Trabajo Fin de Grado

Grado en Ingeniería de Organización Industrial

“PROGRAMACIÓN DE TAREAS CON RECURSOS LIMITADOS EN ESTACIONES DE MONTAJE AERONÁUTICAS MEDIANTE PROGRAMACIÓN LINEAL”

Autor: Manuel Sayago Tinoco
Tutor: Ignacio Eguía Salinas

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

Sevilla, 2021

ORGANIZACIÓN INDUSTRIAL

Trabajo Fin de Grado

Grado en Ingeniería de Organización Industrial

“Programación de tareas con recursos limitados en estaciones de montaje aeronáuticas mediante programación lineal”

Autor:
Manuel Sayago Tinoco

Tutor:
Ignacio Eguía Salinas

Departamento de Ingeniería de Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado: “Programación de tareas con recursos limitados en estaciones de montaje aeronáuticas mediante programación lineal”

Autor: Manuel Sayago Tinoco

Tutor: Ignacio Eguía Salinas

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

Presidente:

Vocal/es:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En primer lugar, quiero agradecerle a mi tutor, Ignacio Eguía Salinas, el darme la oportunidad de realizar este gran trabajo, así como todo el esfuerzo y dedicación que ha depositado en elaboración del mismo y toda la ayuda prestada en cualquier momento.

También, darles las gracias, cómo no, a mis padres, mi hermano y mi pareja. A mis padres por ser apoyo fundamental en todos los aspectos de la vida, siempre a pie de cañón junto a mí sin dejarme caer y, si lo hacía, caían conmigo y me levantaban sin importar nada más.

A mi hermano, ingeniero aeroespacial, principal causante de haberme interesado en la ingeniería y hacerme ver en ella un mundo totalmente nuevo a descubrir. Por celebrar cada alegría como si fuese la suya y no dejarme solo en ninguno de los momentos más desagradables que pudiese haber comprendiéndome siempre.

A mi pareja Mónica, ingeniera química próximamente, quien ha recorrido a mi lado la última parte de este camino, quizás la más difícil para mí. Siendo apoyo incondicional en cada momento. Dándome ese empujoncito final tan necesario en ciertos momentos y consiguiendo que viese las cosas desde una mejor perspectiva. Confiando en mí más de lo que yo mismo lo hacía y haciéndome creer en que puedo conseguir todos y cada uno de mis sueños.

No puedo estar más feliz y agradecido de tener a estas personas en mi vida. Todo lo que he conseguido y consiga en un futuro es y será por y para ellos. Gracias.

Manuel Sayago Tinoco

Resumen

Los problemas de optimización, en los cuales hay una o varias funciones objetivos y una serie de restricciones que tienen que ser cumplidas, son muy diferentes entre sí y muy complicados de resolver. Su resolución permite llegar a la eficiencia en los diferentes casos en los que se puede aplicar.

Dentro del problema de programación de proyectos (*Project Scheduling Problem*, PSP) hay una variedad enorme de diferentes variantes. En este TFG, el ejemplo a resolver consiste en la programación de un conjunto de actividades asignadas a una estación de montaje donde existen recursos limitados, relaciones de precedencia entre las actividades y un tiempo máximo correspondiente al tiempo de ciclo de la línea de montaje.

Para obtener la solución del mismo, se tendrán una serie de actividades relacionadas entre sí mediante relaciones de precedencias, un conjunto de recursos limitados por su disponibilidad, una serie de restricciones de tiempo, coste y capacidad y unas medidas de desempeño. El objetivo es obtener el mejor resultado posible asignando los recursos disponibles a las respectivas actividades, de tal forma que se optimice cada una de las medidas de desempeño.

A la hora de resolver este problema, se pueden encontrar dos casos (ambos se explicarán en este trabajo). El primero de ellos es la programación de proyectos con recursos limitados con un único modo de resolución (*Resource Constrained Project Scheduling Problem*, RCPSP), es decir, las actividades solo se pueden realizar de una sola forma. En el segundo caso, cada actividad puede ejecutarse de distintos modos (*Multi-mode Resource Constrained Project Scheduling Problem*, MRCPSP) por lo que habría que calcular, además del instante de inicio de cada actividad, con qué modo es más conveniente. Las estaciones de montaje aeronáuticas tienen una particularidad asociada a la limitación del número de trabajadores máximo en cada zona de trabajo definida. Esta característica será incorporada al problema MRCPSP como un tercer caso de uso.

En este TFG se diseñarán y resolverán modelos de programación lineal para el problema de asignación de los recursos a las actividades de una forma óptima para los diferentes casos de estos tipos de problemas dentro de una estación de montaje aeronáutica. Para determinar qué solución es la mejor, se van a fijar dos funciones objetivos de minimización: la finalización de la última de las actividades de la estación (*makespan*) y el coste total de los trabajadores necesarios, cumpliendo con las restricciones de precedencia, la limitación de recursos y las limitaciones de capacidad de las zonas de trabajo de la estación de montaje.

ÍNDICE

Índice de Ilustraciones

Índice de Figuras

1. Objetivos y alcance del trabajo

Este capítulo abarca la principal motivación de la realización de este trabajo, los problemas de programación de proyectos con limitación de recursos. Este tipo de problemas se ha abordado a lo largo de la historia por un gran número de autores ya que abarca un gran campo en el mundo de la investigación y desarrollo, y es ampliamente aplicado en diferentes tipos de actividades como pueden ser la fabricación y montaje de productos, los proyectos de obra civil o las operaciones de mantenimiento. Se presentan a continuación los principales objetivos que se quieren mostrar a lo largo del documento y se hace una estructuración de lo que consisten los diferentes capítulos del mismo.

1.1 Introducción

Con el paso del tiempo, se han producido varias Revoluciones Industriales. En todas ellas, ha habido descubrimientos y mejoras en las tecnologías que hacían crecer el sector industrial. El objetivo de todas estas mejoras es la satisfacción de las personas obteniendo, por parte de las empresas, el mayor beneficio posible reduciendo costes, minimizando el tiempo de producción y manteniendo la buena calidad del producto.

Hay que centrarse en la Segunda Revolución Industrial a finales del siglo XIX. En ella aparece un nuevo concepto, la **cadena de montaje**. También aparecieron como fuentes de energía el petróleo y la electricidad y con ellos, la aviación.

Uno de los pioneros de la cadena de montaje fue *Henry Ford* y lo puso en práctica para la fabricación de su automóvil, el *Ford T*. *Henry*, perfeccionando las ideas de *Frederick Winslow Taylor* y de *Ramson Eli Olds*. Del primero de ellos, utilizó su teoría de la división de las tareas de trabajo del proceso productivo. Con ello, realizaba un estudio para maximizar la eficiencia de la producción, la organización y reparto de las tareas entre los trabajadores y disminuir los tiempos ociosos de las actividades. Así, los costes de fabricación disminuían y los beneficios serían mayores.

De *Olds* sacó la idea del ensamblaje en las cadenas de montaje, la cual consistía en estaciones de trabajo fijas a lo largo de la línea, piezas estandarizadas e intercambiables y operarios con un trabajo muy específico, sencillo y repetitivo. Esto consistía en producir las piezas por separado siguiendo una línea en serie para después ensamblarlas en la cadena ordenadamente con el objetivo de abaratar los costes disminuyendo la mano de obra y minimizando los tiempos.

Estos modelos de producción tienen su origen en el sector automovilístico pero más tarde, se introdujeron en la fabricación de cualquier producto mejorando cada vez más su eficiencia, gracias a la utilización de nuevos modelos, maquinarias y aspectos teóricos.

La cadena de montaje está formada por diferentes estaciones en las que se realizan una serie de actividades respetando en todo momento algunas restricciones como pueden ser las relaciones de precedencia o la **limitación del número de recursos**, como en este caso. El producto va pasando por las diferentes estaciones y las diferentes piezas que lo componen se van ensamblando hasta obtener el producto final.

Esto es lo que ocurre en el **sector aeronáutico**, donde se aplica en este TFG. El gran volumen de actividades necesarias para el montaje de sus piezas hace que se tengan diferentes estaciones a lo largo de la cadena de producción. Cada una de estas estaciones trabaja con un número de recursos, los cuales son mayoritariamente mano de obra.

En el estudio que se realiza en este trabajo, otra de las características a tener muy en cuenta en las estaciones de montaje aeronáutico es la **limitación de zonas**. Las zonas son espacios de trabajo dentro de cada estación, en los cuales se realizan una serie de actividades concretas, algunas en paralelo, y que por diversos motivos (personal, herramientas o condiciones físicas) se limita la cantidad de recursos que pueden trabajar a la vez en cada zona.

El principal objetivo de las cadenas de producción, y de las estaciones en sí, es conseguir el producto en el **menor tiempo reduciendo lo máximo posible los costes sin perjudicar la calidad del bien final.**

Por esta razón, el estudio de la programación de proyectos, ***PSP***, es muy importante. Además, se le pueden añadir a este tipo de problemas restricciones como la limitación de recursos transformando el problema principal en ***RCPSP*** o ***MRCPSP***, dependiendo de los diferentes modos que existen para realizar una tarea. O también añadir restricciones de zonas como ocurre en el sector aeronáutico. Estas limitaciones en este tipo de problema condicionan la solución óptima de manera severa.

Bien es sabido que en función de las actividades que se realizarían en cada zona y de la limitación de recursos de las mismas, la solución en aspectos económicos, de recursos y temporales puede variar.

Con todo esto, la asignación de recursos a tareas y la organización de las actividades en zonas deben hacerse de modo que se optimicen determinados criterios de eficiencia. De ahí, la gran complejidad de los problemas de este tipo.

Estas son las principales razones del estudio de este tipo de problema.

1.2 Objetivos

El objetivo principal del trabajo que se expone es el desarrollo de nuevos modelos matemáticos de programación lineal, basados en los problemas de programación de proyectos y su resolución para cualquier estación de montaje aeronáutica con distintos modos de ejecución de las tareas y restricciones de recursos y zonas.

De igual modo, a lo largo de dicho trabajo se presentan una serie de objetivos parciales que se detallan a continuación:

- Se analizarán los dos tipos de problemas de programación de proyectos con limitaciones de recursos (*RCPSP y MRCPSP*), analizando las diferencias entre ellos. En el primer caso se realizan las tareas de un único modo; mientras que, en el segundo, las actividades se pueden ejecutar de diferentes modos, teniendo así diferentes tiempos.

- Se hará un repaso por toda la literatura para comentar cómo distintos autores han resuelto los problemas de *PSP* a lo largo de la historia con diferentes métodos de resolución, chequeando la eficiencia de cada uno de ellos.

- Se darán a conocer los modelos matemáticos estándares, tanto del *RCPSP* como del *MRCSP*, para estudiar las diferencias entre ellos y, más tarde resolver diversos ejemplos de cada tipo de problema mediante un software de programación.

- Se abordará este tipo de problema en la industria aeronáutica mostrando las características de la misma, las diferentes restricciones de recursos y zonas a tener en cuenta para la realización del estudio y explicar el porqué de cada una de ellas. De este modo, se adaptará el problema *MRCPSP* a una estación de montaje aeronáutica y se reformulará el modelo matemático a partir del *MRCPSP* estándar.

- Se realizarán una serie de experimentos utilizando el software de programación *Lingo,* para hallar las soluciones de los problemas *RCPSP* y *MRCPSP* habiendo utilizado la librería *PSPLIB* de Kolisch y Sprecher (1997), pudiendo de este modo hacer una comparación de los resultados en ambos tipos de problemas.

- Se plantearán y resolverán una serie de problemas propios como los explicados con anterioridad respecto a la estación de montaje aeronáutico con sus respectivas restricciones y así, validar y encontrar una solución para ellos. En este tipo de problema, se prestará especial atención a la eficiencia computacional de resolución y a las soluciones óptimas halladas.

1.3 Estructura

En este apartado se va a hacer una breve introducción acerca de lo que se va a presentar en cada uno de los diferentes puntos de este trabajo.

En el *Capítulo 2,* se detallan una serie de conceptos sobre qué son los proyectos y la planificación de las tareas que se realizan en ellos y su forma de representación, así como la forma de calcular los tiempos máximos y mínimos de inicio y fin de cada una de las actividades.
También se explican los dos tipos de problemas que se van a ejecutar en este documento, teniendo en cuenta las limitaciones de recursos y los modos de trabajo de las tareas, *RCPSP* y

MRCPSP respectivamente. De igual modo, se revisan los métodos de solución de estos tipos de problemas a lo largo de la historia.

En el *Capítulo 3,* se detallan los modelos matemáticos de los problemas que se van a abordar.

En el Capítulo 4, se relacionan estos problemas con las estaciones de montaje aeronáuticas. Se explica la función y la importancia de las zonas de trabajo en este tipo de líneas de montaje, así como los tipos de recursos (humanos) que trabajan en ellas y su organización. El final del capítulo se centrará en el modelo matemático de los problemas *MRCPSP* enlazado con las restricciones de zona y recursos de las estaciones de montaje aeronáutico.

En el *Capítulo 5* se analizan los resultados de las soluciones obtenidas mediante *Lingo* de los problemas *RCPSP* y *MRCPSP* proporcionados por la librería *Kolisch y Sprecher (1997)*. Durante el capítulo, se elabora una librería propia, la cual posee problemas propios acerca de las estaciones de montaje aeronáuticas y el estudio de sus soluciones correspondientes mediante el mismo programa citado recientemente.

En el *Capítulo 6* se presentan las diferentes conclusiones obtenidas a lo largo de todo el trabajo y las posibles líneas futuras que pudiera haber.

En el *Capítulo 7* se muestran las citas bibliográficas, de las cuales se ha obtenido toda la información expuesta en el presente trabajo.

En el *Capítulo 8* se encuentran un conjunto de documentos adjuntos que son de gran ayuda para la explicación de todo lo comentado durante el TFG.

2. La planificación de tareas en estaciones de montaje

A lo largo de la vida, las personas se enfrentan a multitud de proyectos. Y no solo a proyectos empresariales sino también a proyectos en la vida cotidiana como la realización de un Trabajo Fin de Grado, organizar un evento o simplemente ir un día de tiendas. Normalmente, cuando se habla de proyecto uno piensa en grandes construcciones muy complejas y de difícil gestión, pero no es así. Los ejemplos nombrados anteriormente lo demuestran.

Entonces, ¿qué es un proyecto? Según el PMI, un **proyecto** es *"un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. Tiene un principio y un final definidos. Se considera finalizado cuando se han llevado a cabo los objetivos, cuando no es posible que se cumplan los objetivos o cuando no existe la necesidad que inició el proyecto".*

De una forma más general, se puede decir que un proyecto es un conjunto de tareas interdependientes, con una fecha de inicio y una fecha de finalización, coordinadas con el fin de cumplir una serie de objetivos basados en el coste, tiempo y calidad.

El **coste** hace referencia al presupuesto que se tiene; el **tiempo** es la fecha límite para que el proyecto esté listo y la **calidad** es la excelencia con la que se hacen las tareas. Estos factores forman el objetivo del proyecto. Estos objetivos se regulan gracias al *Project Management* que es una disciplina que engloba la organización, el planeamiento, la motivación y el control de los recursos con el fin de cumplir los objetivos teniendo en cuenta una serie de limitaciones.

Entonces, ¿qué se puede considerar un proyecto y qué no? Un proyecto es aquel plan que se ha pensado con detenimiento y se sabe cuándo empieza y cuándo acaba. El fin que se busca con ello, es conseguir una necesidad ya sea material o no siempre diferente a todo lo ya existente. Tiene un carácter no repetitivo, es decir, nunca habrá dos proyectos exactamente iguales. Todo aquello que cumpla con dichas características será un proyecto.

Es muy importante también diferenciar proyecto de operación. Una operación no es un proyecto, ya que esta tiene un tiempo de inicio pero no de final. Además, en una operación siempre se repite el mismo proceso con un mismo fin. Todo lo contrario a un proyecto.

Un ejemplo de proyecto podría ser el diseño de un nuevo avión. Se sabe cuándo empezará y en qué momento se termina de producir. Este será el fin del proyecto y el resultado, el avión, es único ya que no hay dos aviones iguales. El diseño del aeroplano, está formado por numerosas tareas a realizar, las cuales llevan un orden y organización y todas están relacionadas unas con otras para conseguir dicho objetivo.

Para diferenciarlo de una operación, se puede decir que una operación es la fabricación de los asientos del avión, los cuales, todos serán iguales y la tarea a realizar para su producción es siempre la misma.

Asimismo, un proyecto se encuentra en la vida cotidiana de las personas como ya se citó anteriormente. Nombrando el ejemplo anterior de ir un día de tiendas, hay un plan que es ir de tiendas. Se sabe a qué hora se saldrá de casa y que, el mismo, terminará una vez llegado a casa. Las actividades a realizar para llegar a la tienda son ordenadas y organizadas (montarse en el coche o ir en bus, echar gasolina, el recorrido que se va a elegir, a qué tienda ir primero, cómo pagar…). El resultado es la obtención de nuevos artículos. Y en ningún otro momento, el resultado de un proyecto similar será el mismo.

Tan importante es el objetivo final del proyecto como cuál es el recorrido que se hace hasta llegar a él. Como ya se apuntó con anterioridad, hay ciertos factores muy a tener en cuenta a la hora de realizar el proyecto: coste, tiempo y calidad. Para conseguir el máximo partido de ellos es necesaria una muy buena organización en todos los aspectos del mismo.

Volviendo al ejemplo del avión, el cual se desarrollará más profundamente en apartados siguientes, la ordenación y organización de las tareas para su producción se hace por estaciones de montaje. Es muy importante la planificación que se hace de las distintas actividades que tiene cada una de ellas, es decir, en qué momento empieza y acaba cada actividad, el numero necesarios de operarios/máquinas para la realización de la misma, los posibles riesgos implicados que pudiese haber (pérdidas de tiempo, paradas de emergencia, mantenimiento de una máquina…), el orden de las tareas, las distintas habilidades de cada operario o la velocidad de trabajar de cada uno de ellos. Es una amplia gama de factores a tener en cuenta que condicionan el coste, tiempo y calidad del proyecto.

Este Trabajo Fin de Grado se centra en la planificación de una cualquiera de estas estaciones de montaje de un avión, teniendo en cuenta los diversos condicionantes para analizar las diferentes situaciones que pudiese haber y llegar a la mejor decisión que favorezca el resultado final del proyecto.

2.1 La planificación de tareas (PSP)

Los proyectos tienen un ciclo de vida formado por cuatro fases: planificación del proyecto, programación del proyecto, control del proyecto y terminación del proyecto.

En la que se centrará este trabajo es en la segunda fase, la programación del proyecto. En ella, se estudia la secuenciación de las actividades y, por lo tanto, los tiempos de inicio y finalización de las mismas. De este modo, se sabe en qué momento hay que empezar cada actividad y en qué momento terminará el proyecto.

Para ello, se tienen en cuenta las limitaciones de los recursos disponibles para la realización del proyecto y las relaciones de precedencia de las actividades que lo forman (una actividad no puede realizarse hasta que sus predecesoras se hayan ejecutado). Esto es de gran utilidad tanto para la planificación interna como externa del proyecto. A esta problemática relacionada con la organización y la ordenación de las actividades se le llamará *Project Scheduling Problem (***PSP***)*.

Todo esto, en primer lugar, se planifica y se programa para que, más tarde, se ejecute dentro de una cadena de producción o línea de montaje. Una cadena de producción es un conjunto de operaciones planificadas que, utilizando una serie de recursos materiales y humanos, obtengan un producto final. Este proceso productivo está dividido en diferentes estaciones de trabajo. En un estudio previo se calcula el número óptimo de estaciones que forma la línea, así como de las actividades asignadas a cada estación. En cada una de las estaciones se aplica un procedimiento diferente al producto hasta conseguir dicho producto final en la última estación. El producto pasa por cada estación siguiendo un orden de precedencia y cumpliendo las posibles limitaciones de recursos que puede haber. A su vez, cada estación está formada de una o más actividades, las cuales siguen también un orden de precedencia y limitación de recursos. Para pasar de una estación a la siguiente se tiene que ejecutar cada una de las actividades de la estación anterior.

Para calcular el instante de inicio y fin de cada actividad de un proyecto, así como el del proyecto en su conjunto, se utilizarán los diagramas **PERT** (*Project Evaluation and Review Technique*). Estos diagramas se crearon en 1957 en la Oficina de Proyectos Espaciales de la Armada de los Estados Unidos para el proyecto *Polaris*. Consisten en la representación visual, mediante nodos y arcos, de las actividades que conforman el proyecto y las relaciones de precedencia entre ellas. Con los diagramas PERT es sencillo calcular el tiempo de ejecución de cada actividad.

A la hora de la representación PERT hay dos formas equivalentes (*Figura 1*). La primera de ella es la conocida como *AEN* (Actividades En Nodos), donde las actividades son los nodos y los arcos son las relaciones de precedencia. La segunda es la *AEA* (Actividades en Arcos), donde tanto las actividades como las relaciones de precedencia se encuentran en los arcos, mientras que los nodos representan instantes de inicio y fin de las mismas. Para las dos representaciones existen un nodo de inicio el cual se conecta con las actividades sin precedencias, y un nodo final al que se conectan todas las actividades sin sucesoras. Los diagramas no pueden formar ciclos dirigidos, ni tener arcos redundantes y los eventos tienen que estar enumerados desde el inicio hasta el fin.

Actividad	Predecesor
A	-
B	-
C	B
D	A,C
E	C
F	C
G	D,E,F

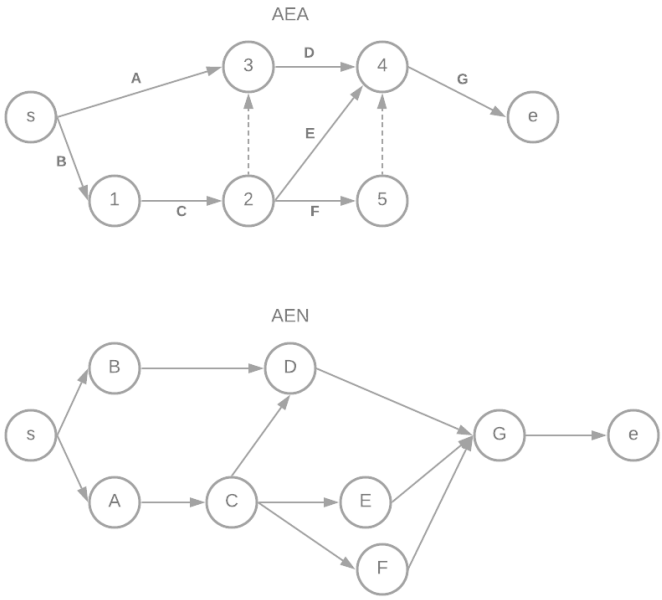


Ilustración 1: Ejemplos de Diagramas PERT en sus dos formas, AEA y AEN

Junto con el grafo PERT se utiliza también el *Critical Path Method (***CPM***).* Este método utiliza la representación gráfica del PERT para calcular los tiempos de inicio y fin de cada actividad y con ello, el tiempo de terminación del proyecto. De esta forma, se puede hallar la ruta crítica del proyecto, es decir, la secuencia de tareas que deben cumplirse a tiempo para obtener el éxito del proyecto sin retraso. Esto quiere decir que aquellas tareas que están en la ruta crítica (actividades críticas) no se pueden ejecutar con retraso. Gracias al CPM, también se pueden calcular los tiempos de holgura que tiene cada actividad no crítica (lo máximo que se puede atrasar y seguir cumpliendo las restricciones de tiempo y precedencias del proyecto).

