias sin resolver para delta=0.1

	delta=0		delta=0.1	
	F.O	t(s)	F.O	t(s)
TOTAL	415,87	3,71	435,24	27,12
N=5	230,58	0,04	271,38	0,03
N=10	393,60	0,20	413,40	0,24
N=15	503,76	1,95	510,47	46,35
N=20	654,43	20,32	655,10	70,26
M=2	295,52	0,70	319,10	56,05
M=3	375,84	2,37	390,27	23,25
M=4	456,14	5,36	475,78	23,26
M=5	520,30	6,07	540,32	17,11

Tabla 14. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0 y 0.1

	DESVIACIÓN ESTÁNDAR CPU TIME	
	delta=0	delta=0.1
TOTAL	12,12	95,94
N=5	0,02	0,01
N=10	0,09	0,19
N=15	1,22	134,94
N=20	25,72	100,14
M=2	1,38	151,90
M=3	3,93	92,27
M=4	15,78	74,36
M=5	16,74	46,06

Tabla 15. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.1 con respecto a situación inicial.

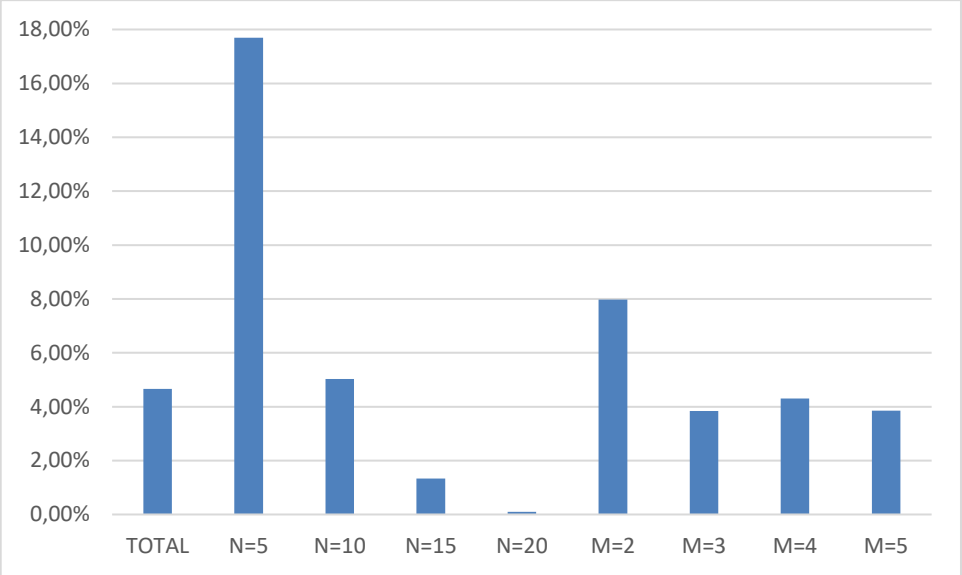

Ilustración 27. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,1 con respecto a delta=0.

5.4. ESCENARIO 3. DELTA=0.2

La situación en este apartado se origina al reducir el valor de épsilon un 20%. Como se puede observar en la tabla 16, Gurobi no resuelve ninguna instancia de N=20 trabajos en el tiempo límite fijado, por tanto, de aquí en adelante todas ellas no estarán tenidas en cuenta en ninguna tabla ni gráfica, ya que, al reducir aún más el valor de épsilon con respecto a los apartados posteriores, el solver de Gurobi no será capaz de resolverlas tampoco, al ser aún más restrictivas. Además, se observa que el 22,50% de las instancias con N=15 tampoco pueden ser resueltas en 900 segundos, pero, sin embargo, en aquellas instancias con N=5 y N=10, el óptimo es encontrado en tiempos muy bajos, tal y como se puede observar en la tabla 17, donde queda reflejado la media de los valores del makespan del agente A y el CPU time necesario para obtenerlos, para todas las instancias resueltas con delta=0.2, es decir, quedando fuera de estudio todas ellas que Gurobi no es capaz de encontrar el óptimo en los 900 segundos fijados. En dicha tabla también queda reflejada la media del makespan de dichas instancias mencionadas, en los casos anteriores de delta=0.1 y delta=0.

En el caso de N=15, el aumento del CPU time necesario para resolver los modelos es significativo con respecto a los casos anteriores, existiendo una diferencia de 90 segundos aproximadamente con respecto a delta=0.

	delta=0.2	
	Suma	Porcentaje
TOTAL	49/160	30,63%
N=5	0/40	0,00%
N=10	0/40	0,00%
N=15	9/40	22,50%
N=20	40/40	100,00%
M=2	15/40	37,50%
M=3	12/40	30,00%
M=4	12/40	30,00%
M=5	10/40	25,00%

Tabla 16. Número de instancias sin resolver para delta=0.2

	delta=0		delta=0.1		delta=0.2	
	F.0	t(s)	F.O	t(s)	F.O	t(s)
TOTAL	368,33	0,61	391,56	4,18	412,31	27,65
N=5	230,58	0,04	271,38	0,03	290,18	0,06
N=10	393,60	0,20	413,40	0,24	437,23	0,50
N=15	513,48	1,88	518,45	14,60	537,74	98,28
N=20	-	-	-	-	-	-
M=2	258,96	0,29	284,00	10,28	321,32	20,19
M=3	317,75	0,54	336,82	2,34	351,57	46,12
M=4	401,93	0,65	426,11	3,60	445,14	35,93
M=5	475,33	0,92	500,03	1,34	514,17	8,90

Tabla 17. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1 y 0.2

	DESVIACIÓN ESTÁNDAR CPU TIME	
	delta=0	delta=0.2
TOTAL	0,96	113,48
N=5	0,02	0,03
N=10	0,09	0,33
N=15	1,02	200,16
N=20	-	-
M=2	0,45	75,66
M=3	0,80	143,42
M=4	0,93	159,10
M=5	1,31	24,16

Tabla 18. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.2 con respecto a situación inicial.

En la ilustración 28 se puede analizar de forma gráfica el aumento de media de los valores óptimos del makespan del agente A que se obtiene al disminuir un 20% el valor de épsilon. La diferencia más amplia se obtiene en aquellas instancias con bajo número de trabajos y máquinas, es decir, las instancias más reducidas. Dicho aumento va disminuyendo cuando el número de trabajos es más grande y queda estabilizado con un mayor número de máquinas, tal y como se puede observar en el apartado 5.3.

En definitiva, de forma práctica se puede concluir que mejorar un 20% el valor del total tardiness del agente B supone un empeoramiento de menos del 5% en instancias con N=15, sin embargo, no se puede garantizar que todas ellas puedan ser resueltas en tiempos viables. Por lo tanto, el caso más favorable se puede concluir que se produce al resolver instancias con 10 trabajos, en la cual se produce un aumento del 10% del makespan, la mitad que lo que se mejora la función objetivo del segundo conjunto de trabajos, y, además, garantizándose un tiempo de resolución muy bajo. Para instancias más pequeñas, con N y M bajos, la mejora de un agente se traduce en un empeoramiento en mayor porcentaje del otro, por lo que no sería eficiente.