Para hallar una solución del método CPM se tienen que determinar tres elementos:

- La **hora de inicio más temprana** para todos los trabajos.
- **La hora de finalización más tardía** de todos los trabajos.
- Qué trabajos no tienen holgura alguna (**actividades críticas**).

Para ello, se utilizan las variables explicadas a continuación,

- $\boldsymbol{S_j'}$: $tiempo\ más\ temprano\ de\ inicio.$
- $\boldsymbol{S_j''}$: $tiempo\ más\ tardío\ de\ inicio.$
- $\boldsymbol{C_j'}$: $tiempo\ más\ temprano\ de\ finalización.$
- $\boldsymbol{C_j''}$: $tiempo\ más\ tardío\ de\ finalización.$
- $\boldsymbol{slack_j}$: $holgura\ de\ la\ actividad \rightarrow\ C_j'' - p_j - S_j'$ siendo $p_j$el tiempo de proceso de la actividad.

Su resolución consta de dos pasos: procedimiento hacia adelante y procedimiento hacia atrás.

El *procedimiento hacia adelante* lo forman tres pasos:

1. Para cada actividad sin precedencia,
   $S_j' = 0$
   $C_j' = p_j$

2. Calcular para cada tarea j siendo k su precedente,
   $S_j' = \max_{k \rightarrow j} C_k'$
   $C_j' = S_j' + p_j$

3. Tiempo máximo de finalización,
   $C_{max} = \max_j C_j'$

En este primer proceso, aquellas tareas que no tengan precedencia alguna, lo antes que pueden empezar es en el instante 0 (fecha de inicio del proyecto); mientras que el menor tiempo de finalización es el de sus tiempos de procesos. A partir de este momento, todas las demás actividades tendrán precedencia, por lo tanto, el momento más temprano para empezar a ejecutarlas es cuando acabe la actividad predecesora con mayor tiempo más temprano de finalización.

El tiempo más temprano que pueden acabar estas actividades es su tiempo de inicio más temprano más su tiempo de proceso. Este paso se repite hasta llegar a la última actividad, y que no tiene sucesoras. El tiempo de finalización máximo del proyecto sería el mayor tiempo entre los tiempos más temprano de finalización de sus predecesoras.

El *procedimiento hacia atrás* es inverso al anterior, se realiza después del anterior y también está constituido por tres pasos:

1. Para cada trabajo sin sucesoras,
   $C_j'' = C_{max}$
   $S_j'' = C_{max} - p_j$

2. Calcular para cada tarea j con predecesoras,
   $C_j'' = \min_{j \to all\ k} S_k''$
   $S_j'' = C_j'' - p_j$

3. Verificar que,
   $0 = \min_j S_j''$

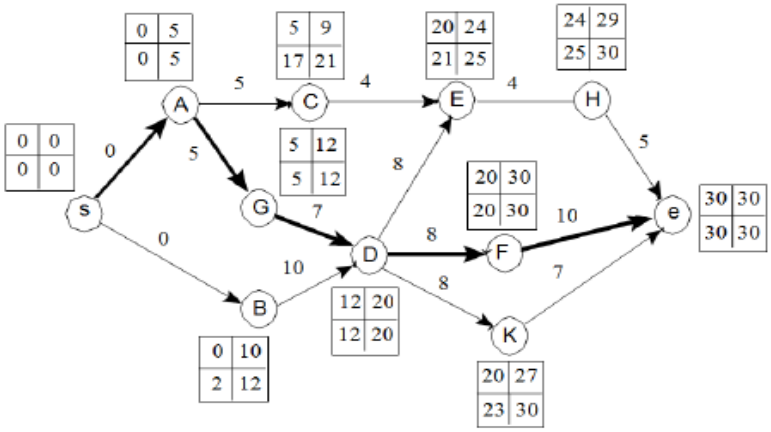
En este segundo procedimiento, se tiene en cuenta el final del primer procedimiento. En este caso, se empieza por la tarea sin sucesoras donde su tiempo más tarde de finalización es el tiempo máximo del procedimiento hacia delante. Los tiempos de inicio más tardíos son el tiempo máximo de finalización menos los tiempos de ejecución de las tareas. Una vez llegados a este punto, los tiempos más tardíos de finalización de las tareas son el mínimo tiempo de los inicios más tardíos de las actividades de todas sus sucesoras. Y claramente, como comprobación del proceso y seguridad de que es correcto, el tiempo más tardío de inicio del proyecto tiene que ser el instante 0.

Hechos estos dos procedimientos, hay que determinar qué tareas son críticas y conforman el camino crítico. Esto se consigue mediante el cálculo de las holguras. Es decir, se calcula las holguras de cada tarea con la expresión citada anteriormente y aquellas que tengan holgura nula serán las críticas y formarán dicho camino.

Para el cálculo de las holguras totales, se puede utilizar la fórmula detallada anteriormente o cualquiera de las dos expresiones que se citan a continuación: $slack_j = C_j' - S_j'$ ó $slack_j = C_j'' - S_j''$

Como ya se dijo, la representación del PERT y/o CPM puede ser en AEN o AEA. Esto no influye a la hora de determinar el camino crítico ya que en ambos casos se realiza del mismo modo.

Actividad	Precedentes	Duración
A	-	5
B	-	10
C	A	4
D	B y G	8
E	C y D	4
F	D	10
G	A	7
H	E	5
K	D	7



Actividad	s	A	B	C	D	E	F	G	H	K	e
Holgura	0	0	2	12	0	1	0	0	1	3	0

Ilustración 2: Ejemplo de Diagrama PERT con CPM

En la *Ilustración 2,* se puede observar cómo se aplican los dos procedimientos para calcular los diferentes tiempos de inicio y finalización usando la representación AEN, así como cuál es el camino crítico del ejemplo. También se calculan las holguras de las actividades para determinar las actividades críticas que forman el camino citado recientemente.

En cada actividad viene indicado en la parte superior izquierda el tiempo de inicio más temprano (S'_j), en la parte derecha superior el tiempo de finalización más temprano (C'_j). En la parte inferior izquierda el tiempo de inicio más tardío (S''_j), mientras que en la parte derecha inferior el tiempo de finalización más tardío (C''_j).

En los arcos se indica el tiempo de proceso de cada actividad. Las actividades *"s"* y *"e"* son las actividades de inicio y fin respectivamente.

Aquellos arcos marcados de color negro son los que formas la ruta crítica y lo que determinan las actividades críticas.

La representación del camino crítico se puede visualizar en un **Diagrama de Gantt**. Este diagrama es un cronograma detallado de las actividades que se tienen que realizar en cada estación. En él se puede ver en qué momento empieza y acaba cada tarea a realizar. En el eje vertical se encuentran las actividades, mientras que en el eje horizontal está la escala de tiempo.

Con el diagrama de Gantt se puede observar que las actividades críticas no pueden cambiar de lugar, mientras que las no críticas, pueden desplazarse en el tiempo dependiendo de su holgura.

Otra representación del Diagrama de Gantt es la que está orientada a las máquinas. La diferencia con el caso anterior es que en el eje vertical, en vez de estar representados los trabajos, se representan las máquinas en las que se realizan las actividades.

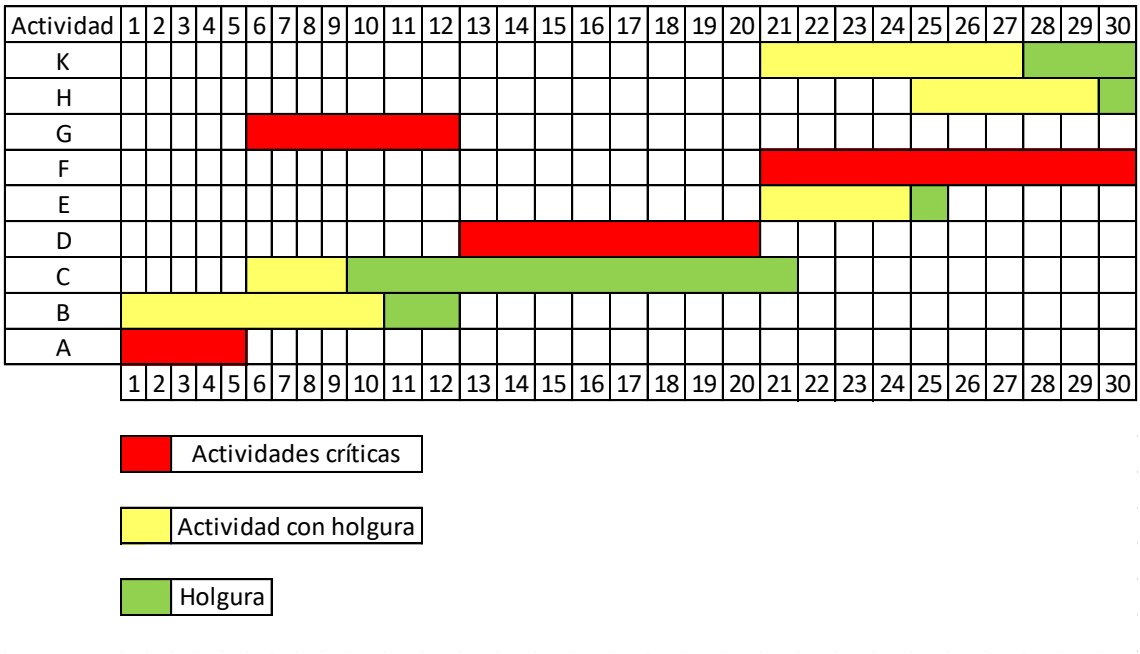


Ilustración 3: Ejemplo de Diagrama de Gantt

En la *Ilustración 3* se muestra un ejemplo del Diagrama de Gantt relacionado con el ejemplo de la *Ilustración 2,* en el cual se puede ver aquellas actividades que son críticas por estar en el camino crítico y su holgura es nula, y aquellas actividades que tienen holgura.

2.2 El problema con limitaciones de recursos (RCPSP)

En los problemas de planificación y programación de las tareas (PSP) cuando existe recursos limitados, el objetivo es el mismo que el anterior pero hay que tener en cuenta que las tareas tienen recursos asignados y éstos no pueden superar la capacidad disponible en cada instante de tiempo. Es decir, en los PSP no solo hay que calcular una ordenación de las diferentes actividades en el tiempo, sino que también es necesario considerar que existe un número limitado de recursos. Por este motivo, surge el problema de **RCPSP** *(Resource Constrained Project Scheduling Problem)*.

Este tipo de problemas, en su versión más estudiada, consiste en minimizar el tiempo de ejecución del proyecto (*makespan*) teniendo en cuenta que las actividades que lo forman siguen una relación de precedencia Fin-Inicio (no puede empezar una tarea hasta que sus predecesoras hayan terminado), y que los trabajos son realizados por un número de recursos que están limitados.

Los recursos existentes y necesarios para la realización de las tareas pueden ser de dos tipos: renovables y no renovables. El primero de ellos son las personas, máquinas, herramientas o el espacio y el tiempo. El segundo se centra en el ámbito económico, energético o en materias primas. La diferencia entre ellos es que los recursos renovables están disponibles en cada periodo, es decir, se renuevan en cada espacio de tiempo y nunca se agotarían; mientras que los recursos no renovables tienen un límite durante el proyecto en su totalidad y a medida que se van realizando las diferentes actividades se van gastando y no se vuelven a recuperar. Algunos de los recursos están limitados tanto en su totalidad durante el proyecto como en los espacios de tiempo en los que se utilizan. Un tercer tipo de recurso sería los recursos (no) renovables, cuya disponibilidad es limitada durante todo el proyecto, pero en ciertos momentos específicos del mismo se renuevan.

Una forma de representar los recursos utilizados en cada momento es mediante el **Diagrama de Carga**. Este diagrama se genera para cada tipo de recurso (especialidad) y es similar al Diagrama de Gantt. Está conformado por dos ejes: en el eje horizontal se muestran los instantes de tiempo, y en el eje vertical se muestra el número de recursos utilizados en cada instante de tiempo, marcando el límite de recursos disponibles. Así, sabiendo también la asignación de las tareas en el tiempo y el número de trabajadores que realiza cada tarea, se calcula el número de recursos necesarios en cada instante de tiempo y si, en algún momento, se sobrepasa el límite, se puede cambiar el instante de inicio de las tareas no críticas gracias a las holguras para que así también se cumplan las restricciones de recursos.

Uno de los principios básicos de los problemas RCPSP consiste en que cada actividad sólo se puede realizar de un único modo, siguiendo unas precedencias con otras actividades, teniendo una duración fija predeterminada, así como una cantidad de recursos asignados a la actividad para su ejecución.

Para comparar algoritmos y métodos de resolución de problemas PSP, se utiliza la librería estándar *PSPLIB*, y que consiste en una gran cantidad de instancias para este tipo de problema RCPSP. Se conocen las soluciones óptimas de estos problemas con el objetivo de minimizar el *makespan*. En este TFG se utilizará esta librería, pero sólo para aquellos problemas que tienen un máximo de 30 actividades. Estos, pueden ser resueltos por métodos exactos, por lo tanto, se puede llegar a conocer sus soluciones óptimas mediante programación lineal.

Para problemas de mayor tamaño, llegar a su solución óptima mediante estos métodos exactos es inviable debido a la gran cantidad de variables binarias y restricciones que existirían, por lo cual habría que utilizar métodos aproximados. Estos métodos generan soluciones admisibles lo más cercana posible del óptimo (*Scholl & Becker, 2006).*

Los RCPSP son un tipo de problema *NP-Hard*, por lo tanto es imposible resolver todas las instancias de forma óptima. Para muchos de ellos habrá que aplicar procedimientos aproximados en vez de métodos exactos para su resolución.

Los estudios más recientes sobre métodos exactos son los de *Kolisch y Padman (2001)* y *Demeulemeester y Herroelen (2002).* De procedimientos aproximados habría que destacar los artículos *de Kolisch y Hartmann (1999), (2000)* y *(2006).*

2.3 El problema con varios modos de trabajo (MRCPSP)

Se han introducido en el punto anterior los problemas con un único modo de ejecución de las actividades. Pero en muchos casos, estas actividades se pueden realizar de más de una forma. Cada una de estas formas tiene un tiempo de proceso y número de recursos necesarios distintos.

Ante esta situación, surge un tipo de problemas nuevo: *Multi-Mode Resource Constrained Project Scheduling Problem (***MRCPSP***).* La función objetivo más estudiada es la misma que para los problemas RCPSP, minimizar el *makespan*, sin sobrepasar el número límite de recursos en cada instante de tiempo. Una diferencia respecto a los problemas de un único modo es que, en los MRCPSP, además de calcular los instantes de inicio de cada actividad, hay que elegir el modo de ejecución de cada actividad, llevando implícito el tiempo y el número de trabajadores necesarios.

Los MRCPSP también son problemas del tipo *NP-Hard*, y por tanto, los métodos exactos son incapaces de alcanzar el óptimo en problemas de tamaño medio-grande. Muchos de los algoritmos no encuentran una solución óptima a partir de 20 actividades y tres modos de ejecutar cada actividad. Los procedimientos aproximados sí son capaces de hallar soluciones factibles y cercanas al óptimo. La dificultad de estos problemas es enorme. Es por ello por lo que se incluye en muchos algoritmos de resolución el pre-proceso formulado por *Sprecher et al. (1997),* en el cual se descartan los modos no ejecutables e ineficientes. En los procedimientos aproximados, la asignación de los modos para una solución básica inicial del algoritmo se hace de una forma totalmente aleatoria.

Para este tipo de problemas existe, de igual modo que en los *RCPSP*, la librería *PSPLIB*. En este caso tiene una variedad de problemas de 10, 12, 14, 16, 18, 20 y 30 tareas con dos recursos renovables y dos recursos no renovables (en este trabajo sólo se contarán con las librerías de recursos renovables). Se conocen las soluciones óptimas de los problemas menores de 30 actividades con el objetivo de minimizar el *makespan*.

2.4 Métodos de resolución para el RCPSP y MRCPSP

Los problemas de Programación de Proyectos con Recursos Limitados han sido estudiados a lo largo de los años por numerosos autores dejando una amplia literatura de la explicación y resolución de estos tipos de problemas. Como ya se citó anteriormente, los problemas RCPSP y MRCPSP, se pueden solucionar mediante Métodos Exactos y Métodos Aproximados.

Los primeros tienen como objetivo conseguir la solución óptima del problema. Pero esto requiere un enorme esfuerzo computacional (lo que llevaría una gran cantidad de tiempo) al tratarse de problemas *NP-Hard*, por lo que los problemas de un tamaño mediano y grande son muy difíciles de resolver.

Los segundos se han desarrollado mucho durante las últimas décadas. Estos métodos consisten en intentar llegar a una solución lo más cercana posible de la solución óptima. Este

tipo de procesamiento requiere un esfuerzo menor y las soluciones se obtienen en un tiempo razonable.

Los estudios más recientes sobre métodos exactos son los de *Kolisch y Padman (2001)* y *Demeulemeester y Herroelen (2002)*. De procedimientos aproximados habría que destacar los artículos de *Kolisch y Hartmann (1999), (2000)* y *(2006).*

2.4.1 Métodos exactos de resolución para el RCPSP y MRCPSP

2.4.1.1 Branch & Bound

También llamado *Ramificación y Poda,* el Branch & Bound es el mejor método exacto para la resolución de este tipo de problemas. Este método consiste en ir explorando todas las diferentes soluciones que se encuentran en el espacio muestral, descartando aquellas que ni son ni conducen a una solución óptima.

La eficiencia del método depende de la cota inferior que se calcule por alguno de los procedimientos que existen. Para ello se relajan algunas restricciones de recursos y se resuelve el problema óptimamente. A mayor relajación, peor cota inferior. Esta cota se irá comparando con otras para ver cuál es mejor hasta que se obtenga la óptima. Al ser un problema de minimización, mientras menor sea la cota inferior la solución será mejor. Esta cota se actualizará si la solución hallada es menor que ella y cumple todas las restricciones impuestas. Interesa tener una cota inferior inicial lo más próxima al óptimo, ya que esto ahorraría tiempo de trabajo al no tener que analizar posibles soluciones y ramas de más.

Además, existen unas reglas de dominancia que permiten eliminar aquellas ramas del árbol que no conducen a la solución óptima. Estas reglas utilizan aspectos matemáticos para saber, a priori, si la solución puede hallarse o no en esa rama para continuar por ella y conseguir una solución completa o eliminarla para acotar la búsqueda del óptimo.

Para este tipo de problema, *Branch & Bound* consiste en que en cada etapa se va formando una secuencia parcial factible con las actividades ya programadas y que se le van asignando. Cada vértice del árbol que se forma es una secuencia parcial. Estas secuencias se irán extendiendo asignándole más actividades siempre y cuando sea factible. Puede llegar un momento en el que estas ramas del árbol se puedan cortar por las reglas de dominancia al no tener una solución óptima o porque se conoce que la solución óptima no se encuentra en esa rama.

Para la fase de ramificación del árbol, es decir, cómo se va dividiendo el árbol hay una serie de métodos. Algunos de ellos son:

- **Árbol de precedencias**: consiste en programar, lo antes posible, cada una de las actividades que todavía no se han programado, pero que sus actividades predecesoras sí, de cada nodo ya que cada uno de ellos tiene una secuencia parcial diferente. Con cada actividad nueva que se pueda programar se formará una nueva rama. El método

concluirá cuando la actividad ficticia del final se pueda programar. De esta forma, cada nodo tendrá tantas ramas como actividades que se puedan programar en él.
Este método está basado en el trabajo de *Talbot (1982)*, fue representado por *Patterson et al. (1989)* y mejorado por *Sprecher (1994).*

- **Alternativas de retraso**: cada nodo tiene un tiempo de secuenciación (el menor de los instantes de finalización de las actividades en curso) en el que pueden comenzar las actividades. Las actividades elegibles son aquellas que pueden iniciarse en ese momento. Si hubiera problemas con los recursos al programar dichas actividades, se determinaría aquellas actividades que están procesándose y cuyo retraso solucionaría el problema de los recursos. A este conjunto de actividades se les denomina alternativa de retraso. Este método fue propuesto por *Christofides et al. (1987)* y, para los MRCPSP*, Sprecher et al. (1997)* habiéndolo analizado antes *Demeulemeester y Herroelen (1818).*

- **Alternativas de extensión**: se denomina alternativa de extensión al conjunto de actividades elegibles que se podría ejecutar en el instante actual son sobrepasar los límites de recursos. cada nodo tiene tantas ramas como alternativas de extensión. La poda se realiza cortando en primer lugar el mejor. Para expandir la búsqueda se utilizan varias reglas de decisión y de dominancia y poda. Este procedimiento fue propuesto por *Stinson (1978)*. *Hartmann & Drexl (1998)* extienden esta técnica para el MRCPSP al combinar alternativas de modo (fijar el modo de aquellas actividades que todavía no se les ha asignado) y alternativas de extensión.

A parte de estos métodos de ramificación, también existen otros como el de conjuntos prohibidos mínimos (*Igelmund & Radermarcher (1983)* o el de relaciones de cada par de actividad *i-j* de *Brucker et al. (1998).*

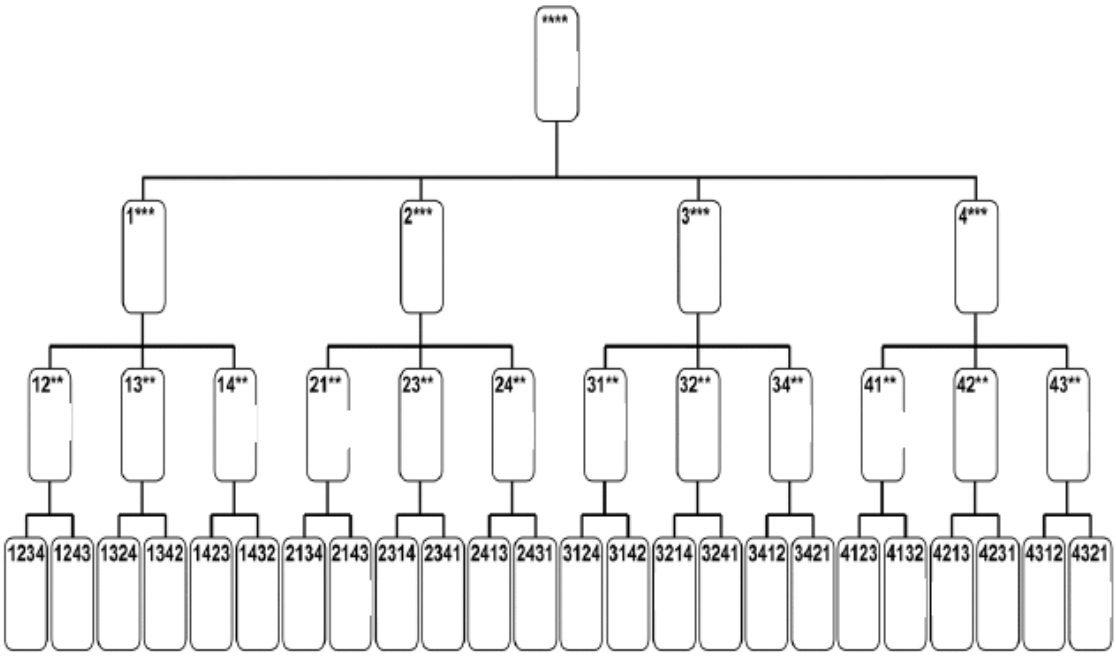


Ilustración 4: Ejemplo de Branch & Bound con ramificación y poda con árbol de precedencias

2.4.1.2 Programación Dinámica (Dynamic Programming)

La *Programación Dinámica* es una metodología adecuada e ideal para aquellos problemas en los que haya que tomar una serie de decisiones de manera secuencial, es decir, una decisión detrás de otra, por ejemplo, en diferentes espacios de tiempo. Esta metodología consiste en encontrar una solución óptima a partir de soluciones ya obtenidas en etapas anteriores mediante un procedimiento iterativo.

La Programación Dinámica se puede usar si el problema a cuestión tiene una serie de características en concreto:

- El problema se puede dividir en una serie de decisiones a tomar siguiendo un orden. Estas decisiones se toman en varias etapas del problema.
- Cada etapa tiene un número finito de posibles estados.
- Cada decisión tomada en una etapa con un estado determinado, lleva a un estado en concreto a la etapa posterior.
- La mejor decisión que se pueda tomar en una etapa es independiente de las posibles decisiones tomadas en las etapas anteriores.
- Pasar de un estado a otro a través de las etapas, incurre en un coste definido previamente. Esta función de costes debe de ser recursiva.

Esta metodología tiene dos enfoques: hacia delante y hacia tras. Este último consiste en empezar desde la última etapa (el objetivo) y tomar la mejor decisión que se pueda en ella. A continuación, se pasa a la penúltima etapa y, nuevamente, se vuelve a tomar la mejor decisión posible en dicha etapa. Hay que tener en cuenta los costes asociados a estas dos etapas. Este procedimiento se repite hasta llegar a la primera.

La Programación Dinámica es difícil de aplicar de forma práctica en los problemas de RCPSP y MRCPSP y, quizás, solamente se llegaría a una solución factible pero no óptima. En primer lugar porque en muchas ocasiones no se cumplirían todas las características necesarias para aplicar esta metodología. Y en segundo lugar porque tomar decisiones es algo muy complejo y que puede variar durante la ejecución del algoritmo.

2.4.1.3 Programación Lineal Entera Mixta (Mixed Integer Porgramming)

La *Programación Lineal* es una herramienta muy importante dentro de la investigación de operaciones, y más después de que *George Dantzig* desarrollase el método simplex para la resolución de problemas lineales. El fundamento básico de esto es que tanto las restricciones del problema como la función o funciones objetivos del mismo tienen que ser combinación lineal de las variables de decisión.

Como bien es sabido, la función objetivo es la expresión que se quiere minimizar o maximizar. Mientras que las restricciones son aquellas funciones que limitan el espacio en el cual se encuentra la solución óptima del problema. Dicha solución se alcanzará con unos valores determinados de las variables de decisión del problema cumplirán todas las restricciones expuestas y obtendrán el mejor valor posible de la función objetivo.

En Programación Lineal si se define un polígono lineal cerrado no vacío como el espacio de búsqueda de la solución óptima y esta solución existe, esta estará en uno de los vértices del polígono.

Dentro del conjunto de problemas de este tipo, las variables que se pueden encontrar pueden ser continuas, enteras, binarias. De aquí nace la Programación Lineal Entera Mixta. La forma de resolver los problemas con variables enteras y/o binarias es mucho más complicada pero existen algoritmos que lo hacen.

Los problemas RCPSP y MRCPSP se presentan como problemas de Programación Lineal Entera Mixta. En este TFG se verán planteamientos estándares de estos problemas mediante modelos de programación lineal mixta-entera.

2.4.1.4 Divide y Vencerás (Divide and Conquer)

En esta metodología, el problema, a priori, difícil de resolver se puede dividir en problemas más pequeños y con resoluciones más sencillas. Una vez resuelto cada uno de los sub-problemas, la idea sería de juntar las soluciones y así tener una única solución óptima. Esta es la idea principal de *Divide and Conquer*.

Esto solo será eficiente si el esfuerzo y tiempo que se requiere para fraccionar el problema, así como determinar las soluciones de cada sub-problema y unirlas en una sola es menor o igual que lo que se tardaría y el esfuerzo que haría falta si se resolviese el problema como uno solo. Pero la solución obtenida de la unión de las soluciones parciales no tiene porqué ser óptima pero, al menos, sí será admisible.

En los problemas RCPSP y MRCPSP, a la hora de la partición de los mismos, esta se puede hacer de diversas formas. Una de ellas puede ser dividir el problema en intervalo de tiempo donde cada uno es un problema de secuenciación de actividades no programadas. En este caso, la segmentación será un proceso iterativas hasta alcanzar un sub-problema lo suficientemente sencillo para poderlo resolver. Una vez halladas todas las soluciones, el algoritmo irá conformando la solución final obtenida a partir de las pequeñas soluciones.

Como se ha explicado con anterioridad, tanto los problemas de RCPSP como de MCRPSP de pequeño tamaño pueden ser resueltos por métodos exactos. Al contrario, en los problemas de mediano y gran tamaño no pueden ser resueltos de esta forma. En ambos casos es necesario consumir una gran cantidad de tiempo. Ante ello, se encuentran los métodos aproximados que, bien es cierto, no llegan a las soluciones óptimas pero sí se aproximas mucho a ellas en un tiempo razonable manteniendo la calidad de la soluciones.

2.4.2 Métodos aproximados de resolución para el RCPSP y MRCPSP

2.4.2.1 Métodos de redistribución de recursos

Estos *métodos de redistribución de recursos* se basan en utilizar como datos de entrada la solución del método de camino crítico PERT/CPM considerando los tiempos más tempranos de inicio de cada actividad. Dicha solución puede ser admisible o no respecto a las limitaciones de recursos.

Si la solución inicial es admisible, los métodos de redistribución que se aplican únicamente considera las holguras positivas, es decir, se mueven las actividades no críticas hacia adelante hasta los tiempos más tardíos de inicio como máximo, con la idea de que la cantidad de recursos usados sea lo más uniforme posible sin modificar el *makespan* inicial. Como medida de la bondad de la solución se puede usar la desviación absoluta media. Estos problemas entran en la categoría de "*Resource Smoothing*".

Si la solución inicial es inadmisible en algún instante de tiempo, los métodos de redistribución que se aplican pueden considerar tanto las actividades no críticas como las actividades críticas para alcanzar soluciones admisibles en toda la duración del proyecto (*makespan*). Por ello, puede ocurrir que se modifique el *makespan* con una duración del proyecto mayo que la inicial. Estos problemas entran en la categoría de "*Resource Levelling*".

Si nos centramos en este último caso, para saber qué tarea desplazar cuando hay inadmisibilidad de recursos en algún instante de tiempo, existen métodos básicos como el de *West-Levy (1977).* Este método se va al primer instante *t* de inadmisibilidad en recursos, y se elige la actividad *j* con mayor tiempo más tardío de finalización (C''_j) entre la lista de tareas que empezarían en ese instante *t*. En caso de empate, se elige entre las actividades con igual C''_j aquélla con menor número de recursos, y si vuelve a haber varias con la misma cantidad, entonces se elige la de menor duración. Dicha actividad se retrasa una unidad de tiempo y se vuelve a empezar el método.

Este método forzará a que, en algunos casos, el *makespan* aumente para alcanzar una solución admisible en cantidad de recursos disponibles frente a los necesarios.

Ejemplo de aplicación del método de West-Levy: sea un proyecto con 6 actividades donde la duración, las actividades predecesoras y el número de recursos necesarios de un tipo se muestran en la tabla adjunta.

Actividad	Predecesoras	Duración	Recursos necesarios
1	-	2	3
2	-	3	1
3	-	1	2
4	1,2	4	2
5	2,3	2	3
6	4	1	3

En la *Ilustración 5* se puede observar el Diagrama de Carga de Recursos obtenido como resultado del método CPM. En la siguiente tabla se muestran los valores obtenidos del tiempo más temprano de inicio (S') y del tiempo más tardío de finalización (C'') para cada tarea.

Actividad	S'	C''
1	0	3
2	0	3
3	0	6
4	3	7
5	3	8
6	7	8

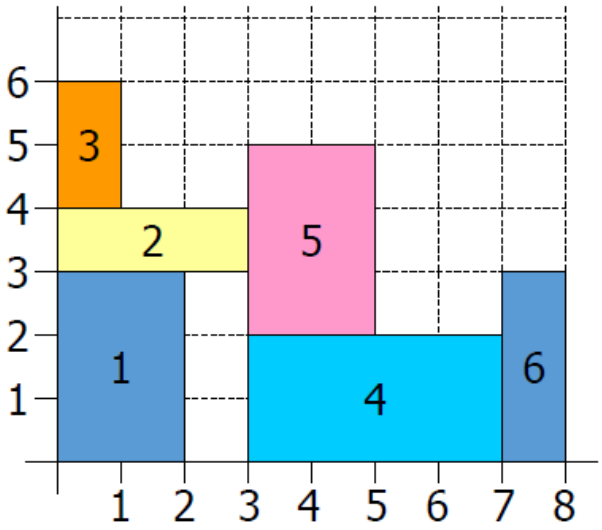


Ilustración 5: Ejemplo de Diagrama de Carga de Recursos según el CPM determinado

Fuente: Eguia Salinas, Ignacio "Apuntes de la asignatura Producción Aeroespacial"

Si el número máximo de trabajadores disponibles en cualquier momento del proyecto es de 5, entonces, se incumplen las restricciones de recursos ya que en el primer intervalo de tiempo son necesarios 6 recursos. No sería una solución factible. Ante esta inadmisibilidad, aplicamos el método de West-Levy eligiendo entre las actividades 1, 2 y 3, aquélla con mayor C'', que en este caso es la 3 y se desplaza una unidad de tiempo. Con ello se resuelve la inadmisibilidad en el primer intervalo pero se pasa al segundo. Se vuelve a aplicar el método en ese intervalo y la única actividad que se iniciaría en ese momento es la 3 que es la que se vuelve a desplazar una unidad de tiempo. Con este desplazamiento la solución se hace admisible para 5 recursos disponibles. El resultado se puede ver en la *Ilustración 6*. Aquí el número máximo de recursos sigue siendo 5, pero en este caso la actividad 3 ha sido desplazada al instante de tiempo 3, y gracias a su holgura no aumenta el *makespan*. Con ello, se consigue que se cumplan las restricciones de recursos ya que en ningún momento se sobrepasa el límite de recursos y el *makespan* no varía.

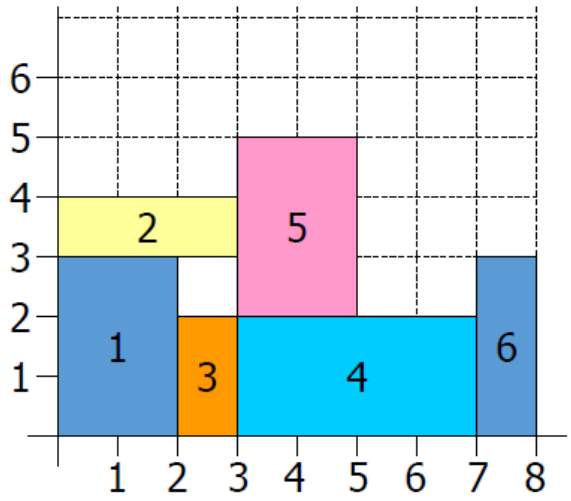


Ilustración 6: Ejemplo de Diagrama de Carga de Recursos con West-Levy para 5 recursos

Fuente: Eguia Salinas, Ignacio "Apuntes de la asignatura Producción Aeroespacial"

Finalmente y en caso de que el número máximo de recursos fuera 4, se observa una inadmisibilidad en el cuarto intervalo. Se aplica el método de West-Levy, eligiendo en entre las actividades 4 y 5 la de mayor valor de C'', en este caso la actividad 5 que se desplaza un intervalo. Se repite el método hasta que las actividades 5 y 6 se iniciarían a la vez. Se elige entonces entre las actividades 5 y 6 la de mayor C'' pero coinciden su valor, por lo que se elige la que necesita menor número de recursos pero vuelve a coincidir el valor, y finalmente se elige la actividad 6 con menor duración. Esta actividad se desplaza hasta que iría después de la actividad 5. En la *Ilustración 7* se refleja el Diagrama de Carga de la solución final, donde se observa que el *makespan,* en este caso, ha aumentado en 2 unidades de tiempo para encontrar una solución admisible con 4 recursos disponibles.

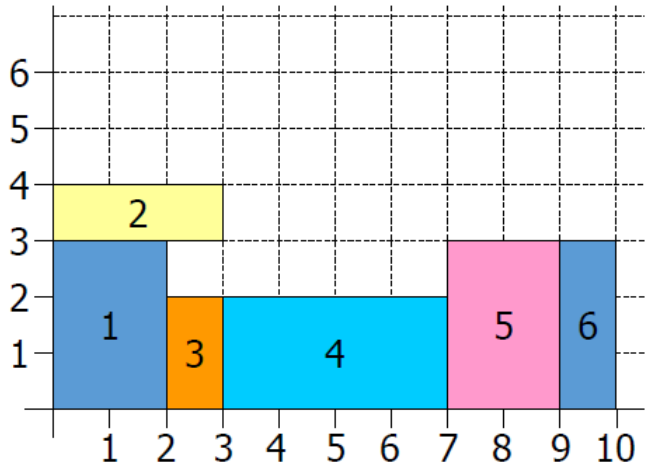


Ilustración 7: Ejemplo de Diagrama de Carga de Recursos con West-Levy para 4 recursos

Fuente: Eguia Salinas, Ignacio "Apuntes de la asignatura Producción Aeroespacial"

2.4.2.2 Métodos Basados en Reglas de Prioridad

Estos métodos utilizan como datos de entrada las salidas del método de camino crítico PERT/CPM. Es decir, para hallar la mejor solución posible del problema, se utilizan los tiempos de inicio y finalización más temprano y tardíos, así como las holguras, estos datos obtenidos son manejados para elaborar una serie de reglas de prioridad para poder ordenar y programar todas las actividades de una secuencia.

Para la resolución mediante este método es necesario, en primer lugar, generar una **secuencia (SGS)** y en segundo lugar, una **regla de prioridad** como se ha nombrado recientemente.

Esquema Generador de Secuencias (Schedule Generation Schemes, SGS)

El *SGS* determina cómo se forma una secuencia factible asignando los tiempos de inicio a cada actividad. La forma en que se construyen las secuencias factibles es creando una secuencia parcial en cada una de las etapa. Una secuencia parcial es aquella donde solamente un subconjunto de tareas tienen asignado un tiempo de finalización. El SGS construye un grupo de todas aquellas actividades que se pueden elegir en cada etapa donde, más tarde, se le asignará una o más actividades.

Existen dos esquemas generadores de secuencias: en serie y en paralelo.

- **En serie**: se forman tantas etapas como actividades haya que programar. A cada etapa del proyecto se le asigna una tarea respetando las relaciones de precedencia y sin exceder los recursos disponibles. Una vez que una actividad ha sido asignada a una etapa, esta se elimina del conjunto de actividades elegibles. Para determinar qué tarea va en cada etapa se utiliza una regla de prioridad y se programa lo antes que sea posible. Entre todas las posibles soluciones factibles que se puedan generar siempre se encontrará la óptima entre ellas. Este tipo de esquema fue propuesto por *Kelley (1963).*

- **En paralelo**: en el esquema en paralelo se programan tantas actividades como sea posible en cada tiempo de decisión. Este es el tiempo de finalización de las actividades en ejecución. Las secuencias que se generan son secuencias sin retraso alguno debido a que cualquier actividad que sea factible puede ser programada en un periodo de tiempo. En las posibles secuencias que se pueden generar, cabe la posibilidad de que ninguna de ellas sea la óptima.

Estos tipos de generados de esquemas de secuencias fueron desarrollados por *Lova & Tormos (2006)* y se aplican a los problemas multi-modos con recursos renovables.

Estos esquemas pueden aplicarse de dos formas distintas. Una de ellas es hacia adelante, es decir, empezando por la actividad ficticia inicial y acabando por la actividad ficticia final; y la otra de ellas es hacia atrás, es decir, de forma viceversa empezando por la actividad ficticia final y acabando por la actividad ficticia inicial invirtiendo, por supuesto, las relaciones de precedencia.

Reglas de Prioridad

Con las *reglas de prioridad* se determina qué actividad se programa en cada momento, es decir, las ordena de una forma específica. Esta ordenación respeta en todo momento las relaciones de precedencia, así como los niveles de recursos disponibles.

Existe una amplia literatura acerca de las reglas de prioridad y en ninguna de todas ellas existe alguna que tenga un mejor desempeño que las demás en todas las instancias.

De las muchas reglas de prioridad que se han desarrollado, a continuación se exponen tres de ellas que se consideran básicas:

- **Ranked Positional Weight Tecnique (RPWT):** Consiste en la asignación de un peso a cada actividad *j*. El peso es calculado por la suma de la duración de la tarea j más la suma de los pesos de sus sucesores. Se empieza por aquellas actividades sin sucesores, las cuales tendrán un peso con el valor de su duración, y a partir de ellas se van calculando el peso de sus predecesoras. Se priorizan las actividades con mayor peso.

$$v_j = p_j + \sum_{i \in D(j)} p_i$$

  Siendo p_i la duración de la tarea sucesora de la actividad *j* y *D (j)* el conjunto de todos los sucesores.

- **Latest Finish Time (LFT**): Esta regla se basa en los tiempos de finalización más tardíos. El peso de cada actividad *j* viene determinado por el tiempo de finalización más tardío de cada una de ellas, siendo este calculado por el método PERT/CPM, pero con el signo cambiando, es decir, negativo. De este modo, se prioriza a aquellas actividad con el peso de mayor valor.

$$v_j = -C_j''$$

- **Minimum Slack (MS):** Este caso está relacionado con las fechas de entrega de las actividades, es decir, para qué fechas tiene que estar realizada cada tarea. Los datos que se utilizaran para este caso son el tiempo más temprano de inicio, así como el tiempo más tardío de finalización y el tiempo de proceso de cada trabajo. Para su priorización se determinara el peso que, a su vez, es la resta del tiempo de proceso y del tiempo más temprano de inicio al tiempo de finalización y finalmente, al igual que en el LFT, se le cambia el signo para priorizar aquellas actividades con mayor valor del peso.

$$v_j = -\left(C_j'' - p_j - S_j'\right)$$

En los *Multi-Mode Resource Constrained Project Scheduling Problem,* como ya es sabido, existen diferentes modos para ejecutar las actividades. Por lo tanto, para priorizar las tareas habría que saber en qué modo se ejecutan cada una de ellas.

Por esta razón, *Boctor (1993) (1996)* plantea tres reglas de prioridad para la selección de los modos basándose en el esquema en paralelo de secuenciación:

- **P-SFM**: Consiste en escoger el modo, en el cual, el proyecto tarde lo menos posible. Es decir, escoger el modo en el que las actividades tenga el menor tiempo de proceso para minimizar al máximo el *makespan*. El inconveniente de esto es que, normalmente, en los modos con menor tiempo de ejecución el nivel de recursos es mayor, dificultando la realización de las actividades en paralelo.

- **P-LCR**: Se utiliza la idea de razón crítica. La razón crítica es la relación entre la utilización máxima de un recurso y la disponibilidad del mismo, siendo el recurso critico aquel que tenga mayor relación. Se calcula el tiempo de inicio más temprano de las actividades, y de ahí se consigue el número de recursos que se necesita en cada espacio de tiempo. El modo que se selecciona es aquel con menor cantidad del recurso más crítico.

- **P-LRP**: Se escoge el modo que minimiza la relación entre la cantidad de un tipo de recursos en un modo en concreto con la disponibilidad del mismo.

Todos estos métodos se pueden mejorar fácilmente para obtener soluciones de mayor calidad. Lo único que habría que hacer es aumentar el número de pasadas, es decir, repetir el proceso. Para ello, se combinan los dos esquemas de secuenciación, las direcciones de secuenciación y las reglas de prioridad obteniéndose así la mejor solución de la instancia.

2.4.2.3 Local Search

Local Search o *Búsqueda Local* consiste en, a partir de una solución obtenida por algún procedimiento, realizar pequeños movimientos en dicha solución para obtener una nueva solución mejor.

Este procedimiento se realiza hasta que no se encuentre ninguna mejor solución que la actual, es decir, se habría llegado a un óptimo local. Esto, que podría parecer bueno (y que no es malo ya que la solución óptima es un óptimo local) se convierte en una limitación del método. El espacio de soluciones suele tener muchos óptimos locales y de estos algunos serán buenos, malos o regulares.

Los movimientos son ligeras modificaciones en la solución actual en busca de alguna otra solución que sea mejor. Las soluciones que se generan a partir de la actual se denominan vecinas; y al conjunto de todas las soluciones, vecindad.

La clave de este procedimiento está en el movimiento que se realiza para la búsqueda de mejores soluciones. De esto va a depender si el algoritmo es eficiente o no. Estos

movimientos pueden generar soluciones aleatoriamente por todo el espacio de búsqueda o generar soluciones muy próximas a la actual.

Si la vecindad es pequeña, la búsqueda corre el riesgo de finalizar rápidamente en un óptimo local; mientras tanto, si la vecindad es grande dicho riesgo es mucho menor y será mucha más lenta. Local Search se basa en el *principio de optimalidad* (las soluciones buenas se encuentran muy cercas entre sí).

El problema que tiene este procedimiento en los RCPSP y MRCPSP es que la búsqueda normalmente queda atrapada en un óptimo local, y aunque se han mejorado en los últimos años siguen sin ser del todo eficientes.

2.4.2.4 Algoritmo Iterado Greedy

Este algoritmo normalmente se usa en combinación de otros. Consiste en la construcción de la solución de forma que se asigna el valor óptimo de la mejor variable disponible de forma independiente. Al aplicar cada asignación por separado, es muy complicado que la solución final sea la óptima.

El algoritmo *Greedy* es un método sencillo y rápido que forma la solución basándose en las características de los elementos que pueden formar a la misma. Para seleccionar la mejor alternativa en cada paso del algoritmo, se utiliza una regla heurística en función del problema en el que se esté. Esta regla se puede usar para dos casos: el primero de ellos es para generar una solución inicial. Y el segundo de ellos es para escapar de un óptimo local.

Para ello, primeramente, se "destruye" parte de la solución de forma aleatoria y, a continuación, se "reconstruye" la solución usando el algoritmo Greedy. Así, la nueva solución generada está muy próxima a la actual debido a que solo se modifican algunos de los elementos de la misma y no en su totalidad.