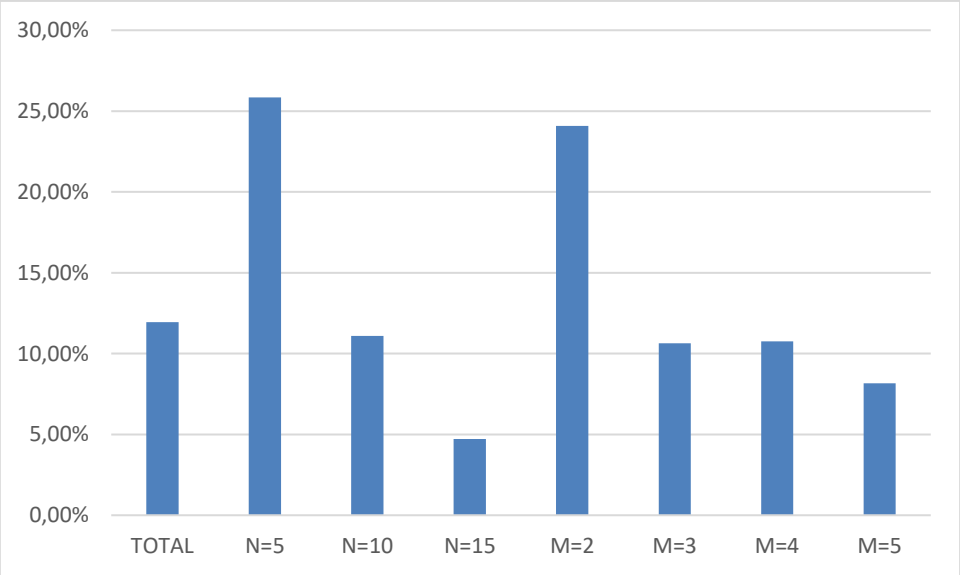

Ilustración 28. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,2 con respecto a delta=0.

5.5. ESCENARIO 4. DELTA=0.3

En este escenario se reduce el valor de épsilon un 30%. Como se comentó en el apartado anterior, Gurobi no es capaz de resolver ninguna instancia con 20 trabajos con los valores de épsilon originados al disminuirlos en tal porcentaje. Además, en esta situación, casi la mitad de las instancias con N=15 tampoco pueden ser resueltas en el tiempo límite fijado

de 900 segundos. Por otro lado, aquellas con tal número de trabajos que sí pueden ser resueltas, sufren un incremento significativo en los tiempos de resolución, pasando de los 22 segundos de media del escenario anterior, a los 220 segundos en el caso actual, tal y como se puede apreciar en la tabla 20. A pesar de ello, son las instancias que sufren un menor empeoramiento de la función objetivo del agente A al mejorar la tardanza total segundo conjunto de trabajos, tal y como refleja la ilustración 29. Dichos porcentajes de empeoramiento son relativos al caso con delta=0.

Asimismo, aquellas instancias con un número bajo de trabajos son resueltas en tiempos muy pequeños, sin embargo, al igual que ocurría en los casos anteriores, sufren un incremento significativo en el makespan de los trabajos del agente A. Dichas informaciones se pueden observar en la tabla 20 y en la ilustración 29.

De nuevo y tal como ocurre en los diferentes casos estudiados en este proyecto, el número de máquinas no tiene una influencia clara en los tiempos de resolución, al no seguir ningún patrón en concreto. Simplemente cabe destacar que solamente el caso de instancias pequeñas, con N=5 y M=2 tienen un empeoramiento del makespan superior en porcentaje a lo que se mejora el criterio del agente B, tal y como se observa en la ilustración 29.

	delta=0.3	
	Suma	Porcentaje
TOTAL	59/160	36,88%
N=5	0/40	0,00%
N=10	0/40	0,00%
N=15	19/40	47,50%
N=20	40/40	100,00%
M=2	16/40	40,00%
M=3	17/40	42,50%
M=4	13/40	32,50%
M=5	13/40	32,50%

Tabla 19. Número de instancias sin resolver para delta=0.3

	delta=0		delta=0.1		delta=0.2		delta=0.3	
	F.O	t(s)	F.O	t(s)	F.O	t(s)	F.O	t(s)
TOTAL	355,27	0,46	379,63	1,79	399,00	4,59	427,52	23,62
N=5	230,58	0,04	271,38	0,03	290,18	0,06	315,03	0,05
N=10	393,60	0,20	413,40	0,24	437,23	0,50	470,30	0,98
N=15	528,00	1,84	528,60	8,39	540,20	21,84	566,95	116,06
N=20	-	-	-	-	-	-	-	-
M=2	248,70	0,23	274,22	5,25	312,57	2,51	338,57	19,15
M=3	301,04	0,31	320,83	0,32	328,29	0,79	363,33	29,82
M=4	391,19	0,54	415,58	0,86	432,96	6,18	456,73	33,54
M=5	459,67	0,73	487,07	1,04	502,78	8,21	532,22	12,37

Tabla 20. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1, 0.2 y 0.3

	DESVIACIÓN ESTÁNDAR CPU TIME	
	delta=0	delta=0.3
TOTAL	0,81	86,13
N=5	0,02	0,02
N=10	0,09	0,66
N=15	0,94	165,55
N=20	-	-
M=2	0,43	76,17
M=3	0,55	93,30
M=4	0,84	121,06
M=5	1,11	35,48

Tabla 21. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.3 con respecto a situación inicial.

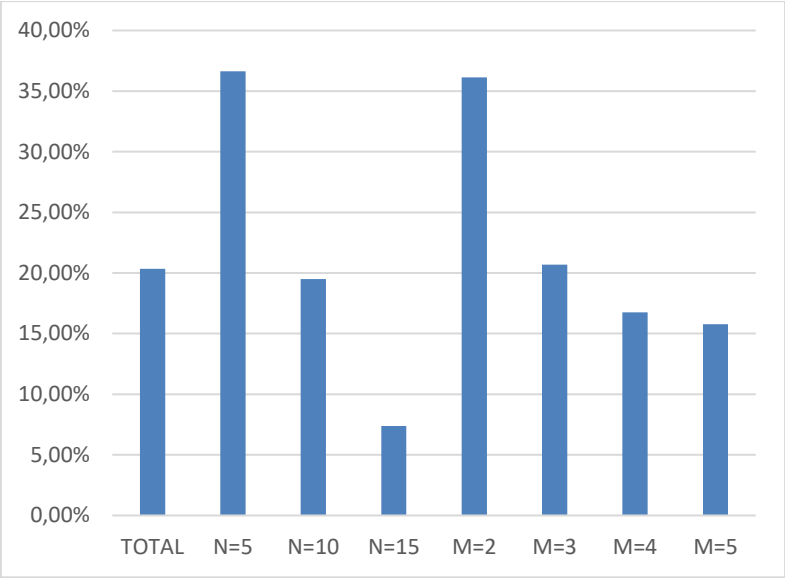

Ilustración 29. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,3 con respecto a delta=0.

5.6. ESCENARIO 5. DELTA=0.4

Una vez realizado el análisis estadístico del escenario en el cual los valores de épsilon se reducían un 30% con respecto a la situación original, en este apartado se procede a incrementar dicho valor hasta un 40%. En dicha situación, tal y como y se puede observar en la tabla 22, un 70% de las instancias con 15 trabajos no pueden ser resueltas en menos de 900 segundos. Además, aquellas que son resueltas tienen un CPU time de resolución de más de 80 segundos de media con respecto al escenario anterior, tal y como indica la tabla 23, a pesar de que tienen un incremento muy bajo de la función objetivo del agente A, en torno al 7%. Este último dato es interesante, puesto que es posible mejorar la función objetivo del agente B en un 40% y, sin embargo, solo empeorar de media un 7% los resultados del primer conjunto de trabajos. Por otro lado, al tener un porcentaje tan bajo de resolución, no parece viable resolver este tipo de instancias de manera exacta con Gurobi, pues llevaría consigo grandes recursos de tiempo, fundamentales en el día a día de las empresas.