2.4.2.5 Tabu Search

La búsqueda *Tabu* fue presentada por *Fred Glover (1986)*. Este método consiste en la búsqueda de las mejores soluciones en distintos espacios dentro de las limitaciones de búsqueda, memorizando y guardando dichas soluciones en una lista denominada lista tabú.
La memoria puede guardar la solución entera o aquellas características más importantes de la solución.

Este proceso es diverso porque visita nuevas áreas no exploradas del espacio de búsqueda y añadir nuevas soluciones a la lista; e intensifica la búsqueda en algunas regiones ya exploradas y así poder analizarla mucho más a fondo.
El método finaliza después de un número determinado de iteraciones o cuando se encuentra una solución muy cercana a la deseada.

Desde que *Glover* estudio este método, muchos otros autores han desarrollado nuevos avances y teorías sobre él. *Baar (1999)* produce dos versiones de *Tabu Search* para los problemas con limitación de recursos. La primera de ellas el vecindario se forma a partir de tres movimientos basados en el CPM. El segundo se basa en un esquema de secuencia y en la generación de vecino y en una regla de descodificación propuesta por *Brucker (1998). Thomas & Salhi (1998)* ejecutan las secuencias definiendo tres movimientos diferentes. En esta teoría se necesita un procedimiento que asegure que todas las secuencias que se generan son factibles. *Klein (2000)* genera un procedimiento denominado RETAPS. *Nonobe e Ibaraki (2002)* utilizan una lista de actividades y una reducción de vecindario. En este enfoque se permite el caso de modos múltiples.

PSPLIB, que es la librería que se usará en este trabajo más adelante, utiliza la lista de actividades y el esquema en serie para decodificar la solución en este tipo de procesamiento.

2.4.2.6 Simulated Annealing

Kirkpatrick et al. (1983) son los desarrolladores de este método, el cual, se basa en el enfriamiento de materiales. No es un método muy eficiente pero sí muy fácil de programar.

Es una metaheurística muy parecida al Local Search pero con la diferencia de que, en este caso, los movimientos en búsqueda de una mejor solución se aceptan o no de una forma probabilística. Es decir, teniendo una solución aleatoria, el *Simulated Annealing* genera un vecino y calcula la diferencia entre las soluciones de ambos vecinos y genera una probabilidad para comprobar si el movimiento es de mejora o de empeoramiento.

$$P_{aceptación} = e^{-\frac{max\{0,\Delta f_{x\to x'}\}}{T}}$$

Si $\Delta f_{x\to x'} < 0$, el movimiento es de mejora ya que la probabilidad se acerca a 1. En caso contrario, la probabilidad se aleja de 1 y el movimiento es de empeoramiento.

El parámetro *T* puede tener distintos valores. A alta *T* se aceptan casi todos los movimientos aunque sean peores. Si la temperatura es baja apenas se aceptan movimientos de empeoramiento.

Por tanto, el parámetro *T* no tiene un valor fijo sino que se le van dando distintos valores durante el proceso. Primero se le dan valores altos (aceptándose casi todos los movimientos) y, poco a poco, la temperatura va disminuyendo muy lentamente (cada vez se aceptan menos movimientos)n y se llega a un óptimo local.

A lo largo de los años, muchos autores han desarrollado varias teorías acerca de esta metaheurística. *Boctor (1996)* resuelve el RCPSP mediante un algoritmo y lo compara con el TS de *Pinson (1994)* siendo el primero de ellos más eficiente. *Slowinski et al. (1994)* ayudan a resolver los MRCPSP mediante un sistema de apoyo a la decisión (*DSS*), el cual identifica estrategias para escoger las actividades que deben empezar en el caso de que hay conflictos con los recursos. *Jozefowska et al. (2001)* generan dos formas de resolver los MRCPSP. La primera de ellas cuenta con soluciones factibles respetando las relaciones de precedencia y

recursos. La segunda de ella también tiene en cuenta las soluciones no factibles y la utilización de recursos no renovables. *Bouleimen y Lecocq (2003)* resuelve tanto el problema de RCPSP como el de MRCPSP. Se utiliza una lista de actividades y otra de modos y el esquema en serie como método de decodificación. *Valls et al.* (2005) generan un *Simulated Annealing* utilizando los métodos de mejora hacia adelante y hacia atrás.

2.4.2.7 Algoritmos Genéticos

Esta metodología fue generada por *John Holland (1975)* y se basa en la evolución de los seres vivos. Consiste en que los individuos/organismos mejor adaptados/más fuertes tienen más posibilidades de sobrevivir y reproducirse mientras que los menos adaptados/menos fuertes desaparecen. Cada individuo es una solución del problema y tiene un valor que muestra su calidad.

Los principales elementos de los algoritmos genéticos son la codificación de los organismos, las funciones de evaluación, *selección*, *cruce* y *mutación*.

La población inicial p , está formada por un conjunto de soluciones factibles que se pueden producir de forma aleatoria. La función de evaluación determina un valor (*fitness*) a cada individuo para medir la calidad de cada solución. La función de selección establece qué individuos generarán a los nuevos.

En el cruce se intercambia información entre los padres, para así determinar qué padre dará la información para la generar la nueva solución. Mientras que en la mutación se modifica la información del individuo para obtener una mayor diversidad de soluciones en la población.

La nueva población generada reemplaza a la anterior y, para no perder la mejor solución obtenida hasta el momento, se traslada a la nueva población.

Los algoritmos genéticos se han estudiado y analizado a fondo para resolver problemas relaciones con la programación de proyectos. Muchos autores han abordado el tema como *Alcaraz y Maroto (2001), (2006)* o *Dbels (2005)* o *Hartman (2002)* que estudiaron la resolución del RCPSP mediante este método. Para los problemas de MRCPSP están los estudios de *Wall (1996),* que solo tienen en cuenta recursos renovables, o *Mori y Tsen (1997)* o *Alcaraz et al. (2003)* que trabajan el problema íntegro.

3. Modelos de programación matemática para la planificación de tareas

Los problemas de *Programación de Proyectos con Recursos Limitados* con un único modo o multimodo han sido analizados y estudiados por muchos autores a lo largo de su historia. Estos problemas tienen una forma estándar para su planteamiento como modelos de programación lineal, y que se presenta a continuación, y son referencias para otras variantes existentes.

3.1 Formulación del problema con limitaciones de recursos (RCPSP)

Los problemas de Programación de Proyecto con Recursos Limitados y con un único modo de ejecución de las tareas (RCPSP) son muy comunes en el mundo de los sistemas de producción. Son muchos los autores que han formulado este problema como *Pritsker et al. (1969), Kaplan (1988), Alvarez-Valdes & Tamarit (1993)* y *Mingozzi et al. (1998).*

A continuación, se presenta el modelo matemático de *Pritsker (1969)* con sólo recursos renovables:

- **Índices:**

  j: actividades (j=1, … , N+1) siendo la actividad N+1 la actividad ficticia final.
  t: períodos de tiempo (t=1, … , H)
  l: tipos de recursos renovables (l= 1, … , M)

- **Parámetros:**

  $\boldsymbol{p_j}$**:** tiempos de procesos de la actividad j
  $\boldsymbol{W_{lj}}$**:** número de recursos renovables necesarios de tipo l para la actividad j
  $\boldsymbol{W_l}$**:** número total de recursos renovables disponibles de cada tipo l
  $\boldsymbol{S_{(j)}}$**:** conjunto de actividades inmediatamente sucesoras de la actividad j

- **Variables:**

  $\boldsymbol{X_{jt}}$ { 1 si la actividad j termina en el tiempo t
  0 en cualquier otro caso

- **Modelo:**

$$Min\ Cmax = \sum_{t=1}^{H} t\ X_{N+1,t}$$

$$sujeto\ a,$$

$$\sum_{t=1}^{H} X_{jt} = 1 \qquad \forall j \qquad (1)$$

$$\sum_{j=1}^{N} W_{lj} \sum_{u=t}^{t+p_j-1} X_{ju} \leq W_l \qquad \forall l;\ \forall t \qquad (2)$$

$$\sum_{t=1}^{H} t\ X_{jt} \leq \sum_{t=1}^{H} t\ X_{j't} - p_{j'} \qquad \forall j;\ \forall j' \in S(j) \quad (3)$$

$$X_{jt} = 0,1 \quad \forall j;\ \forall t$$

El objetivo del modelo es minimizar el *makespan*.
La restricción (1) asegura que todas las actividades son procesadas.
La restricción (2) asegura que los recursos renovables del tipo *l* utilizados en un tiempo t no sobrepasan la disponibilidad de dicho recurso *l*.
La restricción (3) asegura que se cumplen las relaciones de precedencia (si la actividad *j* es predecesora de *j'*, el tiempo de finalización de *j'* es mayor o igual al tiempo de finalización de *j* más la duración de la actividad *j'*).

3.2 Formulación del problema con varios modos de ejecución (MRCPSP)

En el problema estándar de MRCPSP, el objetivo es la asignación de un modo de trabajo a cada actividad, así como un tiempo de inicio y finalización de cada una de ellas, de tal modo que se cumplan las relaciones de precedencia, disponibilidad de recursos y minimizar el tiempo de ejecución del proyecto. Si en algún caso hubiera un solo modo de ejecución de las actividades, este problema se convertiría en un RCPSP.

En los MRCPSP, se permiten diferentes formas de ejecutar cada actividad porque se usen diferentes recursos (máquinas, herramientas, personas) o número de recursos. Los recursos materiales como las herramientas o maquinaria tienen diferentes manejos o características por lo que el tiempo de ejecución de la actividad puede variar si se cambia de recurso material. También si una misma actividad puede ser realizada por una o varias personas, la duración de dicha actividad puede variar.

De esta forma, *Talbot (1982)*, introdujo un modelo de programación lineal para resolver este tipo de problemas. Se presenta a continuación:

- **Índices**

  j: actividades (j=1, … , N+1) siendo la actividad N+1 la actividad ficticia final.
  t: períodos de tiempo (t=1, … , H)
  l: tipos de recursos renovables (l= 1, … , MR)
  l': tipos de recursos no renovables (l'= 1, … , MNR)
  k: modo de ejecución de las actividades (k=1, … , K(j)); K(N+1)=1 para la ejecución de la actividad ficticia.

- **Parámetros**

  $\boldsymbol{p_{jk}}$**:** tiempo de proceso de la actividad j usando el modo k ($p_{N+1,k} = 0$)
  $\boldsymbol{WR_{ljk}}$**:** número de recursos renovables que la actividad j necesita del tipo l usando el modo k
  $\boldsymbol{WR_l}$**:** número total de recursos renovables del tipo l disponibles
  $\boldsymbol{WNR_{ljk}}$**:** número de recursos no renovables que la actividad j necesita del tipo l usando el modo k
  $\boldsymbol{WNR_l}$**:** número total de recursos no renovables del tipo l disponibles
  $\boldsymbol{S_{(j)}}$**:** conjunto de actividades sucesoras inmediatas de la actividad j

- **Variables:**

  $\boldsymbol{X_{jkt}}$ { 1 si la actividad j termina en el tiempo t usando el modo k
  0 en cualquier otro caso

- **Modelo**

$$Min\ Cmax = \sum_{t=1}^{H} t\ X_{N+1,1t}$$

$$sujeto\ a,$$

$$\sum_{k=1}^{K(j)} \sum_{t=1}^{H} X_{jkt} = 1 \qquad \forall j \qquad (1)$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K(j)} WR_{ljk} \sum_{u=t}^{t+p_j-1} X_{jku} \leq WR_l \qquad \forall l;\ \forall t \qquad (2)$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K(j)} WNR_{ljk} \sum_{t=1}^{H} X_{jkt} \leq WNR_{l'} \qquad \forall l' \qquad (3)$$

$$\sum_{k=1}^{K(j)} \sum_{t=1}^{H} t\, X_{jkt} \leq \sum_{k=1}^{K(j)} \sum_{t=1}^{H} t\, X_{j'kt} - p_{j'k} \qquad \forall j;\ \forall j' \in S(j) \quad (4)$$

$$X_{jkt} = 0,1 \quad \forall j;\ \forall k;\ \forall t$$

El objetivo es minimizar el *makespan* del proyecto, al igual que en los problemas RCPSP.
La restricción (1) asegura que las actividades son procesadas en un único modo.
La restricción (2) hace posible que la demanda de los recursos de tipo *l* no exceda de la disponibilidad de los recursos de dicho tipo en cada instante de tiempo *t*.
La restricción (3), al igual que la (2), asegura que la demanda de recursos del tipo *l'* no sobrepase la disponibilidad del mismo tipo de recurso en todo el proyecto.
Por último, la restricción (4), asegura que se cumplan las relaciones de precedencia. De este modo, el tiempo de finalización de la actividad *j'* tiene que ser más grande o igual que el tiempo de finalización de la actividad anterior más el tiempo de proceso de *j'* dependiendo del modo en la que se ejecuta dicha tarea.

4. El problema en estaciones de montaje aeronáuticas

El diseño de los sistemas de producción se basa en los datos y características del producto, así como en las limitaciones tecnológicas de los recursos que se utilizan para su fabricación. Este diseño ayuda a resolver problemas de asignación de recursos o de distribución de los mismos en las zonas de trabajo.

Como ya se explicó anteriormente, una cadena de producción o línea de montaje es un tipo de sistema de producción que divide la fabricación de un producto en diferentes estaciones ejecutando en cada una de ellas una función específica. Su objetivo es obtener la máxima eficiencia posible aumentando la producción y disminuyendo los costes. Una línea de montaje viene marcada por una cadencia de producción denominada *"takt-time"* o tiempo de ciclo, que indica el tiempo que cada producto permanece en cada estación y por tanto marca el intervalo de tiempo entre la salida de dos productos finales en la última estación.

Este trabajo se centra en la planificación de tareas en una estación de montaje de aeronaves con limitaciones de recursos humanos. Algunos desafíos de la investigación operativa en las líneas de montaje de aeronaves se identificaron en Scott (1994). El autor identificó cinco aspectos que suelen caracterizar a las tareas con trabajadores en una línea de montaje de aeronaves: la representación de relaciones de precedencias entre tareas, el uso de trabajadores con diferentes habilidades y nivel de competencia (diferentes perfiles), la reasignación del trabajador a nuevas tareas, la consideración de los turnos y las horas extras, y el modelado de las restricciones de espacio y de los movimientos de trabajadores dentro de las zonas de trabajo. También en Borreguero et al. (2015) se caracterizan las estaciones de trabajo de una línea de montaje final de aeronaves frente a otros sectores como el de automoción o el electrónico. Las principales diferencias son:

- El número de relaciones de precedencia entre tareas es bajo. La mayoría de las tareas tienen una única tarea predecesora directa.

- La mayoría de las tareas son realizadas por un único perfil de trabajador.

- El número de perfiles diferentes en una estación de montaje no es elevado (no superior a 4 perfiles).

- El número de trabajadores que puede realizar cada tarea puede variar. Los tiempos de ejecución de la tarea cambia según el número de trabajadores que la realiza. En general no se requiere un número elevado de trabajadores en cada tarea (no superior a 3 trabajadores).

El problema más común a resolver en una estación de montaje de aeronaves es similar al de cualquier otra con recursos: la programación de las diferentes actividades asignadas a la estación respetando el tiempo de ciclo o *takt-time* de la línea, y la asignación de los recursos que realizarán cada actividad.

Este problema en sí es exactamente igual que los problemas estándar RCPSP y MRCPSP explicados en puntos anteriores, pero con las características antes mencionadas y que son propias del montaje de aeronaves. A los problemas RCPSP se le añaden restricciones de limitación de zonas de trabajo y modos asociados al número de trabajadores que puede ejecutar cada tarea. Ambas características están relacionadas la una con la otra. Por lo tanto, el conjunto de características de este tipo de problemas para las líneas de montaje en el campo de la aeronáutica quedaría de la siguiente forma:

- La línea de montaje está diseñada para un solo producto.
- El número de estaciones que forman la línea es conocido.
- El tiempo de ciclo (*takt-time*) es conocido y el mismo para todas las estaciones.
- Cada estación de trabajo está conformada por un número de tareas conocidas y su tiempo de ejecución final no puede sobrepasar el tiempo de ciclo.
- Cada tarea tiene un tiempo de proceso conocido que puede variar por el número de recursos implicados (trabajadores utilizados en la tarea).
- Las tareas pueden tener relaciones de precedencias con otras tareas (de tipo Fin-Comienzo).
- Cada tarea tiene una serie de recursos humanos (trabajadores) asignados según su especialidad/habilidad (perfil). La cantidad de trabajadores puede variar entre un mínimo y un máximo.
- Cada estación está dividida en zonas de trabajo.
- Cada zona de trabajo tiene una capacidad, es decir, un número máximo de trabajadores que pueden estar a la vez en la zona independiente del perfil.

A continuación, se explican más en detalle las nuevas características de las líneas de montaje aeronáuticas, y cómo se van a incluir en los modelos de programación lineal para el problema MRCPSP.

4.1. Limitaciones de zona de trabajo

Las limitaciones de trabajadores en zonas de trabajo es una característica específica para este tipo de líneas de montaje. Las estaciones de trabajo aeronáuticas se dividen en zonas de trabajo, para organizar las tareas que tienen lugar en la misma (*Ilustración 8*).

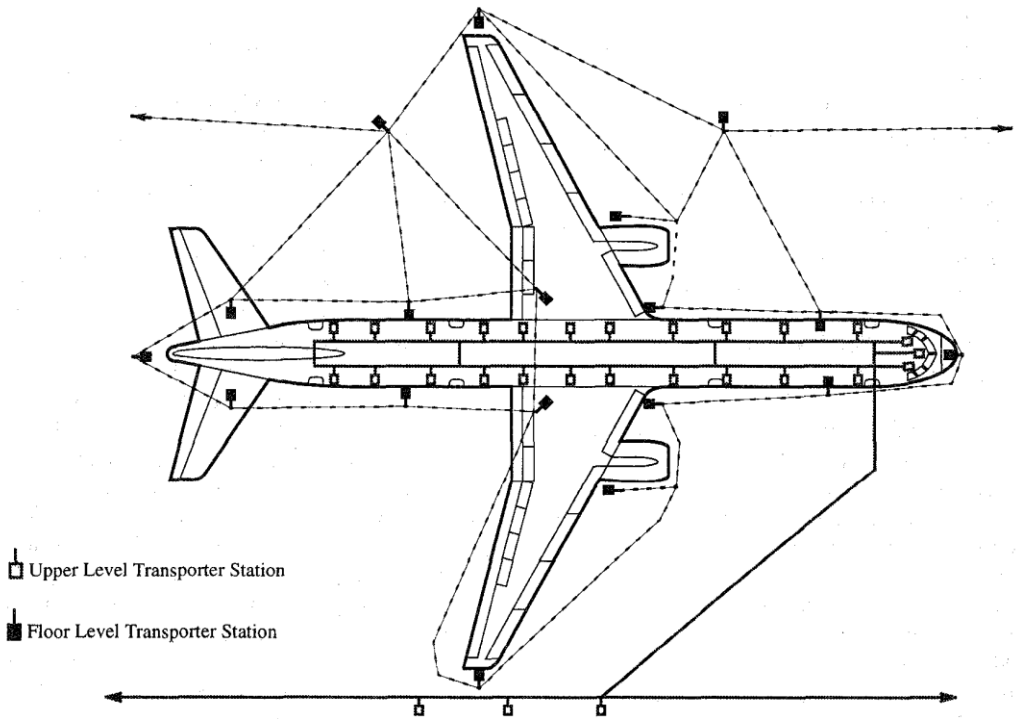


Ilustración 8: Ejemplo de definición de zonas de trabajo en una estación aeronáutica

Fuente: Scott (1994) "Modelling Aircraft Assembly Operations"

El porqué de esta limitación se debe a diversos motivos. Uno de ellos puede ser por condiciones de espacio físicos ya que algunas tareas necesitan un espacio mayor que otras. Otro puede ser por tener que usar las mismas herramientas ya que si dos o más trabajos necesitan las mismas máquinas o herramientas es conveniente que estén lo más próximas posibles para no tener que desplazarlas o duplicarlas o desplazar el producto (algo impensable en este tipo de industria), lo cual incurre en costes muy elevados. Otro motivo de la aplicación de la limitación de zona son las condiciones físicas. Varios de los materiales o partes del producto tienen que protegerse para que no tengan defecto alguno ante las diversas condiciones que puedan darse.

Las zonas de trabajo tienen un número máximo de posiciones y por tanto está limitado el número de trabajadores que pueden trabajar a la vez en cada zona. Por otro lado, cada tarea necesita una cantidad de trabajadores mínima y máxima para su realización. Además, cada tarea se realiza explícitamente en una zona de trabajo. Y finalmente, existe una limitación del número de trabajadores disponibles en cada estación. Por ello, en cada instante de tiempo, la cantidad de trabajadores asignados a todas las tareas no puede superar el máximo disponible, y la cantidad de trabajadores asignados a todas las tareas de una zona no puede superar la capacidad de dicha zona.

Por lo cual, la asignación de los recursos es un tema muy importante que se debe analizar para que todas las restricciones se cumplan y pueda alcanzarse una solución factible.

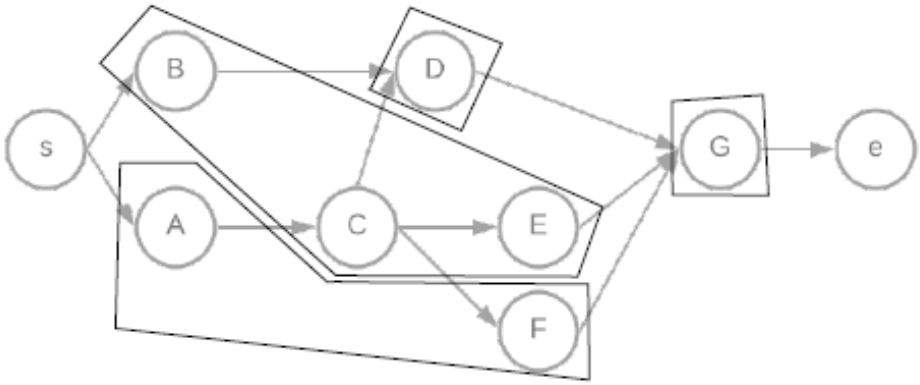


Ilustración 9: Ejemplo de representación por zonas de tareas de una estación.

(Elaboración propia)

La *Ilustración 9* es la representación PERT de una estación. Las diferentes secciones que rodean las actividades son las distintas zonas que dividen la estación y donde se realizan unos trabajos concretos por motivos de espacio, condiciones físicas o por recursos.

4.2. Número de trabajadores por cada tarea

En este Trabajo de Fin de Grado solo se van a tener en cuenta los recursos renovables, obviando los no renovables y los recursos parcialmente (no) renovables. El carácter fundamentalmente manual de las tareas de una estación de montaje aeronáutica hace que se estudie exclusivamente los recursos humanos (trabajadores) como únicos recursos renovables de la misma.

Los trabajadores de una estación de montaje aeronáutica pueden estar dotados de una o más habilidades/especialidades (*skills*). En la industria aeronáutica, normalmente, cada trabajador solo tiene una especialidad debido a la alta cualificación que se debe tener para realizar cada actividad. Por esta razón, se tendrá en cuenta que cada trabajador tiene una sola especialidad y habrá un número determinado de especialidades. De cada especialidad se dispone de un número máximo de trabajadores para la estación de montaje.

Cada tarea a realizar en una estación de montaje la realizará un número de trabajadores de una única especialidad. Aunque existe un número óptimo de trabajadores para la realización de cada tarea, en una estación de montaje aeronáutica se permite que dicho número pueda variar entre un mínimo y un máximo de trabajadores para algunas tareas concretas. La selección del número de trabajadores en esas tareas se convierte en una variable de decisión y tiene como consecuencia distintos tiempos de ejecución de las tareas en función del número de trabajadores (*Ilustración 10*).

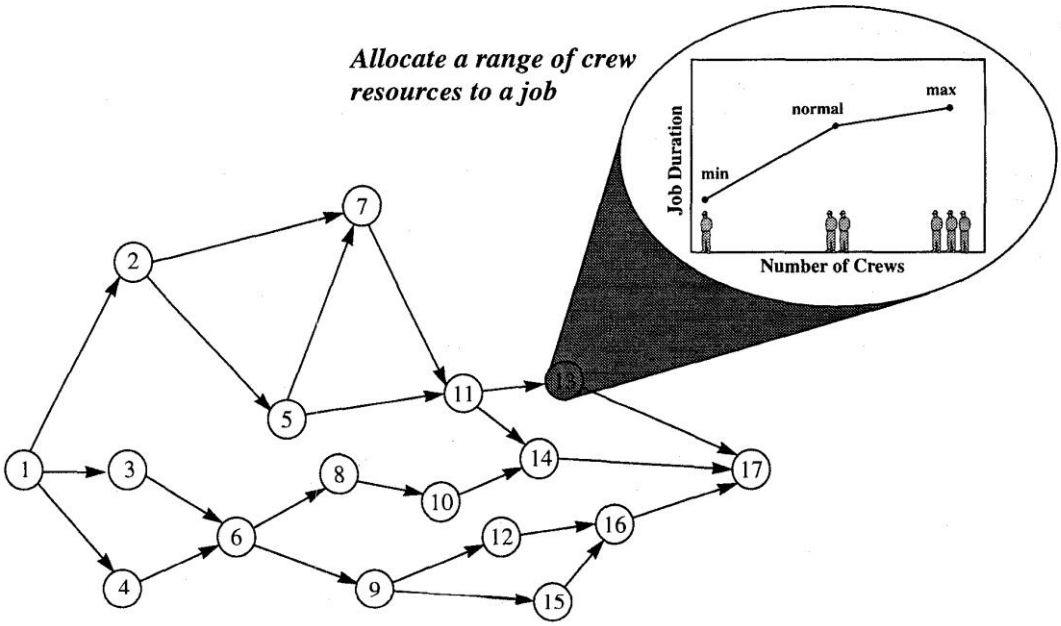


Ilustración 10: Ejemplo del nº de trabajadores que pueden ejecutar una tarea en una estación aeronáutica

Fuente: Scott (1994) "Modelling Aircraft Assembly Operations"

Por tanto, la selección del número de recursos (trabajadores) a realizar en cada tarea tiene efectos tanto en el tiempo de ejecución final de la estación (*makespan*) como en los instantes de inicio de cada tarea. Por un lado, a la hora de determinar los recursos hay que tener en cuenta que un aumento del tiempo en las tareas críticas retrasaría el *makespan* de la estación. Y por otro lado, los recursos renovables son limitados y no se pueden superar los disponibles en cada instante de tiempo.

4.3. Modelo de programación lineal en líneas de montaje aeronáuticas

Para plantear el modelo de programación lineal donde se incluyan las nuevas características asociadas a una estación de montaje aeronáutica, tomaremos como referencia el problema MRCPSP de la siguiente forma:

a) Las limitaciones de trabajadores en las zonas se incluirán como nuevas restricciones del modelo de referencia MRCPSP
b) La selección del número de trabajadores utilizados en cada tarea se incluirá como formas de realizar cada tarea, es decir, cada modo de trabajo de una tarea se corresponde con un número de trabajadores: el modo 1 será para el nº mínimo y el modo K(j) será para el nº máximo de trabajadores que pueden hacer la tarea j.
c) Las especialidades (*skills*) de cada trabajador se corresponden con los tipos de recursos renovables del modelo MRCPSP.

d) El tiempo de ciclo o *takt-time* de una línea de montaje aeronáutica marca el máximo tiempo disponible para la realización de todas las tareas asignadas a cada estación de montaje (*makespan*). Por tanto, se añadirá una restricción que cumpla dicho criterio.
e) La función objetivo más utilizada en los problemas MRCPSP consiste en minimizar el *makespan*. Esta función no tiene mucho sentido cuando se fija un tiempo de ciclo pues como se acaba de comentar, en toda estación de montaje el *makespan* tiene que ser menor que el tiempo de ciclo o *takt-time* de la línea. Por eso, el uso intensivo de mano de obra sugiere que el criterio más adecuado se debe basar en los costes de los recursos. En concreto, se va a incluir una nueva función objetivo basada en minimizar el coste total de los trabajadores, a partir de los costes unitarios de cada trabajador por especialidad, del total de trabajadores necesarios en la estación y del tiempo de ciclo.

A continuación, se propone un modelo lineal basado en el problema MRCPSP donde se añaden limitaciones de zonas y se mantienen las limitaciones de cada tipo de recurso o especialidad. Además se plantearán varias funciones objetivos basadas en minimizar el makespan y en minimizar el coste total de los trabajadores. Cabe recordar que el problema MRCPSP es una variante del problema RCPSP en el cual hay varios modos de trabajo en las tareas. En nuestro caso, cada modo de trabajo de una tarea se corresponde a un número de trabajadores asignado a dicha tarea (en un RCPSP solo hay un único modo de trabajo).

Con todo ello,

- **Índices**

  j: tareas (j=1, … , N+1) siendo la tarea N+1 la tarea ficticia final.
  t: períodos de tiempo (t=1, … , H)
  l: tipos de recursos renovables o especialidad del trabajador (l= 1, … , M)
  z: zonas de trabajo (z= 1, … , Z)
  k: modo de ejecución de la tarea *j* (k=1, … , K(j)); el modo *k* necesita un número específico de trabajadores con una especialidad concreta

- **Parámetros**

  $\boldsymbol{p_{jk}}$**:** tiempo de proceso de la tarea *j* usando el modo *k*
  $\boldsymbol{W_{ljk}}$**:** número de trabajadores que la tarea *j* utiliza de la especialidad *l* usando el modo *k*
  $\boldsymbol{W_l}$**:** número total de trabajadores de la especialidad *l* disponibles
  $\boldsymbol{S_{(j)}}$**:** conjunto de tareas sucesoras inmediatas de la tarea *j*
  CYCLE: tiempo de ciclo de la línea de montaje
  $\boldsymbol{WK_z}$: capacidad de la zona *z* (número máximo de trabajadores permitidos en la zona *z* en cualquier instante de tiempo
  $\boldsymbol{WC_l}$: coste unitario (de cada trabajador por unidad de tiempo) para la especialidad *l*

  $\boldsymbol{K_{zj}}$ { 1 si la tarea *j* se ejecuta en la zona *z*
  0 en cualquier otro caso

- **Variables:**

  $\boldsymbol{CMax}$: makespan (tiempo de finalización de la última tarea en la estación)
  $\boldsymbol{WMax_l}$: número máximo de trabajadores de la especialidad l necesario en algún instante de tiempo

  $\boldsymbol{X_{jkt}}$ { 1 si la tarea j termina en el tiempo t usando el modo k
  0 en cualquier otro caso

- **Modelo:**

$$Minimizar\ \{F1; F2\}$$

$$F1 = CMax$$

$$F2 = \sum_{l=1}^{M} WMax_l \cdot WC_l \cdot CYCLE$$

$$sujeto\ a$$

$$\sum_{k=1}^{K(j)} \sum_{t=1}^{H} X_{jkt} = 1 \qquad \forall j \qquad (1)$$

$$\sum_{k=1}^{K(j)} \sum_{t=1}^{H} t\, X_{jkt} \leq \sum_{k=1}^{K(j)} \sum_{t=1}^{H} t\, X_{j'kt} - p_{j'k} \qquad \forall j;\ \forall j' \in S(j) \quad (2)$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K(j)} \sum_{l=1}^{M} K_{zj}\, W_{ljk} \sum_{u=t}^{t+p_j-1} X_{jku} \leq WK_z \qquad \forall z;\ \forall t \qquad (3)$$

$$\sum_{k=1}^{K(j)} t\, X_{jkt} \leq CMax \qquad \forall j;\ \forall t \qquad (4)$$

$$CMax \leq CYCLE \qquad (5)$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K(j)} W_{ljk} \sum_{u=t}^{t+p_j-1} X_{jku} \leq WMax_l \qquad \forall l;\ \forall t \qquad (6)$$

$$WMax_l \leq W_l \qquad \forall l \qquad (7)$$

$$X_{jkt} = 0{,}1 \quad \forall j;\ \forall k;\ \forall t$$

El modelo tiene dos objetivos: uno es minimizar el makespan y el otro es minimizar el coste total de los trabajadores en la estación, es decir, la suma para todas las especialidades de multiplicar el coste unitario de cada trabajador de ese perfil por el número máximo de trabajadores necesarios y por todo el tiempo de ciclo de la estación. Se considera pues que es necesario que el trabajador esté en la estación todo el tiempo de ciclo aunque haya instantes de tiempos ociosos.

Las restricciones (1) indican que todas las tareas tienen que ser ejecutadas.

Las restricciones (2) hacen posible que las relaciones de precedencia de las tareas se cumplan.

Las restricciones (3) aseguran que el número máximo de trabajadores por zona es respetado en cada instante de tiempo.

Las restricciones (4) aseguran que el tiempo de finalización de cualquier tarea es menor o igual que el *makespan*; mientras que la restricción (5) obliga a que el *makespan* tiene que ser como máximo el tiempo de ciclo.

Las restricciones (6) aseguran que el número de trabajadores de la especialidad *l* necesarios para hacer las tareas en cada instante *t* es menor que el máximo necesario en todo el tiempo de ciclo; mientras que las restricciones (7) obliga a que dicho valor máximo necesario en cada especialidad *l* no sobrepase la cantidad disponible de trabajadores de dicho tipo de especialidad para la estación.

5. Experimentación

En este capítulo, como ya se ha citado anteriormente, se van a formular y resolver 3 tipos de experimentos:

(1) Experimentación sobre problemas RCPSP de la librería PSPLIB de *Kolisch* y *Sprecher (1997)*
(2) Experimentación sobre problemas MRCPSP de la librería PSPLIB de *Kolisch* y *Sprecher (1997)*
(3) Experimentación sobre nuevos problemas MRCPSP, basados en la librería PSPLIB de *Kolisch* y *Sprecher (1997)*, a los que se les añade ciertas características específicas como las zonas de trabajo.

Los modelos desarrollados para estos tipos de problemas pueden ser de gran utilidad en otras discusiones del mismo ámbito de *scheduling*. Por esta razón, es de gran importancia el estudio de los mismos.

Para la resolución de estos experimentos se ha utilizado el programa *Lingo 18.0 x64* de *Microsoft*. El recurso utilizado para la experimentación es un *Windows 10 Home* con un procesador *Intel(R) Core (TM) i7-8550U* de 2GHz y 12GB en *RAM*.

Este capítulo está estructurado de forma que, en la sección 5.1, se explican las instancias de los problemas RCPSP y MRCPSP de la literatura. En la sección 5.2 se exponen las instancias nuevas generadas con las restricciones de zonas. En la sección 5.3 se detalla el método de resolución de los mismos; mientras que en 5.4 se analizan los resultados obtenidos de los apartados anteriores.

5.1 Librería de problemas de la literatura

A lo largo de los años, fueron muchos los procedimientos heurísticos y exactos que surgieron a raíz de los problemas de programación de proyectos. Muchos investigadores generaron sus propios problemas, y esto llevaba a una gran controversia en la comparación de los métodos para resolver los mismos.

La librería PSPLIB *(Kolisch y Sprecher ,1997)* contiene diferentes conjuntos de instancias para varios tipos de problemas de programación de proyectos con recursos limitados, así como las soluciones óptimas de muchas de ellas obtenidas por métodos exactos.

Los conjuntos de datos se pueden utilizar para la evaluación de procedimientos de solución para problemas de programación de proyectos con recursos limitados de modo único y multimodo. Las instancias han sido generadas por el generador de proyectos estándar *ProGen*, propuesto por *Kolisch et al. (1995). ProGen* utiliza un diseño de experimentos factorial basando en dos conjuntos de parámetros. El primero de ellos está compuesto por parámetros que son iguales para cada conjunto de prueba:

- Número de actividades
- Número de modos en que cada actividad puede ejecutarse
- Número de recursos renovables existentes
- Disponibilidad máxima de cada recurso renovable
- Número de recursos no renovables existentes
- Disponibilidad máxima de cada recurso no renovable
- Número de sucesores de la actividad ficticia inicio
- Número de predecesores de la actividad ficticia fin
- Número de sucesores y predecesores de las actividades no ficticias
- Duración de las actividades

El segundo conjunto son parámetros variables, es decir, que cambian de uno a otro:

- Complejidad de la Red: número medio de relaciones de precedencia no repetidas por cada actividad.
- Factor de Recurso: expresa el número de recursos diferentes que las actividades no ficticias utilizan.
- Grado de restricción de los recursos: relación entre los recursos necesarios y la disponibilidad de los mismos.

5.1.1 Librería de problemas RCPSP de la literatura

Los primeros experimentos (1), se han ejecutado para los problemas *j30* de la PSPLIB de Kolisch para RCPSP. Hay que recordar que en este tipo de problemas las tareas solo se pueden realizar de un único modo, por lo que cada actividad tiene un único tiempo de ejecución.

Los problemas *j30* se corresponden con 480 instancias en 48 grupos (*Parameter*) de 10 instancias por grupo. Todas ellas tienen 30 trabajos reales más dos ficticios (inicial y final). Cada uno de ellos posee entre 1 y 3 sucesores. Los recursos de estas instancias son de 4 tipos diferentes, todos ellos renovables.

El formato de los ficheros *j30* de la librería PSPLIB es el mismo para todos ellos. A modo de ejemplo se muestra el formato del fichero "*j301_1.sm*" correspondiente a la instancia "*1_1*":

```
******************************************************************************
file with basedata            : j30_17.bas
initial value random generator: 28123
******************************************************************************
projects                      :  1
jobs (incl. supersource/sink ):  32
horizon                       :  158
RESOURCES
  - renewable                 :  4   R
  - nonrenewable              :  0   N
  - doubly constrained        :  0   D
******************************************************************************
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  MPM-Time
    1     30      0       38       26       38
******************************************************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
   2        1          3           6  11  15
```

```
  3        1          3           7   8  13
  4        1          3           5   9  10
  5        1          1          20
  6        1          1          30
  7        1          1          27
  8        1          3          12  19  27
  9        1          1          14
 10        1          2          16  25
 11        1          2          20  26
 12        1          1          14
 13        1          2          17  18
 14        1          1          17
 15        1          1          25
 16        1          2          21  22
 17        1          1          22
 18        1          2          20  22
 19        1          2          24  29
 20        1          2          23  25
 21        1          1          28
 22        1          1          23
 23        1          1          24
 24        1          1          30
 25        1          1          30
 26        1          1          31
 27        1          1          28
 28        1          1          31
 29        1          1          32
 30        1          1          32
 31        1          1          32
 32        1          0
************************************************************************
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2  R 3  R 4
------------------------------------------------------------------------
  1      1     0       0    0    0    0
  2      1     8       4    0    0    0
  3      1     4      10    0    0    0
  4      1     6       0    0    0    3
  5      1     3       3    0    0    0
  6      1     8       0    0    0    8
  7      1     5       4    0    0    0
  8      1     9       0    1    0    0
  9      1     2       6    0    0    0
 10      1     7       0    0    0    1
 11      1     9       0    5    0    0
 12      1     2       0    7    0    0
 13      1     6       4    0    0    0
 14      1     3       0    8    0    0
 15      1     9       3    0    0    0
 16      1    10       0    0    0    5
 17      1     6       0    0    0    8
 18      1     5       0    0    0    7
 19      1     3       0    1    0    0
 20      1     7       0   10    0    0
 21      1     2       0    0    0    6
 22      1     7       2    0    0    0
 23      1     2       3    0    0    0
 24      1     3       0    9    0    0
 25      1     3       4    0    0    0
 26      1     7       0    0    4    0
 27      1     8       0    0    0    7
 28      1     3       0    8    0    0
 29      1     7       0    7    0    0
 30      1     2       0    7    0    0
 31      1     2       0    0    2    0
 32      1     0       0    0    0    0
************************************************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  R 3  R 4
   12   13    4   12
************************************************************************
```

En dicho formato se observa,

- 30 trabajos reales y 2 ficticios: `jobs (incl. supersource/sink ):  32`

- 4 tipos de recursos renovables: `- renewable:  4   R`
- Cada actividad tiene el nº de modos (1), el nº de sucesoras y los códigos de las sucesoras:

```
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
```

- Cada actividad y modo tiene la duración y el nº de recursos necesarios de cada tipo:

```
jobnr. mode duration  R 1  R 2  R 3  R 4
  1      1     0       0    0    0    0
```

- Cada tipo de recurso tiene la cantidad máxima disponible:

```
  R 1  R 2  R 3  R 4
   12   13    4   12
```

Para los problemas *j30*, el valor del *Makespan* es conocido en cada instancia, así como el *CPU-Time* de aplicar un método de Branch & Bound *(Demeulemeester y Herroelen, 1997)*. Dichos experimentos se hicieron en un IBM PC PS2 Model p75 con procesador 80486 a 25 MHz y 16 MB de RAM.

La experimentación (1) en este TFG se ha realizado sobre las mismas 480 instancias *j30*, ejecutando el modelo de programación lineal descrito en el apartado 3.1 usando el software *Lingo*. Las instancias se hacen de forma ordenada, de menor a mayor número de variables binarias (NVB). Como se conoce el *Makespan* de cada instancia, se ha considerado como número de periodos (H) dicho valor. Por tanto,

$$NVB = (32 - 1) \times Makespan$$

La función objetivo para validar el modelo ha sido minimizar el *Makespan*.

ORDEN	Parameter	Instance	Makespan	CPU-Time [sec.]	Nº Var Bin
1	16	7	34	0,01	1054
2	12	8	35	0,01	1085
3	32	7	35	0,01	1085
4	7	6	35	0,02	1085
5	14	6	35	0,75	1085
6	16	3	36	0,01	1116
7	11	7	36	0,5	1116
8	20	10	37	0	1147
9	12	3	37	0,01	1147
10	6	6	37	0,04	1147
11	4	9	38	0,01	1178
12	19	9	38	0,01	1178
13	24	8	38	0,01	1178
14	2	1	38	0,02	1178
15	11	10	38	0,02	1178
16	8	9	39	0,01	1209
17	19	4	39	0,02	1209
18	6	8	39	0,36	1209
19	1	5	39	0,48	1209
20	19	1	40	0,01	1240

Ilustración 11: Ejemplo de tabla recogida de datos de problemas RCPSP

En la *Ilustración 11* se puede observar el orden en el cual se van a ejecutar los primeros 20 experimentos indicando, en cada uno de ellos, el NVB, así como el *makespan* correspondiente y el *CPU-Time* original.

5.1.2 Librería de problemas MRCPSP de la literatura

Los segundos experimentos (2), se han ejecutado para los problemas *n0* de la PSPLIB de Kolisch para MRCPSP. En los problemas MRCPSP las tareas tienen más de un modo de ejecución posible, por lo tanto tienen varios tiempos de ejecución posibles. En su resolución habría que seleccionar aquel modo que resulte más óptimo para el problema.

Los problemas *n0* se corresponden con 480 instancias en 48 grupos (*Parameter*) de 10 instancias por grupo. Estos ejemplos tienen entre 10 y 20 trabajos reales más dos ficticios, y 3 modos de trabajo en cada uno de ellos. Cada trabajo tiene entre 1 y 3 sucesores; y 2 tipos de recursos renovables.

El formato de los ficheros *n0* de la librería PSPLIB es el mismo para todos ellos y similar al de los *j30* ya descritos. A modo de ejemplo se muestra el formato del fichero "*n01_2.mm*" correspondiente a la instancia "*1_2*":

```
************************************************************************
file with basedata            : me1_.bas
initial value random generator: 466396357
************************************************************************
projects                      :  1
jobs (incl. supersource/sink ):  12
horizon                       :  65
RESOURCES
  - renewable                 :  2   R
  - nonrenewable              :  0   N
  - doubly constrained        :  0   D
************************************************************************
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  MPM-Time
    1     10      0        8        6        8
************************************************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
   2        3          3           5   7   9
   3        3          2           8   9
   4        3          3           5   6   7
   5        3          2          10  11
   6        3          2           8   9
   7        3          2          10  11
   8        3          2          10  11
   9        3          1          12
  10        3          1          12
  11        3          1          12
  12        1          0
************************************************************************
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2
------------------------------------------------------------------------
  1      1     0       0    0
  2      1     1       0   10
         2     6       6    0
         3     9       0    8
  3      1     3       4    0
         2     5       0    7
         3     7       0    4
  4      1     1       7    0
```

```
        2     5       0   10
        3     8       5    0
  5     1     1       0    5
        2     2       0    4
        3     6       2    0
  6     1     2       0    3
        2     3       8    0
        3     6       7    0
  7     1     6       6    0
        2     8       4    0
        3     9       3    0
  8     1     1      10    0
        2     2       8    0
        3     3       0    6
  9     1     1       7    0
        2     4       6    0
        3     4       0    6
 10     1     1       6    0
        2     1       0    7
        3     3       0    6
 11     1     1       0    3
        2     8       4    0
        3    10       3    0
 12     1     0       0    0
************************************************************************
RESOURCEAVAILABILITIES:
  R 1  R 2
    7    7
************************************************************************
```

En dicho formato se observa,

- 10 trabajos reales y 2 ficticios: `jobs (incl. supersource/sink ):  12`
- 2 tipos de recursos renovables: `- renewable:  2   R`
- Cada actividad tiene el nº de modos, el nº de sucesoras y los códigos de las sucesoras:

```
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
   2        3          3           5   7   9
```

- Cada actividad y modo tiene la duración y el nº de recursos necesarios de cada tipo:

```
jobnr. mode duration  R 1  R 2
  1      1     0       0    0
  2      1     1       0   10
         2     6       6    0
         3     9       0    8
```

- Cada tipo de recurso tiene la cantidad máxima disponible:

```
  R 1  R 2
    7    7
```

De igual modo que en el caso anterior, se conocen el valor mínimo de *makespan* y el *CPU-Time* de aplicar un método de Branch & Bound *(Sprecher y Drexl, 1996).* Dichos experimentos se hicieron en un IBM PC con procesador 80486dx a 66 MHz y 16 MB de RAM.