Con respecto a las instancias pequeñas, con 5 o 10 trabajos, de nuevo, tal y como ocurría en los escenarios anteriores, Gurobi es capaz de resolverlas de forma exacta en períodos de tiempo muy cortos. Sin embargo, el aumento de la función objetivo del primer agente en las instancias con N=5 es superior al 50% con respecto a la situación original de delta=0, por lo que supera lo que se reduce en el segundo agente. Si se analizan aquellas que tienen un número de trabajos igual a 10, el incremento de los valores del makespan es menor al 30%, por lo que estas instancias son las más adecuadas para resolverlas de forma exacta en este escenario, ya que tampoco sufren un gran incremento en los tiempos de resolución.

	delta=0.4	
	Suma	Porcentaje
TOTAL	68/160	42,50%
N=5	0/40	0,00%
N=10	0/40	0,00%
N=15	28/40	70,00%
N=20	40/40	100,00%
M=2	18/40	45,00%
M=3	18/40	45,00%
M=4	18/40	45,00%
M=5	14/40	35,00%

Tabla 22. Número de instancias sin resolver para delta=0.4

	delta=0		delta=0.1		delta=0.2		delta=0.3		delta=0.4	
	F.O	t(s)	F.O	t(s)	F.O	t(s)	F.O	t(s)	F.O	t(s)
TOTAL	340,25	0,34	366,96	1,71	386,41	1,73	413,30	3,25	445,98	12,75
N=5	230,58	0,04	271,38	0,03	290,18	0,06	315,03	0,05	346,85	0,05
N=10	393,60	0,20	413,40	0,24	437,23	0,50	470,30	0,98	506,35	1,90
N=15	545,09	1,89	545,64	13,12	551,55	12,24	563,36	23,12	586,91	98,37
N=20	-	-	-	-	-	-	-	-	-	-
M=2	239,23	0,16	265,91	5,46	305,23	1,33	329,82	3,45	362,86	22,07
M=3	284,00	0,19	306,62	0,18	314,62	0,38	345,81	0,82	377,24	1,43
M=4	356,36	0,23	384,91	0,28	401,09	0,65	422,59	1,91	451,86	4,83
M=5	457,54	0,70	486,00	0,97	500,65	4,05	530,58	6,17	566,85	20,71

Tabla 23. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1, 0.2, 0.3 y 0.4

	DESVIACIÓN ESTÁNDAR CPU TIME	
	delta=0	delta=0.4
TOTAL	0,69	59,18
N=5	0,02	0,02
N=10	0,09	1,55
N=15	1,06	149,45
N=20	-	-
M=2	0,22	95,50
M=3	0,29	3,33
M=4	0,40	13,28
M=5	1,12	66,99

Tabla 24. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.4 con respecto a situación inicial.

Como se puede observar en la ilustración 30, el análisis con respecto al número de máquinas sigue la tendencia del resto de escenarios, con un incremento grande de empeoramiento de la función objetivo del agente A en el caso del menor número de máquinas, sobre todo en el cruce con N=5. En el caso de un mayor número de máquinas, el incremento del makespan se va reduciéndose progresivamente, pero sin una clara influencia, a diferencia a lo que ocurre en el caso del número de trabajos.

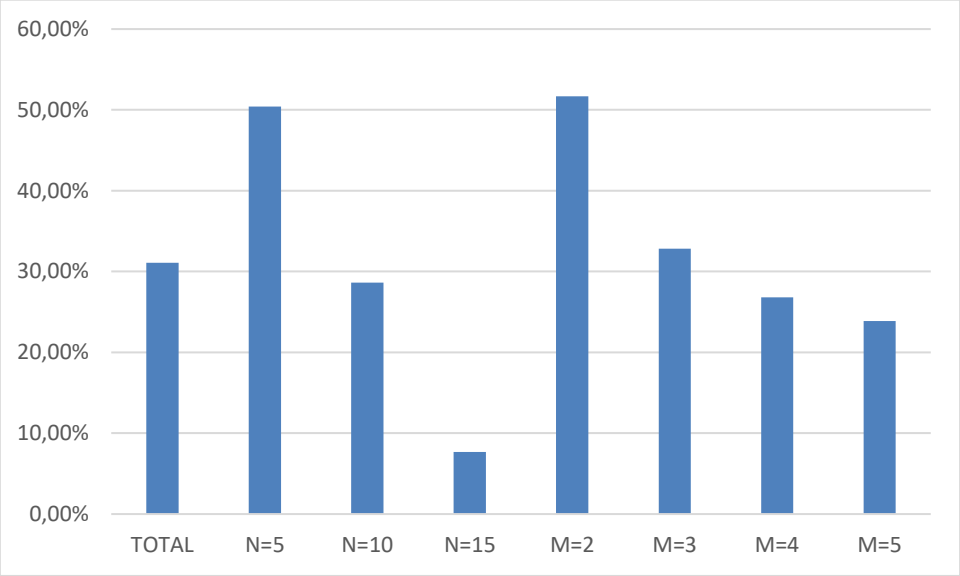

Ilustración 30. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,4 con respecto a delta=0.

5.7. ESCENARIO 6. DELTA=0.5

En este apartado se analiza el último escenario estudiado en este proyecto, esto es, el caso más restrictivo de los seis analizados, ya que los valores de épsilon se reducen un 50% con respecto a la situación original.

Como es de esperar, una gran cantidad de instancias no pueden ser resueltas por Gurobi en el tiempo límite fijado de 900 segundos, y es que el programa no ha encontrado la solución óptima en el 85% de los casos con N=15, por lo que, al igual que en el apartado 5.6, no es viable resolver dichas instancias de manera exacta con este método. Además, tal y como se puede comprobar en la tabla 26, aquellas que son resueltas tienen un tiempo de resolución alto en comparación al resto de escenarios.

Las instancias pequeñas son resueltas de manera muy eficiente con tiempos de resolución muy bajos, sin embargo, tal y como ocurre en los apartados anteriores, son éstas las que

tienen un aumento mayor en la función objetivo del primer agente. Cabe destacar el caso de aquellas instancias con 2 máquinas, donde el aumento es superior al 60%.

Para el caso de N=10, el porcentaje de empeoramiento con respecto a la situación original es de menos del 40% y los tiempos de resolución son muy bajos, por lo que se puede confirmar, al igual que en el apartado anterior, que son las instancias más favorables para resolver en este caso.