En este caso, el orden de la experimentación se produce del mismo modo que anteriormente, es decir, en función del número de variables binarias (NVB), el cual se calcula de la siguiente forma:

$$NVB = (NT \times NM \times Makespan) - (4 \times Makespan)$$

Donde *NT* es el número de trabajos (=12, 14,…, 22), *NM* es el número de modos (=3) y *Makespan* es el valor conocido en la literatura. El miembro de la derecha se debe a que hay variables binarias fijadas en los modelos.

La función objetivo sigue siendo minimizar el *Makespan* para validar el modelo.

ORDEN	Parameter	Instance	Makespan	CPU-Time [sec.]	Nº Trabajos	Nº Var Bin
1	3	9	9	0,03	12	288
2	7	8	9	0,03	12	288
3	4	7	10	0	12	320
4	15	3	9	0	14	342
5	3	1	11	0	12	352
6	8	3	11	0	12	352
7	4	3	11	0,04	12	352
8	12	8	10	0,03	14	380
9	3	4	12	0	12	384
10	4	8	12	0	12	384
11	4	9	12	0,03	12	384
12	8	9	12	0,03	12	384
13	6	8	12	0,04	12	384
14	2	3	13	0	12	416
15	3	2	13	0	12	416
16	3	6	13	0	12	416
17	8	5	13	0	12	416
18	4	2	13	0,03	12	416
19	6	3	13	0,03	12	416
20	12	2	11	0,03	14	418

Ilustración 12: Ejemplo de tabla recogida de datos de problemas MRCPSP

En la *Ilustración 12* se muestra el orden en el cual se van a ejecutar los primeros 20 experimentos indicando, en cada uno de ellos, el NVB y el *makespan*, así como el *CPU-Time* del método de optimización original. También se indica el número de trabajos que compone cada una de las instancias.

5.2 Librería de problemas propios

En este punto, como ya se ha citado anteriormente, se explica el diseño de los problemas propios elaborados para una estación de montaje aeronáutica.

Con estos experimentos se quieren probar una serie de características, que a continuación se citan, de este tipo de estaciones respecto a los problemas estándares de RCPSP y MRCPSP.

- **A.** Cada tarea puede ser realizada por un número de trabajadores de una especialidad, el cual puede variar. A más trabajadores, el tiempo de ejecución de la tarea es menor. De esta forma, se considera que cada modo de trabajo se corresponde con un número determinado de trabajadores de una especialidad realizando dicha actividad.
- **B.** Existen zonas de trabajo en la estación. Cada tarea se realiza en una zona de trabajo.
- **C.** Existe una limitación del número de trabajadores para trabajar a la vez en cada zona.
- **D.** El tiempo de ciclo de la estación es mayor o igual al *makespan*.
- **E.** Cada especialidad de trabajador tiene un coste unitario (€/hora). Se estudiará el coste total de los trabajadores necesarios durante el tiempo de ciclo en la estación.

Al tener en cuenta todas estas características, el problema a tratar se convierte en un MRCPSP por (A) con restricciones de zonas (B y C). Además, hay que fijar un tiempo de ciclo (D); y existirá una nueva función objetivo a estudiar (E), además del *makespan* como en los casos anteriores.

Para el estudio de estos problemas se han aprovechado las instancias "*n0"* de la PSPLIB, las cuales son del MRCPSP con 2 tipos de recursos renovables, entre 10 y 20 trabajos reales, 3 modos de trabajo por cada uno de ellos, y en cada modo se utiliza un solo tipo de recurso de forma que si el número de recursos necesario crece, el tiempo de proceso decrece.

A los ejemplos planteados en "*n0"* se le añaden los siguientes datos para conformar el problema deseado:

- El Tiempo de Ciclo es un **25%** mayor al mínimo *makespan* proporcionado en la PSPLIB (redondeado al entero superior).
- Se añadirán **dos zonas de trabajo** en la estación.
- Las tareas con número **par** se ejecutarán en la **zona uno**, mientras que las **impares** en la **zona dos**.
- La capacidad máxima en las zonas de trabajo viene determinada por la **media del número máximo de trabajadores** disponibles de cada tipo (redondeado al entero superior).
- Cada problema se resolverá con **dos funciones objetivos**:
  - Minimizar el makespan
  - Minimizar el coste total de los trabajadores en la estación durante todo el tiempo de ciclo.

A esta batería de problemas MRCPSP con restricciones de Zona se les denominará **MRCPSP-Z**.

El orden de la experimentación de los MRCPSP-Z se produce del mismo modo que anteriormente, es decir, en función del número de variables binarias (NVB), el cual se calcula de la siguiente forma:

$$NVB = (NT \times NM \times TiempoCiclo) - (4 \times TiempoCiclo)$$

Donde *NT* es el número de trabajos (=12, 14,…, 22), *NM* es el número de modos (=3) y *TiempoCiclo* es el valor calculado del Tiempo de Ciclo (25% superior al *Makespan*). El miembro de la derecha se debe a que hay variables binarias fijadas en los modelos.

En la *Ilustración 13* se puede ver el orden de los primeros 20 experimentos basándose en el número de variables binarias (NVB) de cada uno, así como el *makespan* proporcionado por la PSPLIB. Se muestra el valor del Tiempo de ciclo que se propone. También se muestra el número de trabajos que compone cada instancia y el *CPU-Time* del método antes descrito para el MRCPSP (sin zonas ni tiempo de ciclo).

ORDEN	Parameter	Instance	Makespan	Tiempo de ciclo	CPU-Time[sec.]	Nº Trabajos	Nº Var Bin
1	3	9	9	12	0,03	12	384
2	7	8	9	12	0,03	12	384
3	4	7	10	13	0	12	416
4	15	3	9	12	0	14	456
5	3	1	11	14	0	12	448
6	8	3	11	14	0	12	448
7	4	3	11	14	0,04	12	448
8	12	8	10	13	0,03	14	494
9	3	4	12	15	0	12	480
10	4	8	12	15	0	12	480
11	4	9	12	15	0,03	12	480
12	8	9	12	15	0,03	12	480
13	6	8	12	15	0,04	12	480
14	2	3	13	17	0	12	544
15	3	2	13	17	0	12	544
16	3	6	13	17	0	12	544
17	8	5	13	17	0	12	544
18	4	2	13	17	0,03	12	544
19	6	3	14	18	0,03	12	576
20	12	2	11	14	0,03	14	532

Ilustración 13: Ejemplo de tabla de datos de problemas MRCPSP-Z con restricciones de zonas

5.3 Resolución de los problemas

5.3.1 Modelo en LINGO del problema RCPSP

Para hallar la solución de cada una de las instancias propuestas en los experimentos (1), (2) y (3), se ha optado por utilizar el programa de optimización *Lingo 18.0 x64.*

En los experimentos de RCPSP (1) y MRCPSP (2) se han programado en Lingo los modelos matemáticos estándares que se expusieron en capítulos anteriores. Lo único que habría que añadirle serían los datos que PSPLIB facilita, y usando *makespan* como horizonte de periodos en cada caso, y variando el número de trabajos en los MRCPSP.

En primer lugar, se muestra el modelo desarrollado para Lingo en los problemas RCPSP y que se formuló en el apartado 3.1.

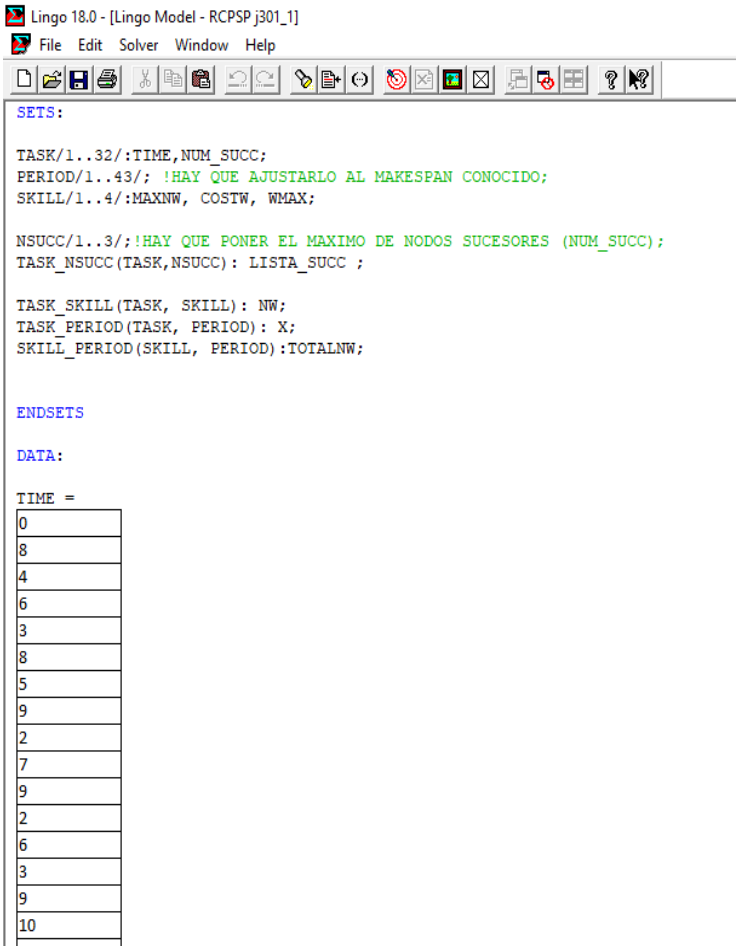


Ilustración 14: Entrada de datos en el experimento RCPSP j301_1

En la *Ilustración 14* se puede comprobar el inicio de la programación de un ejemplo RCPSP en Lingo. En primer lugar, se indica el número de trabajos existentes en el conjunto *"TASK",* el *makespan* conocido por la PSPLIB en el conjunto *"PERIOD"* y el número de tipos de recursos que hay en el conjunto *"SKILL",* así como el máximo de nodos sucesores en el conjunto "*NSUCC*", que siempre será 3. Con estos 4 conjuntos, se definen los conjuntos combinados "*TASK_NSUCC*", "*TASK_SKILL*", "*TASK_PERIOD*" y "*SKILL_PERIOD*". Dentro de cada conjunto se definen los datos y las variables como atributos de los mismos.

A continuación, se trasladan los datos de PSPLIB del experimento al programa como se puede observar en las *Ilustraciones 14, 15, 16, 17 y 18*: tiempos de ejecución de cada actividad (*TIME*), número de sucesores de cada actividad (*NUM_SUCC*) y los sucesores de cada una de ellas (*LISTA_SUCC*), el número de recursos necesarios de cada tipo para cada actividad (*NW*) y el máximo de cada tipo (*MAXNW*), y finalmente los costes de cada tipo de recurso (*COSTW*) aunque no serán usados.

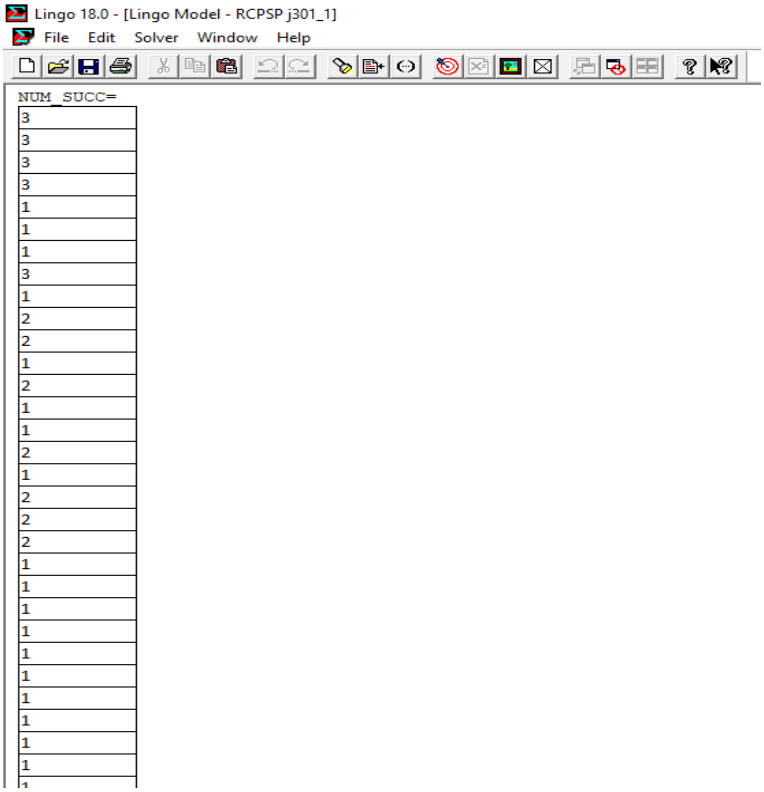


Ilustración 15: Número de sucesores y sucesores de cada trabajo en el experimento RCPSP j301_1

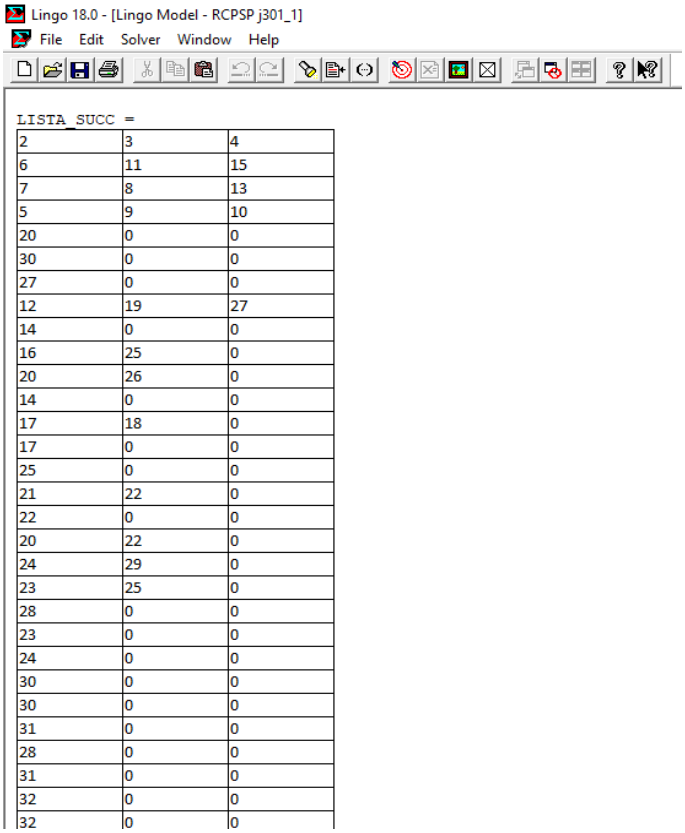


Ilustración 16: Número de sucesores y sucesores de cada trabajo en el experimento RCPSP j301_1

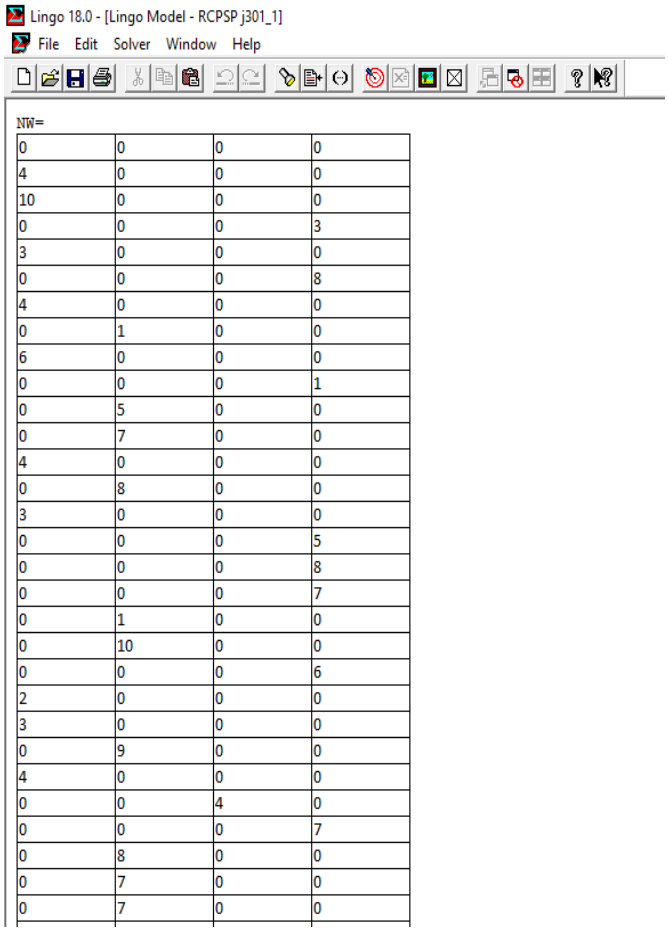


Ilustración 17: Necesidad de cada tipo de recurso en cada trabajo en el experimento RCPSP j301_1

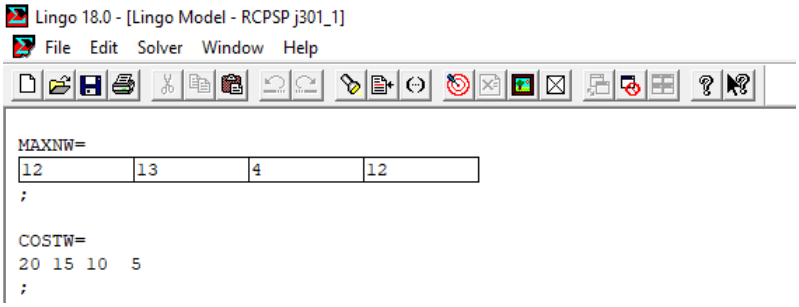


Ilustración 18: Número máximo de cada tipo de recurso en el experimento RCPSP j301_1

En la *Ilustración 19* se muestra cómo se guardan los resultados que se obtengan en un documento "*.txt*".

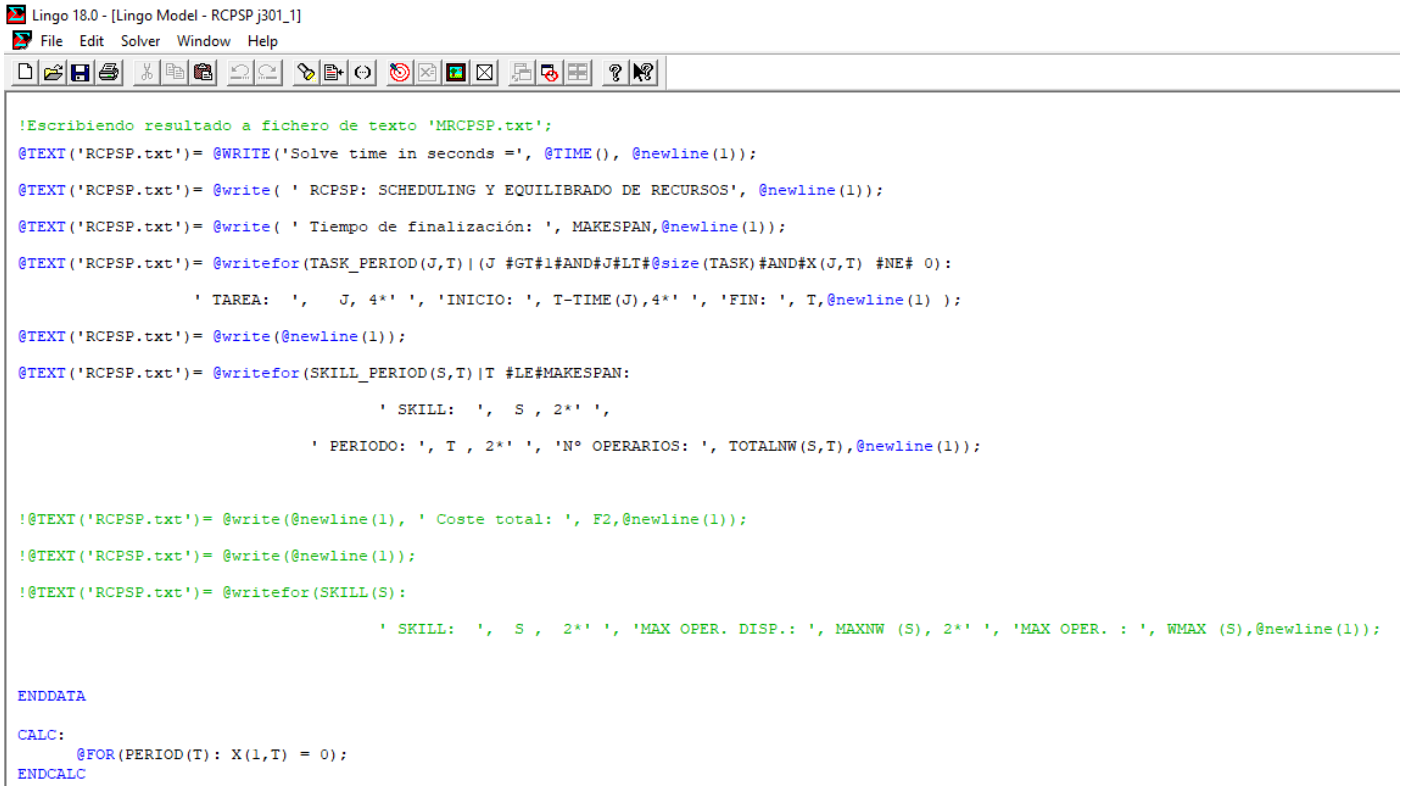


Ilustración 19: Programación para obtener un documento .txt con los resultados del ejemplo j301_1

En la *Ilustración 20* se exponen cómo se programan las diferentes restricciones de las que está compuesto el problema RCPSP y que se explicaron en el apartado 3.1.

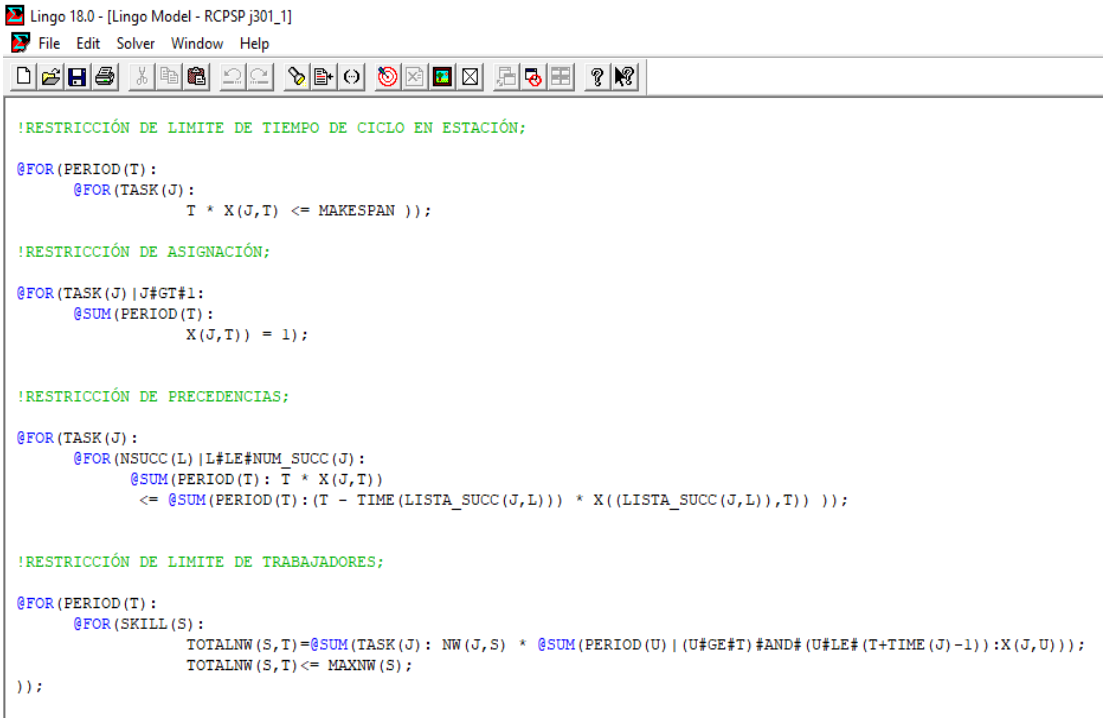


Ilustración 20: Programación de restricciones del problema RCPSP j301_1

En la *Ilustración 21* se puede observar la programación de la función objetivo que minimiza el *makespan*, así como la definición de las variables binarias.