La tendencia de los resultados obtenidos al variar el número de máquinas es la misma que en los escenarios anteriores, siendo las instancias con M=2 la que obtienen mayores incrementos en la función objetivo global de los modelos, tal y como se comentó en apartados precedentes. En el caso de la existencia de un mayor número de máquinas. Dicho porcentaje se reduce, siendo M=4 el más favorable, con un incremento del 30% con respecto al escenario delta=0. Por último, cabe destacar que el número de instancias no resueltas en función del número de máquinas es similar en los cuatro casos, por lo que no es un parámetro importante en este análisis.

	delta=0.5	
	Suma	Porcentaje
TOTAL	74/160	46,25%
N=5	0/40	0,00%
N=10	0/40	0,00%
N=15	34/40	85,00%
N=20	40/40	100,00%
M=2	19/40	47,50%
M=3	20/40	50,00%
M=4	18/40	45,00%
M=5	17/40	42,50%

Tabla 25. Número de instancias sin resolver para delta=0.5

	delta=0		delta=0.1		delta=0.2		delta=0.3		delta=0.4		delta=0.5	
	F.O	t (s)	F.O	t (s)	F.O	t (s)	F.O	t (s)	F.O	t (s)	F.O	t (s)
TOTAL	326,07	0,25	354,62	0,34	374,89	0,51	403,19	1,71	436,42	3,39	455,20	15,11
N=5	230,58	0,04	271,38	0,03	290,18	0,06	315,03	0,05	346,85	0,05	349,60	0,05
N=10	393,60	0,20	413,40	0,24	437,23	0,50	470,30	0,98	506,35	1,90	539,73	2,22
N=15	521,50	2,00	522,17	8,47	525,17	4,44	540,00	19,62	560,00	37,94	594,67	237,93
N=20	-	-	-	-	-	-	-	-	-	-	-	-
M=2	227,33	0,12	255,19	1,66	296,38	0,61	321,33	1,06	353,52	1,72	371,48	12,26
M=3	273,95	0,13	297,70	0,13	306,10	0,20	338,85	0,46	371,40	0,74	398,10	0,96
M=4	356,36	0,23	384,91	0,28	401,09	0,65	422,59	1,91	451,86	4,83	462,23	34,80
M=5	440,95	0,53	473,27	0,82	488,45	0,80	519,32	3,88	558,59	6,75	580,36	21,63

Tabla 26. Comparativa de Media de makespan y CPU time para las instancias resueltas para delta=0, 0.1, 0.2, 0.3, 0.4 y 0.5.

	DESVIACIÓN ESTÁNDAR CPU TIME	
	delta=0	delta=0.5
TOTAL	0,58	83,58
N=5	0,02	0,03
N=10	0,09	1,51
N=15	1,24	253,28
N=20	-	-
M=2	0,15	50,71
M=3	0,11	1,25
M=4	0,40	143,98
M=5	1,02	79,83

Tabla 27. Comparación desviación estándar CPU time de resolución para las instancias resueltas en escenario delta=0.1 con respecto a situación inicial.

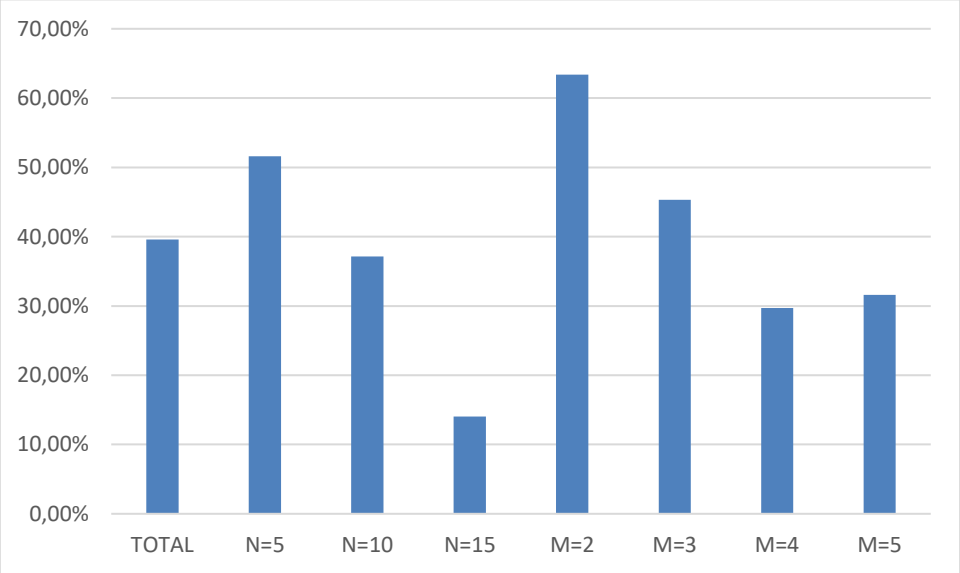

Ilustración 31. Porcentaje de aumento del makespan de las instancias resueltas con delta=0,4 con respecto a delta=0.

5.8. RESUMEN DE LOS ESCENARIOS

Una vez analizados de forma independiente todos los escenarios estudiados y resueltos de manera exacta mediante Gurobi, en este apartado se aportan diversas tablas y una gráfica en la cual se resumen los resultados obtenidos en los apartados anteriores, con el objetivo de obtener una mejor comprensión de estos.

En la tabla 28 se reflejan los porcentajes de instancias resueltas en cada uno de los escenarios estudiados en función de cada parámetro. Por otro lado, en la tabla 29 se puede observar el

porcentaje de aumento del makespan y el incremento del CPU time (en segundos) de todos los escenarios con respecto a delta=0, excluyendo las instancias no resueltas en cada caso.

Por último, en la ilustración 32 se pueden comprobar de forma gráfica los porcentajes de aumento del makespan mencionados de la tabla 29. Cabe destacar los resultados en M=2, ya que son los que presentan los mayores porcentajes de aumento con respecto a delta=0, siendo dichas instancias las más desfavorables en este aspecto.

	INSTANCIAS SIN RESOLVER					
	delta=0	delta=0.1	delta=0.2	delta=0.3	delta=0.4	delta=0.5
TOTAL	0,00%	13,13%	30,63%	36,88%	42,50%	46,25%
N=5	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
N=10	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
N=15	0,00%	7,50%	22,50%	47,50%	70,00%	85,00%
N=20	0,00%	45,00%	100,00%	100,00%	100,00%	100,00%
M=2	0,00%	25,00%	37,50%	40,00%	45,00%	47,50%
M=3	0,00%	12,50%	30,00%	42,50%	45,00%	50,00%
M=4	0,00%	7,50%	30,00%	32,50%	45,00%	45,00%
M=5	0,00%	7,50%	25,00%	32,50%	35,00%	42,50%

Tabla 28. Porcentaje de instancias resueltas en todos los escenarios.

	delta=0.1		delta=0.2		delta=0.3		delta=0.4		delta=0.5	
	F.O	Δt	F.O	Δt	F.O	Δt	F.O	Δt	F.O	Δt
TOTAL	4,66%	23,42	11,94%	27,04	20,34%	23,16	31,07%	12,41	39,60%	14,87
N=5	17,69%	-0,01	25,85%	0,02	36,63%	0,00	50,43%	0,01	51,62%	0,00
N=10	5,03%	0,04	11,08%	0,30	19,49%	0,78	28,65%	1,70	37,13%	2,02
N=15	1,33%	44,40	4,72%	96,40	7,38%	114,22	7,67%	96,48	14,03%	235,93
N=20	0,10%	49,94	-	-	-	-	-	-	-	-
M=2	7,98%	55,35	24,08%	19,90	36,14%	18,92	51,68%	21,91	63,41%	12,14
M=3	3,84%	20,88	10,64%	45,58	20,69%	29,51	32,83%	1,25	45,32%	0,84
M=4	4,31%	17,90	10,75%	35,28	16,75%	33,00	26,80%	4,60	29,71%	34,57
M=5	3,85%	11,04	8,17%	7,98	15,78%	11,64	23,89%	20,01	31,62%	21,10

Tabla 29. Porcentaje de aumento del makespan y del CPU time de todos los escenarios con respecto a delta=0.

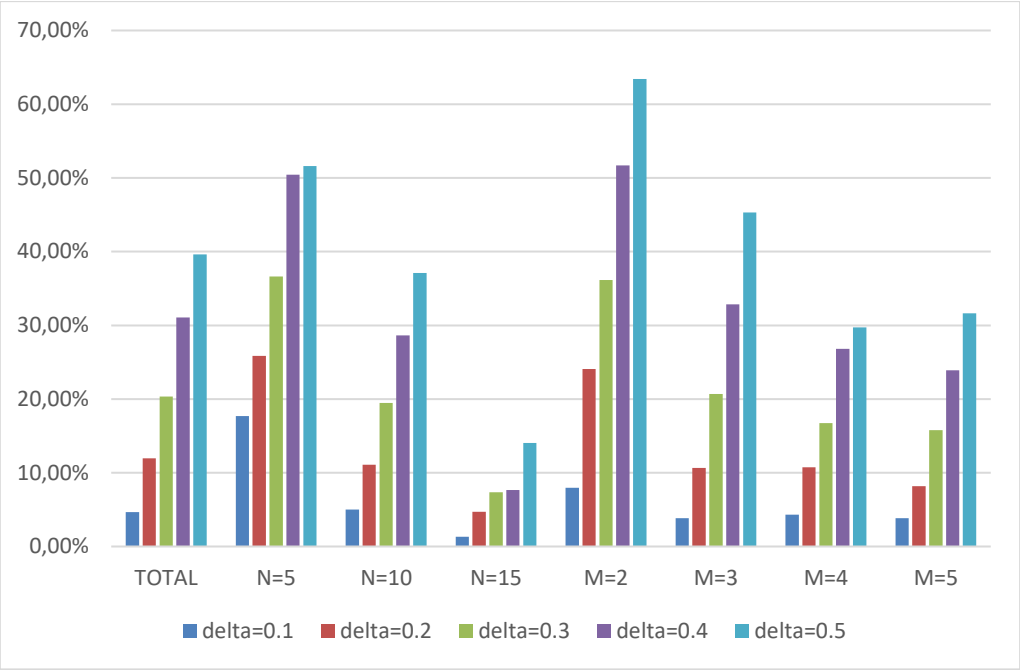

Ilustración 32. Porcentaje de aumento del makespan de todos los escenarios con respecto a delta=0.

CAPÍTULO 6.

CONCLUSIONES.

En este proyecto se ha hecho un estudio de los problemas de programación multi agente (MASP, "multi-agent scheduling problem"), elaborándose una revisión de la literatura para conocer el alcance de los estudios y análisis realizados hasta el momento. Una de las conclusiones que se pueden obtener de dicha revisión es la ausencia de estudios acerca de la resolución de forma exacta de este tipo de problemas, debido fundamentalmente a los grandes tiempos de resolución necesarios para ello. Sin embargo, es útil conocer el comportamiento de dichos modelos en instancias pequeñas, con pocos trabajos y pocas máquinas. Por tanto, en el capítulo 4 se ha realizado la resolución de forma exacta mediante Gurobi de diversos modelos existentes en la literatura adaptados a un problema multiagente escogido entre los estudiados en la literatura (épsilon-constraint, que consiste en minimizar el makespan del agente A sujeto a una restricción superior para la tardanza total del agente B). Dicho proceso se ha realizado para escoger el modelo más eficiente entre SGST, TS2 y TS3, siendo el primero de ellos el que presenta mejores resultados en términos de CPU. Además, se observó que N es el parámetro más influyente en los tiempos de resolución de cada instancia.

Posteriormente, en el capítulo 5, se hace un estudio de la influencia del parámetro épsilon en los resultados obtenidos con el modelo SGST, creando diversos escenarios variando el parámetro delta para originar instancias con valores de épsilon más reducidos. Una vez analizados todos los escenarios estudiados y resueltos de manera exacta mediante Gurobi, se pueden obtener una serie de conclusiones que se exponen a continuación.

En primer lugar, hay que destacar que si lo que se desea es priorizar la función objetivo makespan del primer agente en detrimento de la tardanza total del segundo conjunto de trabajos, es muy eficiente resolver las instancias estudiadas en este proyecto (de hasta 20 trabajos y 5 máquinas) de manera exacta con Gurobi y con el método para calcular épsilon propuesto, puesto que todas ellas son resueltas en tiempos de resolución bajos.

Por otro lado, se puede confirmar con los estudios realizados que dicho método aporta unos valores de épsilon algo holgados con respecto a las fechas de entrega. Por tanto, se han realizado diversas modificaciones en dicho valor para hacer el modelo más restrictivo, con la desventaja de que dicha mejora de la función objetivo del segundo agente perjudicará el makespan del primer conjunto de trabajos. Por tanto, según sean las necesidades a la hora de satisfacer las funciones objetivos en la elaboración de un programa, se han estudiado diversos escenarios para comprobar qué resultados pueden arrojar.

En dichos escenarios se ha comprobado que solo en el primero de ellos, en el cual se reduce un 10% los valores del makespan con respecto al método original propuesto en el apartado 4.4, Gurobi es capaz de resolver las instancias con N=20. Por tanto, en el resto de los

escenarios, con una reducción mayor de los valores de épsilon, hay que recurrir a los métodos aproximados propuestos en la literatura para resolverlos.

Para las instancias con 15 trabajos, los resultados obtenidos son similares, puesto que a pesar de que las que son resueltas dentro del límite de tiempo fijado, tienen un porcentaje de aumento del makespan muy pequeño, a medida que delta aumenta, el número de éstas disminuye, llegando a alcanzar el 85% el porcentaje de las instancias no resueltas con N=15 en el caso de delta=0.5. Por tanto, para este tipo de instancias y en escenarios con valores de delta altos, también habría que recurrir a métodos aproximados, permitiéndose la resolución de forma exacta para escenarios con una poca reducción de los valores de épsilon con respecto al método original.

Por otro lado, las instancias con N=10 presentan resultados excelentes, ya que todas ellas son resueltas con CPU time muy bajos en todos los escenarios estudiados. Además, presentan un porcentaje de empeoramiento del makespan inferior a lo que se mejora en la tardanza total del segundo agente al reducir épsilon. Por tanto, se puede concluir que dichas instancias pueden ser resueltas de manera exacta con Gurobi de una forma muy eficiente, no teniendo que recurrir a soluciones aproximadas.

Además, si las instancias con N=5 son analizadas, se puede observar que el porcentaje de empeoramiento del makespan es superior a lo que se mejora en la tardanza total, por lo que, a pesar de que son resueltas en tiempos muy bajos, presentan malos resultados si lo que se desea es priorizar el makespan.

Con respecto al número de máquinas, se observa que cuando dicho valor es bajo, la situación es similar a lo explicado anteriormente, ya que, el porcentaje de empeoramiento del makespan es superior a lo que se mejora en la función objetivo del segundo agente, por lo que, para este tipo de instancias, deben ser resueltas con este método cuando se priorice la tardanza total sin importar mucho el valor del makespan del primer conjunto de trabajos.

Para el resto de valores del número de máquinas, las tendencias en los tiempos de resolución son irregulares y no se puede encontrar un patrón, por lo que se llega a la conclusión de que no es un parámetro importante al analizar las soluciones.

En términos totales, el grado de empeoramiento del makespan del primer conjunto de trabajos es inferior al porcentaje de mejora de la tardanza total del segundo agente, por lo que se concluye que en general, reducir los valores de épsilon no repercuten en demasía en los valores de la función objetivo original.

Por último, es evidente indicar que en todos los casos aumenta el porcentaje de empeoramiento de la función objetivo con respecto aumentamos delta, puesto que una mejora del total tardiness repercutirá en mayor o menor medida en un aumento del makespan del agente A, como se ha indicado en los párrafos anteriores.