También se puede ver que aparece una segunda función objetivo relacionada con costes pero, en este caso, no está disponible porque no se ha considerado para los experimentos (1).

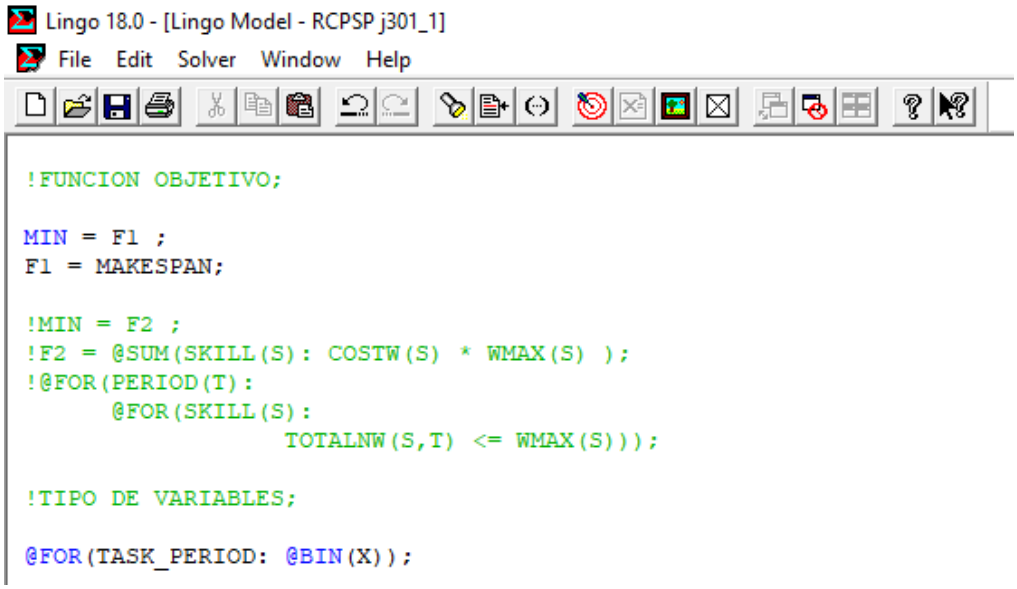


Ilustración 21: Programación de función objetivo y tipo de variables del problema RCPSP j301_1

En la *Ilustración 22*, se muestra la solución obtenida para una instancia del problema RCPSP. Se muestra el CPU-Time que ha tardado Lingo en dar la solución óptima, el menor tiempo de finalización en la estación (*makespan*), el inicio y fin de cada una de las actividades, así como el número de operarios necesario de cada especialidad en cada instante de tiempo. Se observa que el *makespan* obtenido coincide con el conocido en la literatura y usado como horizonte (43).

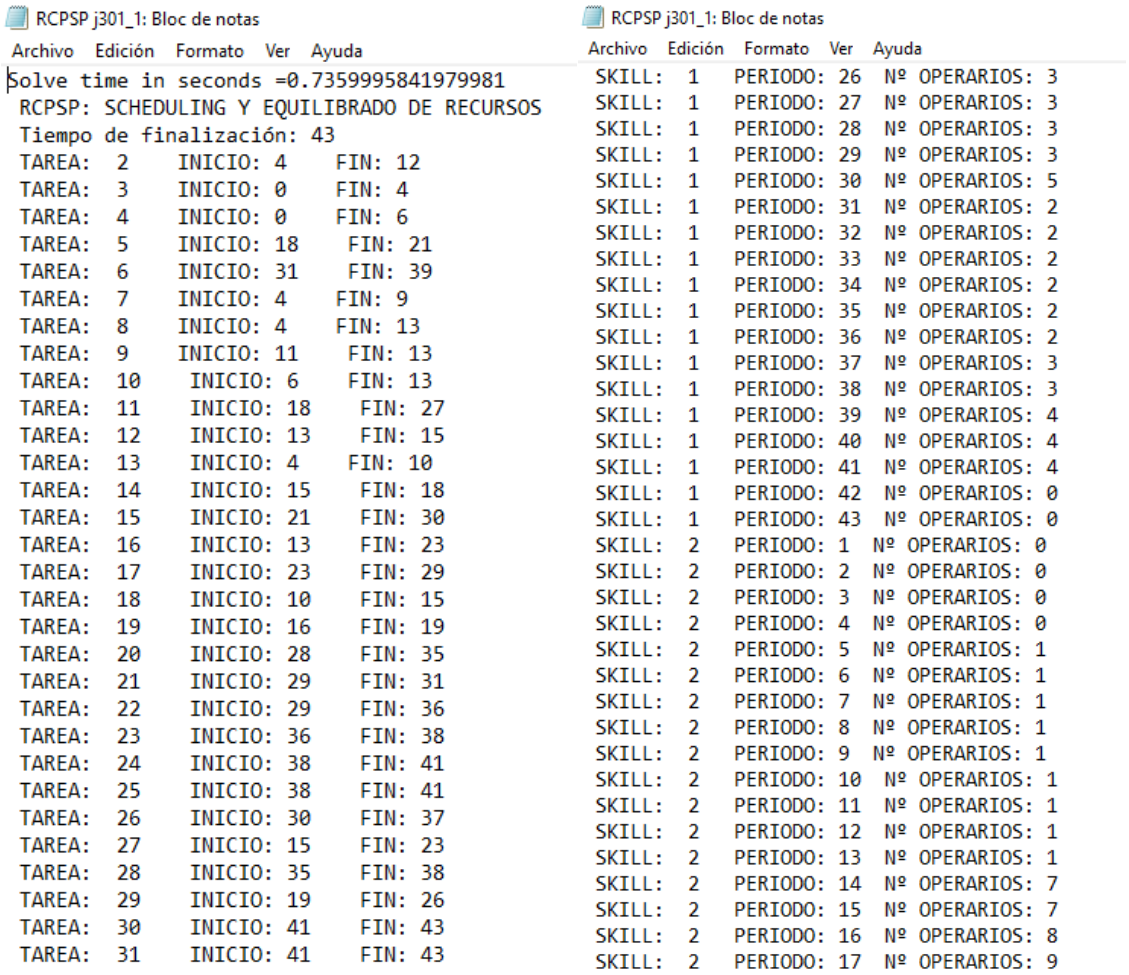


Ilustración 22: Solución del problema de RCPSP j301_1

5.3.2 Modelo en LINGO del problema MRCPSP

En segundo lugar, se muestra el modelo desarrollado para Lingo en los problemas MRCPSP y que se formuló en el apartado 3.2.

Los modelos de los problemas MRCPSP en Lingo, se desarrollan del mismo modo explicado para el RCPSP, con la única salvedad de que, en ellos, también hay que incluir el dato del modo de trabajo en el cual se va a trabajar.

En la *Ilustración 23* se puede ver que se añade un nuevo conjunto "*MODE*" asociado a los modos de trabajo. Se añaden los conjuntos combinados "*TASK_MODE*", "*TASK_MODE_SKILL*" y "*TASK_MODE_PERIOD*".

Respecto al modelo anterior, hay que introducir en primer lugar el número de modos de trabajo de cada una de las tareas (*NUM_MODE*) y los tiempos de ejecución de esas tareas para cada modo (*TIME*).

Lingo 18.0 - [Lingo Model - MRCPSP n01_2]

File Edit Solver Window Help

```
SETS:

TASK/1..12/:NUM_MODE,NUM_SUCC;!HAY QUE PONER EL NUMERO DE TAREAS;
PERIOD/1..15/;!HAY QUE AJUSTARLO AL MAKESPAN CONOCIDO;
SKILL/1..2/:MAXNW, COSTW, WMAX;
MODE/1..3/;

NSUCC/1..3/;!HAY QUE PONER EL MAXIMO DE NODOS SUCESORES (NUM_SUCC);
TASK_NSUCC(TASK,NSUCC): LISTA_SUCC ;

TASK_MODE(TASK, MODE): TIME;
TASK_MODE_SKILL(TASK, MODE, SKILL): NW;

TASK_MODE_PERIOD(TASK, MODE, PERIOD): X;

SKILL_PERIOD(SKILL, PERIOD):TOTALNW;


ENDSETS

DATA:

NUM_MODE =
```

1
3
3
3
3
3
3
3
3
3
1

Lingo 18.0 - [Lingo Model - MRCPSP n01_2]

File Edit Solver Window Help

TIME =

0
0
0
1
2
3
3
2
3
1
2
3
1
2
3
2
2
3
6
2
3
1
2
3
1
2
3
1
2
3

Ilustración 23: Entrada de datos en el experimento MRCPSP n01_2

En la *Ilustración 24* se terminan de insertar los datos restantes en el modelo: número de sucesores (*NUM_SUCC*), lista de tareas sucesoras (*LISTA_SUCC*), recursos necesarios para cada tarea y modo (NW) y el máximo disponible de cada tipo de recurso (*MAXNW*).

Lingo 18.0 - [Lingo Model - MRCPSP n01_2]

File Edit Solver Window Help

NUM_SUCC=

3
3
2
3
2
2
2
2
1
1
1
0

;

LISTA_SUCC =

2	3	4
5	7	9
8	9	0
5	6	7
10	11	0
8	9	0
10	11	0
10	11	0
12	0	0
12	0	0
12	0	0
0	0	0

;

Lingo 18.0 - [Lingo Model - MRCPSP n01_2]

File Edit Solver Window Help

NW=

0	0
0	0
0	0
0	10
6	6
9	0
4	0
5	0
7	0
7	0
5	0
8	5
0	5
2	0
6	2
0	3
3	8
6	7
6	0
8	4
9	3
10	0
2	8
3	0
7	0
4	6
4	0
6	0
1	0
3	0

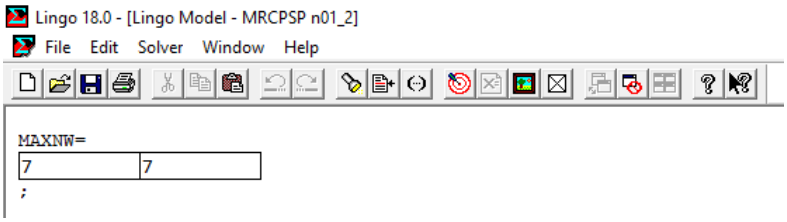


Ilustración 24: Inserción de datos para el problema MRCPSP n01_2

La *Ilustración 25* muestra la programación necesaria en Lingo para crear un documento "*.txt*" para guardar las soluciones del problema planteado.

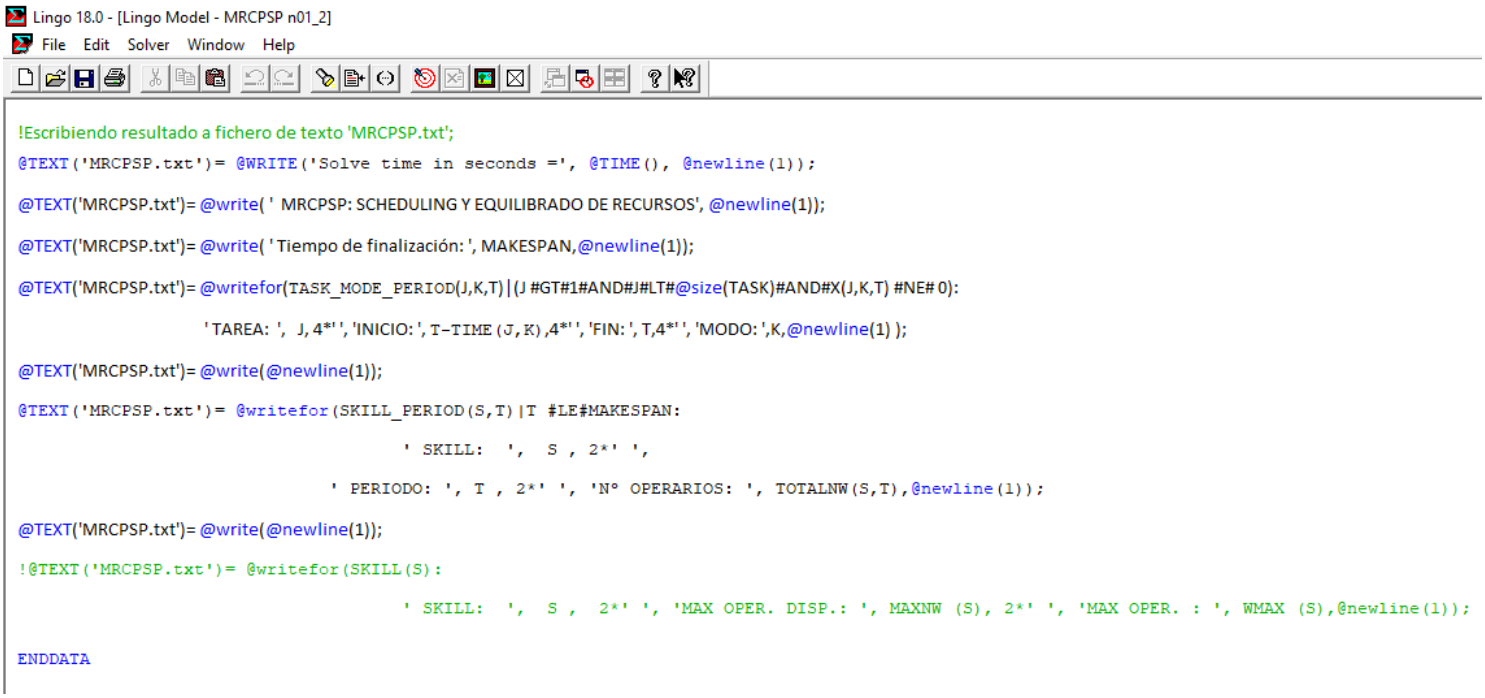
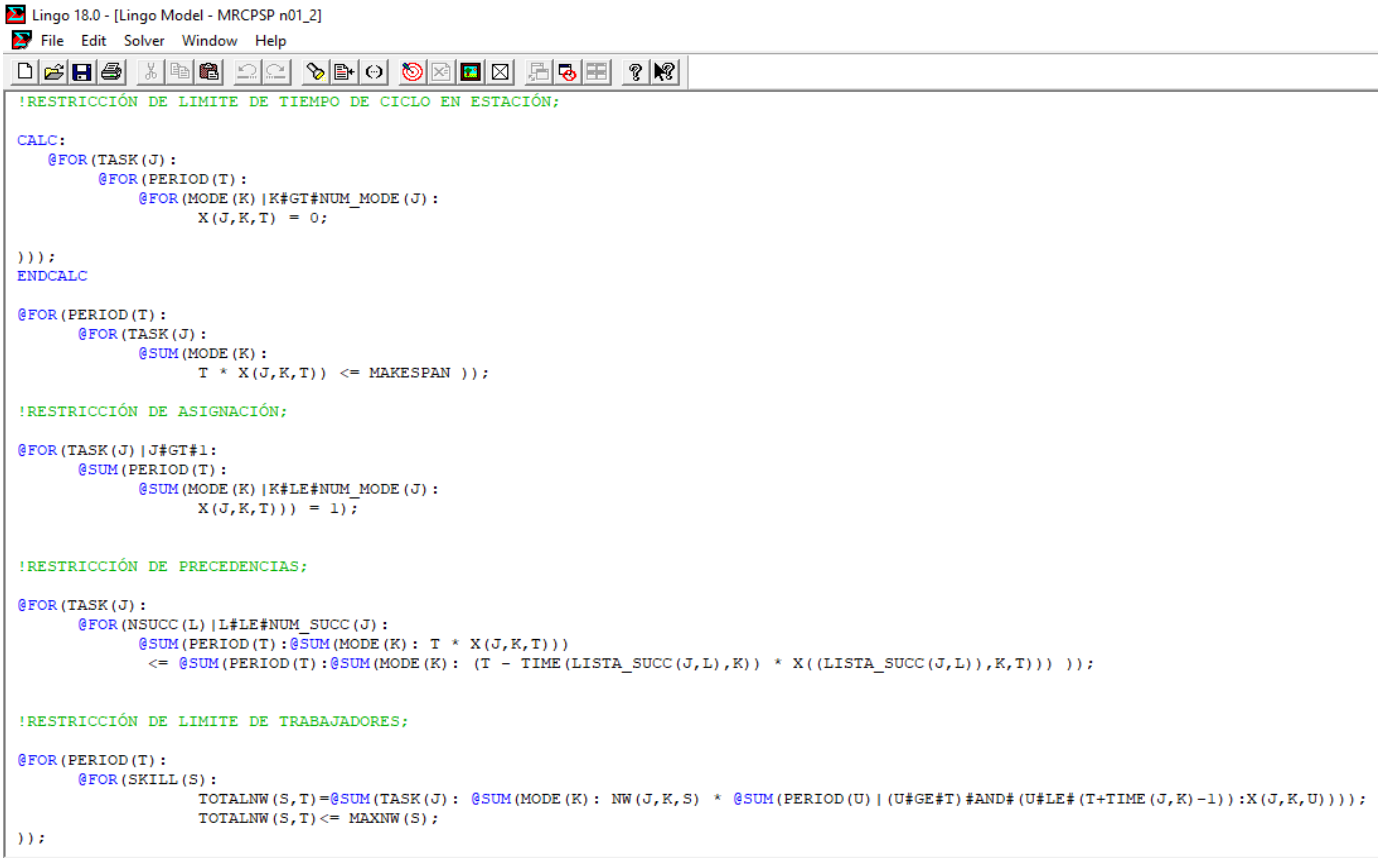


Ilustración 25: Programación para obtener un documento .txt con los resultados del ejemplo MRCPSP n01_2

La *Ilustración 26* muestra cómo se programan las diferentes restricciones de las que está compuesto el problema MRCPSP y que se explicaron en el apartado 3.2.

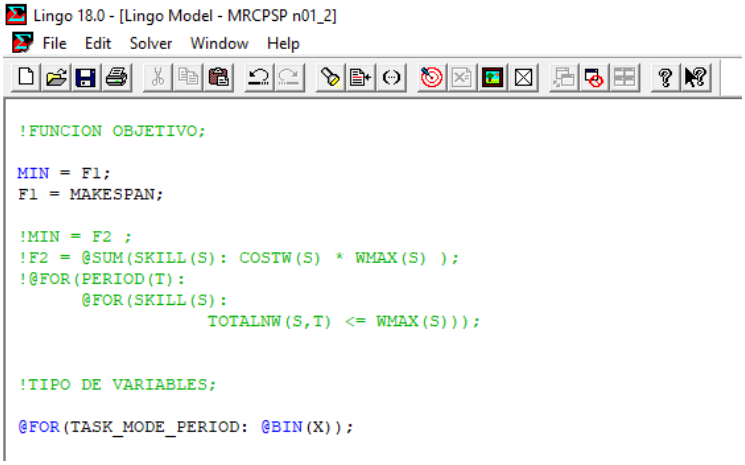



```
!RESTRICCIÓN DE LIMITE DE TIEMPO DE CICLO EN ESTACIÓN;

CALC:
   @FOR(TASK(J):
        @FOR(PERIOD(T):
             @FOR(MODE(K)|K#GT#NUM_MODE(J):
                  X(J,K,T) = 0;

)));
ENDCALC

@FOR(PERIOD(T):
      @FOR(TASK(J):
            @SUM(MODE(K):
                  T * X(J,K,T)) <= MAKESPAN ));

!RESTRICCIÓN DE ASIGNACIÓN;

@FOR(TASK(J)|J#GT#1:
      @SUM(PERIOD(T):
            @SUM(MODE(K)|K#LE#NUM_MODE(J):
                  X(J,K,T))) = 1);


!RESTRICCIÓN DE PRECEDENCIAS;

@FOR(TASK(J):
      @FOR(NSUCC(L)|L#LE#NUM_SUCC(J):
            @SUM(PERIOD(T):@SUM(MODE(K): T * X(J,K,T)))
             <= @SUM(PERIOD(T):@SUM(MODE(K): (T - TIME(LISTA_SUCC(J,L),K)) * X((LISTA_SUCC(J,L)),K,T))) ));


!RESTRICCIÓN DE LIMITE DE TRABAJADORES;

@FOR(PERIOD(T):
      @FOR(SKILL(S):
            TOTALNW(S,T)=@SUM(TASK(J): @SUM(MODE(K): NW(J,K,S) * @SUM(PERIOD(U)|(U#GE#T)#AND#(U#LE#(T+TIME(J,K)-1)):X(J,K,U))));
            TOTALNW(S,T)<= MAXNW(S);
));
```

Ilustración 26: Programación de restricciones del problema MRCPSP

En la *Ilustración 27* se programa la función objetivo del problema, así como la definición de las variables binarias.

En este caso, de nuevo, hay una segunda función objetivo relacionada con los costes pero no se trabaja con ella, y está deshabilitada porque no es uno de los objetivos de este trabajo en los problemas MRCPSP.




```
!FUNCION OBJETIVO;

MIN = F1;
F1 = MAKESPAN;

!MIN = F2 ;
!F2 = @SUM(SKILL(S): COSTW(S) * WMAX(S) );
!@FOR(PERIOD(T):
      @FOR(SKILL(S):
                  TOTALNW(S,T) <= WMAX(S)));


!TIPO DE VARIABLES;

@FOR(TASK_MODE_PERIOD: @BIN(X));
```

Ilustración 27: Programación de la función objetivo y del tipo de variable en el problema MRCPSP en Lingo

En la *Ilustración 28* se puede observar la solución del experimento *MRCPSP n01_2*. En ella, aparece el CPU-Time que tarda Lingo en proporcionar la solución óptima del problema y el

makespan del mismo. Además, se obtiene el instante de inicio y fin de cada tarea junto con el modo de trabajo seleccionado. Por último, se muestra para cada tipo de operario y en cada momento la cantidad de ellos que se necesita.