Todas estas afirmaciones acerca de la influencia del parámetro épsilon en los resultados obtenidos se pueden consultar en las tablas 28 y 29, y de forma gráfica en la ilustración 32, donde se encuentra un resumen de los datos generados en los estudios realizados.

ANEXO.

Modelo SGST formato LP. EJEMPLO.

```
Minimize
CMAX
Subject to
T2 + T3 <= 21
CMAX - C10 >= 0
CMAX - C11 >= 0
T2 - C12 >= - 2
T3 - C13 >= - 6
C00 >= 5
C01 >= 2
C02 >= 1
C03 >= 3
C10 - C00 >= 2
C11 - C01 >= 4
C12 - C02 >= 4
C13 - C03 >= 3
C00 - C01 + 24 D01 >= 5
C00 - C02 + 24 D02 >= 5
C00 - C03 + 24 D03 >= 5
C01 - C02 + 24 D12 >= 2
C01 - C03 + 24 D13 >= 2
C02 - C03 + 24 D23 >= 1
C10 - C11 + 24 D01 >= 2
C10 - C12 + 24 D02 >= 2
C10 - C13 + 24 D03 >= 2
C11 - C12 + 24 D12 >= 4
C11 - C13 + 24 D13 >= 4
C12 - C13 + 24 D23 >= 4
C01 - C00 - 24 D01 >= -22
C02 - C00 - 24 D02 >= -23
C03 - C00 - 24 D03 >= -21
C02 - C01 - 24 D12 >= -23
C03 - C01 - 24 D13 >= -21
C03 - C02 - 24 D23 >= -21
C11 - C10 - 24 D01 >= -20
C12 - C10 - 24 D02 >= -20
C13 - C10 - 24 D03 >= -21
C12 - C11 - 24 D12 >= -20
C13 - C11 - 24 D13 >= -21
C13 - C12 - 24 D23 >= -21
Generals
CMAX C00 C01 C02 C03 C10 C11 C12 C13 T2 T3
Binary
D01 D02 D03 D12 D13 D23
End
```

CÓDIGO SGST

```
#include <stdio.h>
#include <stdlib.h>
#include<schedule_lib.h>
int FSmakespan(int n,int m,VECTOR_INT secuencia,MAT_INT T);
int tardanzatotalsegundogrupo(int n,int m,int na,VECTOR_INT secuencia,MAT_INT T,
MAT_INT D);
int makespanprimergrupo(int n,int m,int na,VECTOR_INT secuencia,MAT_INT T);
int main(int argc, char *argv[])
{
   //1. DECLARAMOS VARIABLES.
int a; // índice para el cálculo del número de posibles combinaciones de inserción.
int i; // índice para las filas de matriz de tiempos de procesado.
int j; // índice para las columnas de matriz de tiempos de procesado.
int q; // índice para recorrer el vector secuencia en el algoritmo NEH.
int w; // índice para realizar todas las posibles combinaciones de inserción de un trabajo.
int m = 0; // número de maquinas
int n = 0; // número de trabajos
int suma; // tiempo por trabajo
int makespanaux; // makespan auxiliar
int makmax; //número muy alto de makespan para comparar en la primera combinación del
algoritmo NEH.
MAT_INT T; // matriz de tiempos de procesado
MAT_INT D; //matriz de fechas de entrega
int x;
VECTOR_INT secuencia; // secuencia ordenada descendente
VECTOR_INT vtiempos; // vector de tiempos medios
VECTOR_INT vini; // vector auxiliar para hallar la secuencia mediante makespan
VECTOR_INT secaux; // vector auxiliar para hallar la secuencia mediante makespan
VECTOR_INT control; // vector auxiliar auxiliar para hallar la secuencia mediante makespan
VECTOR_INT NEHA;  // vector secuencia optima del algoritmo NEHa
VECTOR_INT NEHB;  // vector secuencia optima del algoritmo NEHb
int epsilon; //valor de epsilon.
int r; //índice para recorrer numero de máquinas
//2.....LEEMOS DATOS.....
D=loadPTimes_nrows(argv[2],&n,&x,YES);
T=loadPTimes_nrows(argv[1],&n,&m,YES);
printf("\n");
int na=n/2; //número de trabajos del agente A.
int nb=n-na; //número de trabajos del agente B.
//3.....CALCULAMOS SUMA DE TIEMPOS DE PROCESADO.....
vtiempos = DIM_VECTOR_INT(na);
for(i=0;i<na;i++)
 {
 suma=0;
 for (j=0;j<m;j++)
 {
 suma+=T[j][i];
 }
```

```
 vtiempos[i]=suma;
 }
printf("El vector de suma de tiempos del primer conjunto de trabajos es:");
print_int_vector(vtiempos,na);

 //4.....................................OBTENEMOS SECUENCIA ORDENADA DESCENDENTE.....................
 secuencia = DIM_VECTOR_INT(na);
 sortLVector(vtiempos,secuencia,na,'D');  //ordena vtiempos de forma descendente y te da la
secuencia.
printf("La secuencia ordenada descendente del primer conjunto de trabajos es:");
print_int_vector(secuencia,na);
// 5.....................................CREAMOS UN VECTOR PARA IR CALCULANDO SECUENCIA.....................
 vini = DIM_VECTOR_INT(1);
 vini[0]=secuencia[0];

 //6..................................... ITERACIÓN DEL ALGORITMO NEH.....................................
 for (q=1; q<na; q++)
 {
 a=q+1; //Por ejemplo en la primera iteración, tenemos 1 trabajo que puede ser insertado en
2 posiciones. 1 más que trabajos haya ya fijos.
 makmax=100000000000;
control = DIM_VECTOR_INT(a);
 secaux = DIM_VECTOR_INT(a);
 for (w=0; w<a; w++)
 {
 copyIVector(vini,secaux,q);
insertIVector(secaux,a,secuencia[q],w); //Se inserta en secaux el trabajo secuencia[q] en la
posición w. a es el tamaño del nuevo vector generado
makespanaux=FSmakespan(a,m,secaux,T); //calculamos el makespan de esa secuencia.
 if (makespanaux < makmax)
 {
 makmax = makespanaux;
copyIVector(secaux,control,a);
 }
 }
free(secaux);
 free(vini);
 vini = DIM_VECTOR_INT(a);
 copyIVector(control,vini,a);
if (a==na) //paramos la iteración. ya se han probado todas las posibilidades.
 {
 NEHA = DIM_VECTOR_INT(na);
 copyIVector(control,NEHA,na);
 }
 free(control);
 }

// 7.....................................RESULTADOS.....................................
 printf("\nEl makespan de la secuencia de trabajos NEHAes %d", makmax);
 printf("\nLa secuencia solucion del algoritmo NEHA es:");
 print_int_vector(NEHA,na);
 printf("\n");
```

```
 free(secuencia);
 free(vtiempos);
 free(vini);
 free(secaux);

//.............................EMPEZAMOS CON EL SEGUNDO CONJUNTO DE TRABAJOS.....................
vtiempos = DIM_VECTOR_INT(nb);
int k=0;
for(i=na;i<n;i++)
 {
 suma=0;
 for (j=0;j<m;j++)
 {
 suma+=T[j][i];
 }
 vtiempos[k]=suma;
 k+=1;
 }
printf("El vector de suma de tiempos del segundo conjunto de trabajos es:");
print_int_vector(vtiempos,nb);

 secuencia = DIM_VECTOR_INT(nb);
 sortLVector(vtiempos,secuencia,nb,'D');
 printf("La secuencia ordenada descendente del segundo conjunto de trabajos es:");
print_int_vector(secuencia,nb);

for(i=0;i<nb;i++)
{
   secuencia[i]=secuencia[i]+na;
}
printf("La secuencia ordenada descendente del segundo conjunto de trabajos en variables
globales es:");
print_int_vector(secuencia,nb);
printf("\n");

vini = DIM_VECTOR_INT(na+1);
copyIVector(NEHA,vini,na);
insertIVector(vini,na+1,secuencia[0],na);
for (q=1; q<nb; q++)
 {
 a=q+1; //Por ejemplo en la primera iteración, tenemos 1 trabajo que puede ser insertado en
2 posiciones. 1 más que trabajos haya ya fijos.
 makmax=100000000000;
control = DIM_VECTOR_INT(na+a);
 secaux = DIM_VECTOR_INT(na+a);
 for (w=0; w<a; w++)
 {
 copyIVector(vini,secaux,na+q);
insertIVector(secaux,na+a,secuencia[q],na+w); //Se inserta en secaux el trabajo secuencia[q]
en la posición w. a es el tamaño del nuevo vector generado
makespanaux=FSmakespan(na+a,m,secaux,T); //calculamos el makespan de esa secuencia.
```

```
 if (makespanaux < makmax)
 {
 makmax = makespanaux;
copyIVector(secaux,control,na+a);
 }
 }
free(secaux);
 free(vini);
 vini = DIM_VECTOR_INT(na+a);
 copyIVector(control,vini,na+a);
if (a==nb) //paramos la iteración. ya se han probado todas las posibilidades.
 {
 NEHB = DIM_VECTOR_INT(na+a);
 copyIVector(control,NEHB,na+a);
 }
 free(control);
 }

 printf("\nEl makespan de la secuencia de trabajos es %d", makmax);
 printf("\nLa secuencia solucion del algoritmo NEHB es: ");
 for(i=na;i<n;i++)
 {

    printf("%d ",NEHB[i]);
 }
 printf("\n");
printf("\nLa secuencia TOTAL obtenida de NEHA Y NEHB es:");
 print_int_vector(NEHB,n);
epsilon=tardanzatotalsegundogrupo(n,m,na,NEHB,T,D);
printf("\nEl epsilon es %d",epsilon);


/////.....................ESCRIBIMOS EL MODELO.....................................
FILE * fichero=fopen(argv[3],"w");
int P=0;
for(r=0;r<m;r++)
{
   for(i=0;i<n;i++)
   {
      P=P+T[r][i];
   }
}
fprintf(fichero,"Minimize\n");
fprintf(fichero,"CMAX\n");
fprintf(fichero,"Subject to\n");
for(i=na;i<n;i++)
{
      if (i==n-1)
      fprintf(fichero,"T%d <= %d\n",i,epsilon);
      else
         fprintf(fichero,"T%d + ",i);
}
```

```
for(i=0;i<na;i++)
{
      fprintf(fichero,"CMAX - C%d%d >= 0\n",m-1,i);
}
for(i=na;i<n;i++)
{
      fprintf(fichero,"T%d - C%d%d >= - %d\n",i,m-1,i,D[0][i]);
}
for(i=0;i<n;i++)
{
   fprintf(fichero,"C%d%d >= %d\n",0,i,T[0][i]);
}
for(r=0;r<m-1;r++)
{
   for(i=0;i<n;i++)
   {
        fprintf(fichero,"C%d%d - C%d%d >= %d\n",r+1,i,r,i,T[r+1][i]);
   }
}
for(r=0;r<m;r++)
{
   for(i=0;i<n;i++)
   {
      for(k=i+1;k<n;k++)
       fprintf(fichero,"C%d%d - C%d%d + %d D%d%d >= %d\n",r,i,r,k,P,i,k,T[r][i]);
   }
}
for(r=0;r<m;r++)
{
   for(i=0;i<n;i++)
   {
      for(k=i+1;k<n;k++)
       fprintf(fichero,"C%d%d - C%d%d - %d D%d%d >= %d\n",r,k,r,i,P,i,k,T[r][k]-P);
   }
}


fprintf(fichero,"Generals\n");
fprintf(fichero,"CMAX ");
for(r=0;r<m;r++)
{
   for(i=0;i<n;i++)
   {
      fprintf(fichero,"C%d%d ",r,i);
      }
   }

   for(i=na;i<n;i++)
   {
      fprintf(fichero,"T%d ",i);
      }
fprintf(fichero,"\nBinary\n");
```

```
for(i=0;i<n;i++)
{
   for(k=i+1;k<n;k++)
   {
      fprintf(fichero,"D%d%d ",i,k);
   }
}
fprintf(fichero,"\nEnd");
}
```

```
/////F     U     N     C     I     O     N     E     S

int FSmakespan(int n,int m,VECTOR_INT secuencia,MAT_INT T)
{
int makespan;
int i; int j;
MAT_INT ct;
ct=DIM_MAT_INT(m,n);
//calculamos la finalización del primer trabajo de la secuencia en la primera máqquina.
ct[0][0] = T[0][secuencia[0]];
//calculamos la finalización del primer trabajo de la secuencia en el resto de máquinas.
for(i=1;i<m;i++)
{
    ct[i][0] = ct[i-1][0] + T[i][secuencia[0]];
}
//calculamos la finalización de todos los trabajos en la primera máquina.
 for(j=1;j<n;j++)
{
    ct[0][j] = ct[0][j-1] +  T[0][secuencia[j]];
 }
//calculamos la finalización de todos los trabajos en el resto de máquinas.
for(j=1;j<n;j++)
   {
for(i=1;i<m;i++)
{
  ct[i][j] =iMax(ct[i][j-1], ct[i-1][j])+T[i][secuencia[j]] ;
}
}
//calculamos el makespan.
makespan=ct[m-1][n-1];
return makespan;
}
```

```
int tardanzatotalsegundogrupo(int n,int m,int na,VECTOR_INT secuencia,MAT_INT T,
MAT_INT D)
{
int tardanzaj=0;
```

```
int i; int j;
MAT_INT ct;
ct=DIM_MAT_INT(m,n);
//calculamos la finalización del primer trabajo de la secuencia en la primera máqquina.
ct[0][0] = T[0][secuencia[0]];
//calculamos la finalización del primer trabajo de la secuencia en el resto de máquinas.
for(i=1;i<m;i++)
{
     ct[i][0] = ct[i-1][0] + T[i][secuencia[0]];
}
//calculamos la finalización de todos los trabajos en la primera máquina.
 for(j=1;j<n;j++)
{
     ct[0][j] = ct[0][j-1] +  T[0][secuencia[j]];
 }
//calculamos la finalización de todos los trabajos en el resto de máquinas.
for(j=1;j<n;j++)
   {
for(i=1;i<m;i++)
{
  ct[i][j] =iMax(ct[i][j-1], ct[i-1][j])+T[i][secuencia[j]] ;
}
   }
//calculamos la tardanza del segundo conjunto de trabajos
for(j=0;j<n;j++)
{
   if(secuencia[j]>=na && ct[m-1][j]>D[0][secuencia[j]])
{
       tardanzaj=tardanzaj+(ct[m-1][j]-D[0][secuencia[j]]);
   }
}
return tardanzaj;
}

int makespanprimergrupo(int n,int m,int na,VECTOR_INT secuencia,MAT_INT T)
{
int i; int j; int makespanaux=0;
MAT_INT ct;
ct=DIM_MAT_INT(m,n);
//calculamos la finalización del primer trabajo de la secuencia en la primera máqquina.
ct[0][0] = T[0][secuencia[0]];
//calculamos la finalización del primer trabajo de la secuencia en el resto de máquinas.
for(i=1;i<m;i++)
{
     ct[i][0] = ct[i-1][0] + T[i][secuencia[0]];
}
//calculamos la finalización de todos los trabajos en la primera máquina.
 for(j=1;j<n;j++)
{
     ct[0][j] = ct[0][j-1] +  T[0][secuencia[j]];
 }
//calculamos la finalización de todos los trabajos en el resto de máquinas.
```

```
for(j=1;j<n;j++)
   {
for(i=1;i<m;i++)
{
  ct[i][j] =iMax(ct[i][j-1], ct[i-1][j])+T[i][secuencia[j]] ;
}
}
//calculamos la tardanza del segundo conjunto de trabajos
for(j=0;j<n;j++)
{
   if((secuencia[j]<na) && (ct[m-1][j]>makespanaux))
   {
    makespanaux=ct[m-1][j];
   }
}
return makespanaux;
}
```

CÓGIDO CREAR INSTANCIAS.

En este código se refleja cómo crear instancias con R=0.2 Y T=0.6 y con el parámetro de p como la suma de los tiempos de proceso de los trabajos del agente B.

```
#include <stdio.h>
#include <stdlib.h>
#include<schedule_lib.h>
int main(int argc, char *argv[])
{
MAT_INT T; // matriz de tiempos de procesado

int n,m,i,j;
T=loadPTimes_nrows(argv[1],&n,&m,YES);
MAT_INT D[n][1]; //matriz de fechas de entrega
int p=0;
int na=n/2;
for(i=na;i<n;i++)
 {
 for (j=0;j<m;j++)
 {
 p+=T[j][i];
 }
 }
 printf("%d\n",p);
 float a,b,r,t;
r=0.2;
t=0.6;
 printf("\n\n\n%f\n",t);
 printf("%f\n",r);
```

```
 a=p*(1-t-(r/2));
 b=p*(1-t+(r/2));
 printf("%f\n",a);
 printf("%f\n",b);
 int intervaloinferior=b-a+1;
 int intervalosuperior=a;
srand(time(NULL));
int nMaquinas=1;
for(i=0;i<n;i++)
{
for(j=0;j<nMaquinas;j++)
{
   D[i][j]=(rand()% intervaloinferior + intervalosuperior);
}

}
FILE * fichero=fopen(argv[2],"w");
 fprintf(fichero,"%d\n",m);
  fprintf(fichero,"%d\n",n);
  for(i=0;i<n;i++)
  {
      fprintf(fichero,"%d\n",D[i][0]);
  }
}
```

CÓDIGO EXPORTAR RESULTADOS FUNCIÓN OBJETIVO.

En este programa se exportan los resultados de Gurobi (valor de la función objetivo del modelo, makespan del agente A) a un fichero. Para ello se hace uso de una batería que carga sucesivamente todos los ficheros.sol de Gurobi.

```
#include <stdio.h>
#include <stdlib.h>
#include<schedule_lib.h>
int main(int argc, char *argv[])
{
   FILE * fichero=fopen(argv[1],"r");
   FILE * resultados=fopen(argv[2],"a");
   fseek(fichero,20,SEEK_SET);
   int b;
   fscanf(fichero,"%d",&b);
   fprintf(resultados,"%d\n",b);
}
```

BIBLIOGRAFÍA.

Ben Ltayef, A., Loukil, T., & Teghem, J. (2009). Rescheduling a permutation flowshop problems under the arrival a new set ofjobs. In 2009 International conference on computers and industrial engineering, CIE 2009 (pp. 188–192).

Baker KR, Smith JC (2003) A multiple-criterion model for machine scheduling

Agnetis A, Mirchandani PB, Pacciarelli D, Pacifici A (2004) Scheduling problems with two competing agents.

Cheng, T. C. E., Ng, C. T., & Yuan, J. J. (2006). Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. Theoretical Computer Science.

Framiñán Torres, J. M. 2018. Librería Schedule en C para Programación de Operaciones. Escuela Técnica Superior de Ingeniería. Universidad de Sevilla.

V.T. Kindt., & J.-C Billaut. (2002). Theory, Models and Algorithms, Second Edi. Springer Berlin Heidelberg, 2002.

Ahmadi-Darani, M. H., Moslehi, G., & Reisi-Nafchi, M. (2018). A two-agent scheduling problem in a two-machine flowshop. *International Journal of Industrial Engineering Computations*, *9*(3), 289–306. https://doi.org/10.5267/j.ijiec.2017.8.005

Ameneiros, G. (2018). *Autor: Gonzalo Ameneiros Martínez Tutora: Paz Pérez González*.

Fan, B. Q., & Cheng, T. C. E. (2016). Two-agent scheduling in a flowshop. *European Journal of Operational Research*, *252*(2), 376–384. https://doi.org/10.1016/j.ejor.2016.01.009

Fernandez-Viagas, V., & Framinan, J. M. (2019). A best-of-breed iterated greedy for the permutation flowshop scheduling problem with makespan objective. *Computers and Operations Research*, *112*(August). https://doi.org/10.1016/j.cor.2019.104767

Lee, W. C., Chen, S. K., Chen, C. W., & Wu, C. C. (2011). A two-machine flowshop problem with two agents. *Computers and Operations Research*, *38*(1), 98–104. https://doi.org/10.1016/j.cor.2010.04.002

Lei, D. (2015). Variable neighborhood search for two-agent flow shop scheduling problem. *Computers and Industrial Engineering*, *80*, 125–131. https://doi.org/10.1016/j.cie.2014.11.024

Luo, W., Chen, L., & Zhang, G. (2012). Approximation schemes for two-machine flow shop scheduling with two agents. *Journal of Combinatorial Optimization*, *24*(3), 229–239. https://doi.org/10.1007/s10878-011-9378-2

Mor, B., & Mosheiov, G. (2014). Polynomial time solutions for scheduling problems on a proportionate flowshop with two competing agents. *Journal of the Operational Research Society*, *65*(1), 151–157. https://doi.org/10.1057/jors.2013.9

Perez-Gonzalez, P., & Framinan, J. M. (2014). A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research*, *235*(1), 1–16.

https://doi.org/10.1016/j.ejor.2013.09.017

Pinedo, M. L. (2016). *Scheduling Theory, Algorithms, and Systems* (5th ed. 20). Springer International Publishing. https://doi.org/10.1007/978-3-319-26580-3

Shiau, Y. R., Tsai, M. S., Lee, W. C., & Cheng, T. C. E. (2015). Two-agent two-machine flowshop scheduling with learning effects to minimize the total completion time. *Computers and Industrial Engineering*, *87*, 580–589. https://doi.org/10.1016/j.cie.2015.05.032

Stafford, E. F., Tseng, F. T., & Gupta, J. N. D. (2005). Comparative evaluation of MILP flowshop models. *Journal of the Operational Research Society*, *56*(1), 88–101. https://doi.org/10.1057/palgrave.jors.2601805

Tseng, F. T., & Stafford, E. F. (2008). New MILP models for the permutation flowshop problem. *Journal of the Operational Research Society*, *59*(10), 1373–1386. https://doi.org/10.1057/palgrave.jors.2602455

L.Graham, E.L.Lawler, J.K.Lenstra & A.H.G.Rinnooy Kan (1979). *Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey.*