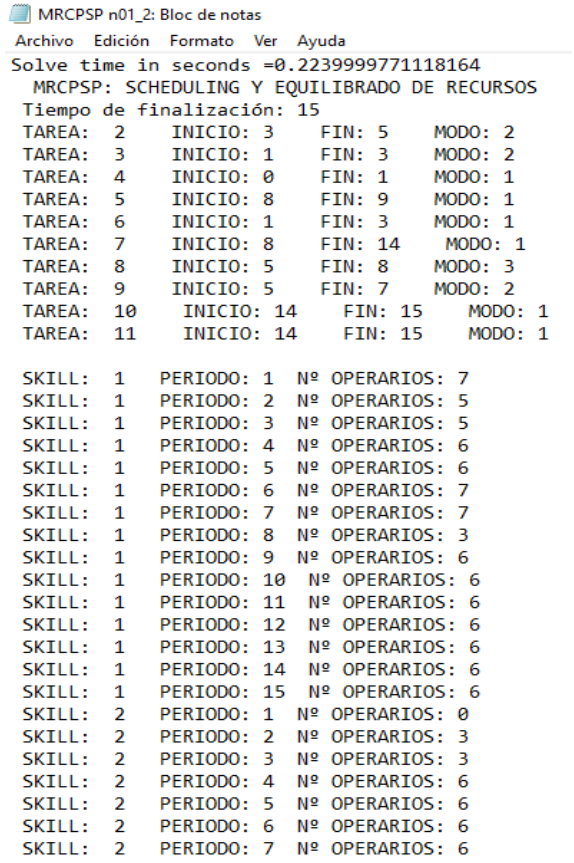


Ilustración 28: Solución del problema MRCPSP n01_2

5.3.3 Modelo en LINGO del problema MRCPSP-Z

Estos experimentos explicados se corresponden con los problemas estándares de la programación de proyectos con recursos limitados, tanto con un único modo de trabajo (RCPSP) como con varios modos de trabajo (MRCPSP).

A continuación, se va a exponer un tercer tipo de experimentos (3), cuyo modelo ya ha sido explicado en el apartado 4.3. Las instancias han sido generadas a partir de las mismas instancias del problema MRCPSP añadiéndole restricciones de zona. Este tipo de problema se produce en las estaciones de montaje aeronáuticas y les denominamos MRCPSP-Z.

En este caso, se trabaja con dos funciones objetivos: minimizar el makespan; y minimizar los costes totales de los trabajadores. Los datos de entrada son los mismos para ambas funciones objetivos siendo la única diferencia habilitar en Lingo la función que se desee obtener. Se tendrán que elaborar dos documentos de soluciones diferentes ya que en uno se trata de minimizar el makespan y calcular los costes (sin minimizarlos), y otro documento en el que se realiza lo contrario donde los datos de entrada son los mismos.

En la *Ilustración 29* se puede ver que se añade un nuevo conjunto "*ZONE*" asociado a las 2 zonas de trabajo. Se añaden los conjuntos combinados "*TASK_ZONE*" y "*ZONE_PERIOD*".

Respecto al modelo anterior, hay que introducir en primer lugar la capacidad máxima de trabajadores en cada zona de trabajo (*MAXNZ*) y en qué zona se realiza cada tarea (*NZ*).

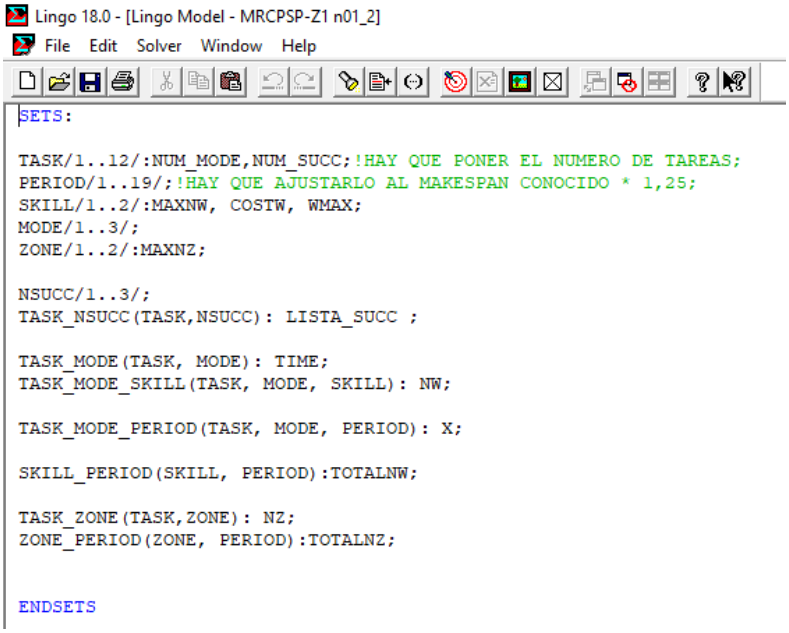


Ilustración 29: Entrada de datos y variables para la programación del MRCPSP-Z n01_2

En las siguientes *Ilustraciones 30 y 31*, se introducen los datos de los modos de trabajo de cada tarea, los tiempos de ejecución de cada tarea en cada modo, el número de sucesores y las tareas sucesoras de cada tarea, los recursos necesarios en cada tarea y modo, y el número de recursos disponibles de cada tipo.

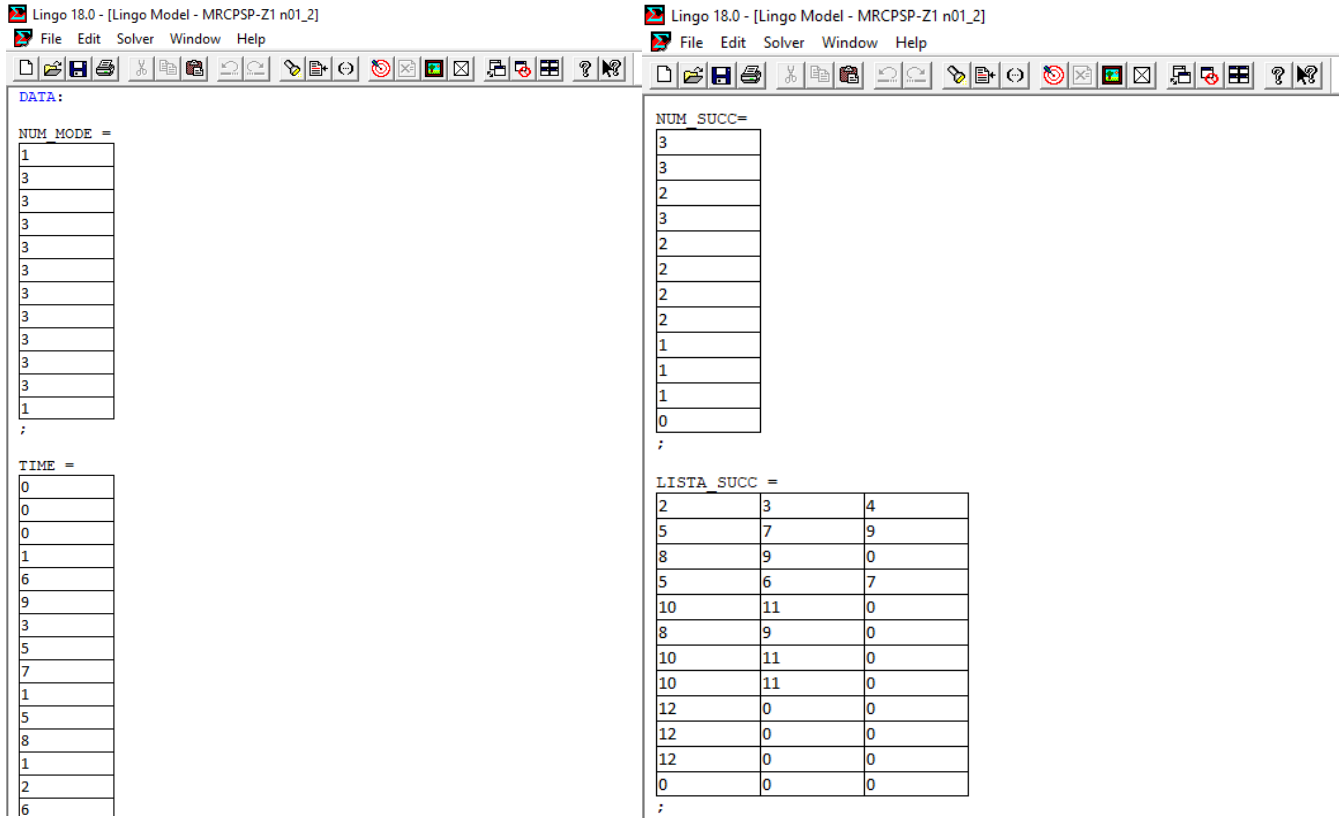


Ilustración 30: Datos del problema de MRCPSP-Z n01_2

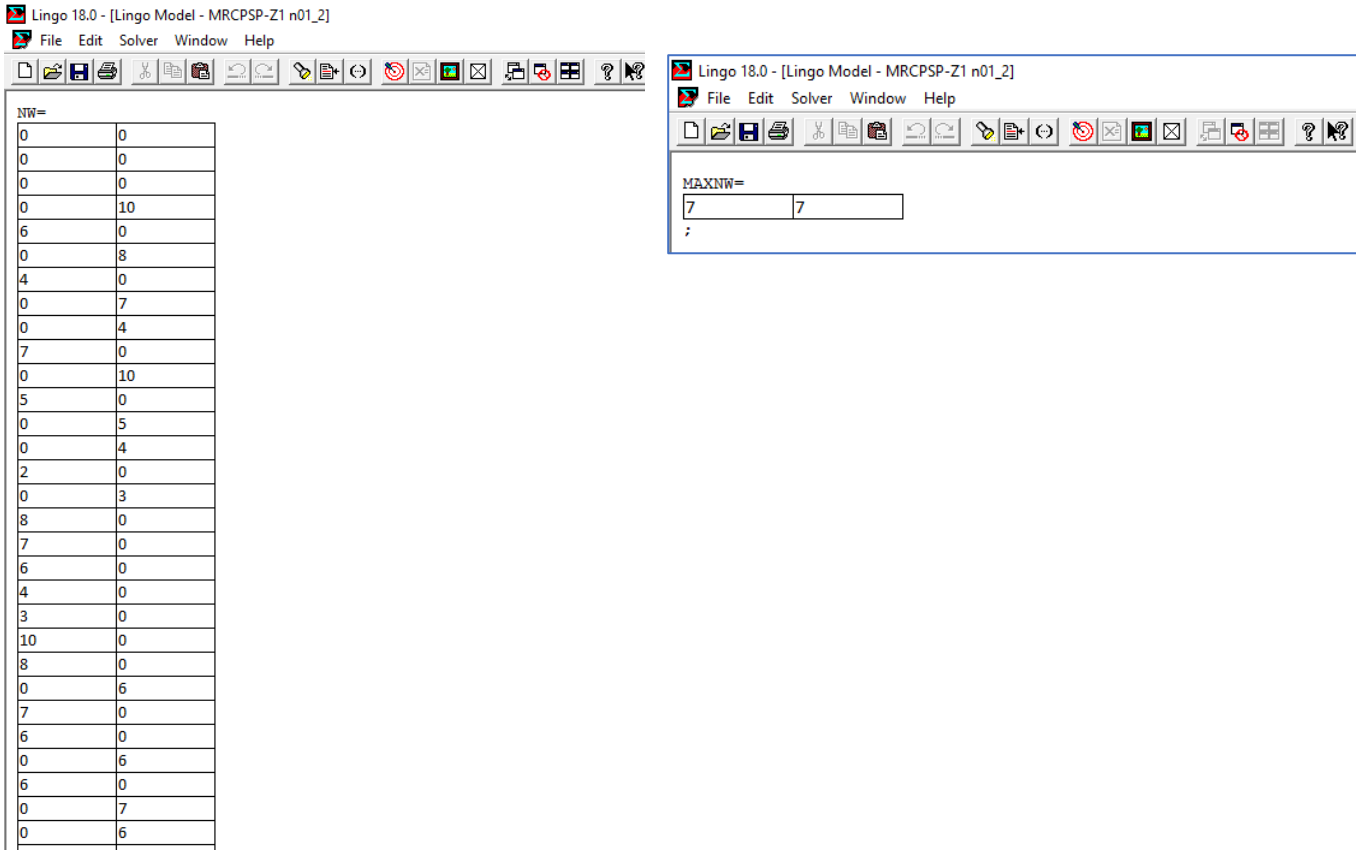


Ilustración 31: Datos del problema MRCPSP-Z n01_2

La *Ilustración 32* muestra los nuevos datos necesarios para este tipo de problemas: *MAXNZ* indica el número de trabajadores máximo que pueden entrar en cada zona y los *NZ* indican en qué zona se realiza cada tarea, siendo un 1 aquellas que se realizan en la zona y un 0 si no. Las tareas ficiticas no se asignan a ninguna zona. Se observa que las tareas impares se asignan a la zona 1 y las pares a la zona 2 por convenio.

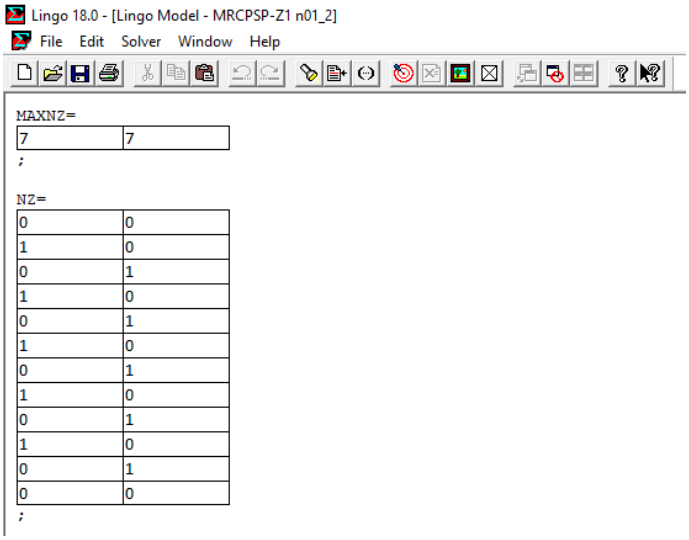


Ilustración 32: Datos de las zonas de trabajo en el problema MRCPSP-Z n01_2

Como en los experimentos anteriores, en este también se dedica una parte de la programación en Lingo a crear un fichero para guardar la solución del problema (*Ilustración 33*).

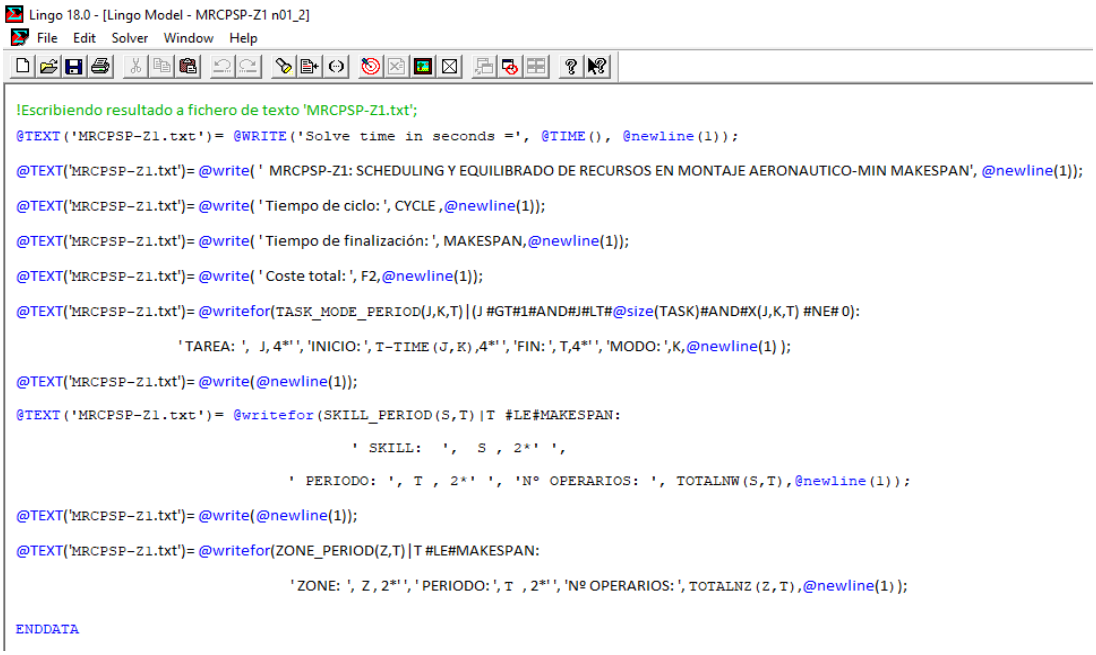


Ilustración 33: Programación para obtener un documento .txt con los resultados del ejemplo MRCPSP-Z n01_2

En las *Ilustraciones 34 y 35* se detallan todas las restricciones que limitan el problema MRCPSP-Z que se explicaron con anterioridad en el apartado 4.3.

Lingo 18.0 - [Lingo Model - MRCPSP-Z1 n01_2]
File Edit Solver Window Help

```
!RESTRICCIÓN DE LIMITE DE TIEMPO DE CICLO EN ESTACIÓN;

CALC:
   @FOR(TASK(J):
        @FOR(PERIOD(T):
             @FOR(MODE(K)|K#GT#NUM_MODE(J):
                  X(J,K,T) = 0;

)));
CYCLE=@SIZE(PERIOD);
ENDCALC

@FOR(PERIOD(T):
      @FOR(TASK(J):
            @SUM(MODE(K):
                 T * X(J,K,T)) <= MAKESPAN ));
MAKESPAN <= CYCLE;

!RESTRICCIÓN DE ASIGNACIÓN;

@FOR(TASK(J)|J#GT#1:
      @SUM(PERIOD(T):
            @SUM(MODE(K)|K#LE#NUM_MODE(J):
                 X(J,K,T))) = 1);


!RESTRICCIÓN DE PRECEDENCIAS;

@FOR(TASK(J):
      @FOR(NSUCC(L)|L#LE#NUM_SUCC(J):
            @SUM(PERIOD(T):@SUM(MODE(K): T * X(J,K,T)))
             <= @SUM(PERIOD(T):@SUM(MODE(K): (T - TIME(LISTA_SUCC(J,L),K)) * X((LISTA_SUCC(J,L)),K,T))) ));
```

Ilustración 34: Restricciones del problema MRCPSP-Z

Lingo 18.0 - [Lingo Model - MRCPSP-Z1 n01_2]
File Edit Solver Window Help

```
!RESTRICCIÓN DE LIMITE DE TRABAJADORES;

@FOR(PERIOD(T):
     @FOR(SKILL(S):
          TOTALNW(S,T)=@SUM(TASK(J): @SUM(MODE(K): NW(J,K,S) * @SUM(PERIOD(U)|(U#GE#T)#AND#(U#LE#(T+TIME(J,K)-1)):X(J,K,U))));
          TOTALNW(S,T)<= MAXNW(S);
));

!RESTRICCIÓN DE LIMITE EN ZONAS;

@FOR(PERIOD(T):
     @FOR(ZONE(Z):
          TOTALNZ(Z,T)=@SUM(TASK(J): @SUM(SKILL(S): @SUM(MODE(K): NZ(J,Z) * NW(J,K,S)* @SUM(PERIOD(U)|(U#GE#T)#AND#(U#LE#(T+TIME(J,K)-1)):X(J,K,U))))); ));

@FOR(PERIOD(T):
     @FOR(ZONE(Z):
          TOTALNZ(Z,T) <= MAXNZ(Z)));
```

Ilustración 35: Restricciones adicionales del problema MRCPSP-Z

Para terminar, se programan las dos funciones objetivos, y se definen las variables binarias. Se observa que la dimensión de las mismas sigue siendo la del conjunto combinado "*TASK_MODE_PERIOD*". Son más el NVB para el MRCPSP-Z que para el MRCPSP porque la dimensión del conjunto "*PERIOD*" es un 25% mayor.

Respecto a las dos funciones objetivos, en la *Ilustración 36*, la funcion objetivo del *makespan* (F1) está habilitada, mientras que la relacionada con los costes de los trabajadores (F2) no lo está. Se obtendrá un valor de costes pero no estará minimizado. De forma inversa ocurre en la *Ilustración 37*. Se observa que la F2 tiene un término infinitesimal que multiplica el *makespan*, para que si existen soluciones alternativas de coste mínimo, elija la de menor *makespan*.

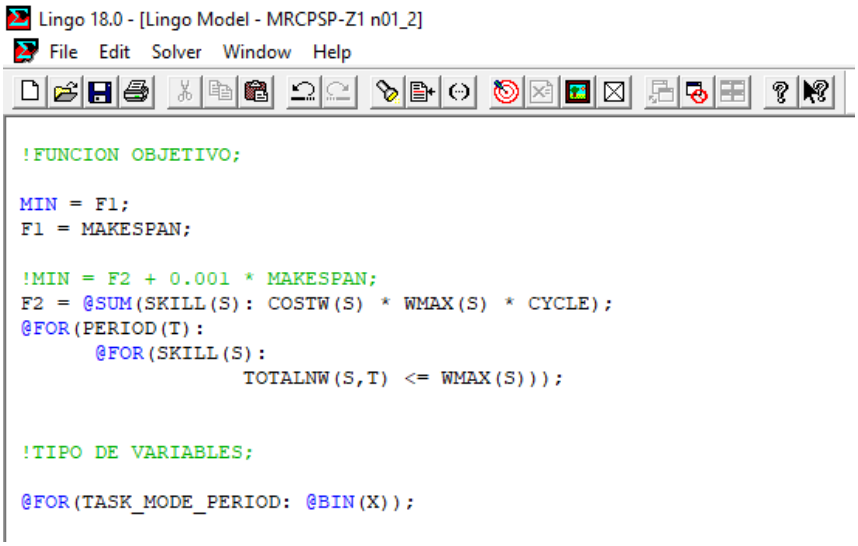


Ilustración 36: Programación de funciones objetivo minimizando el makespan del problema MRCPSP-Z

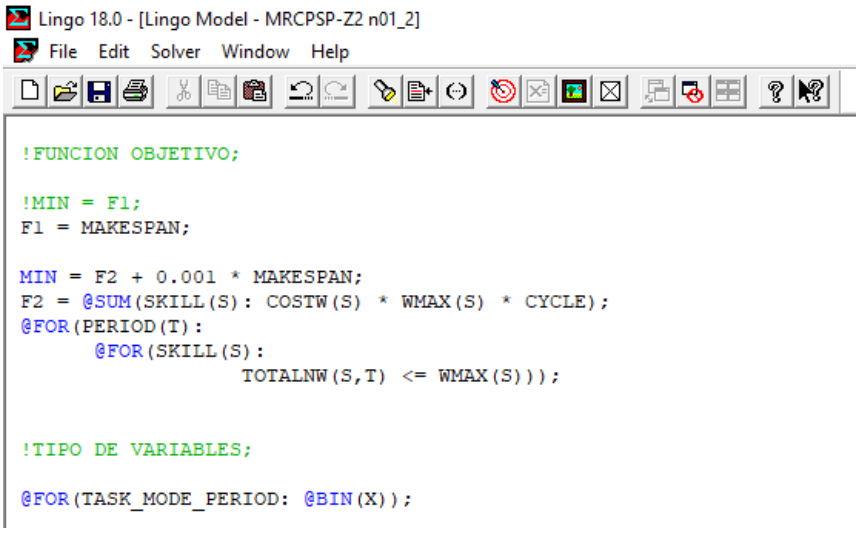


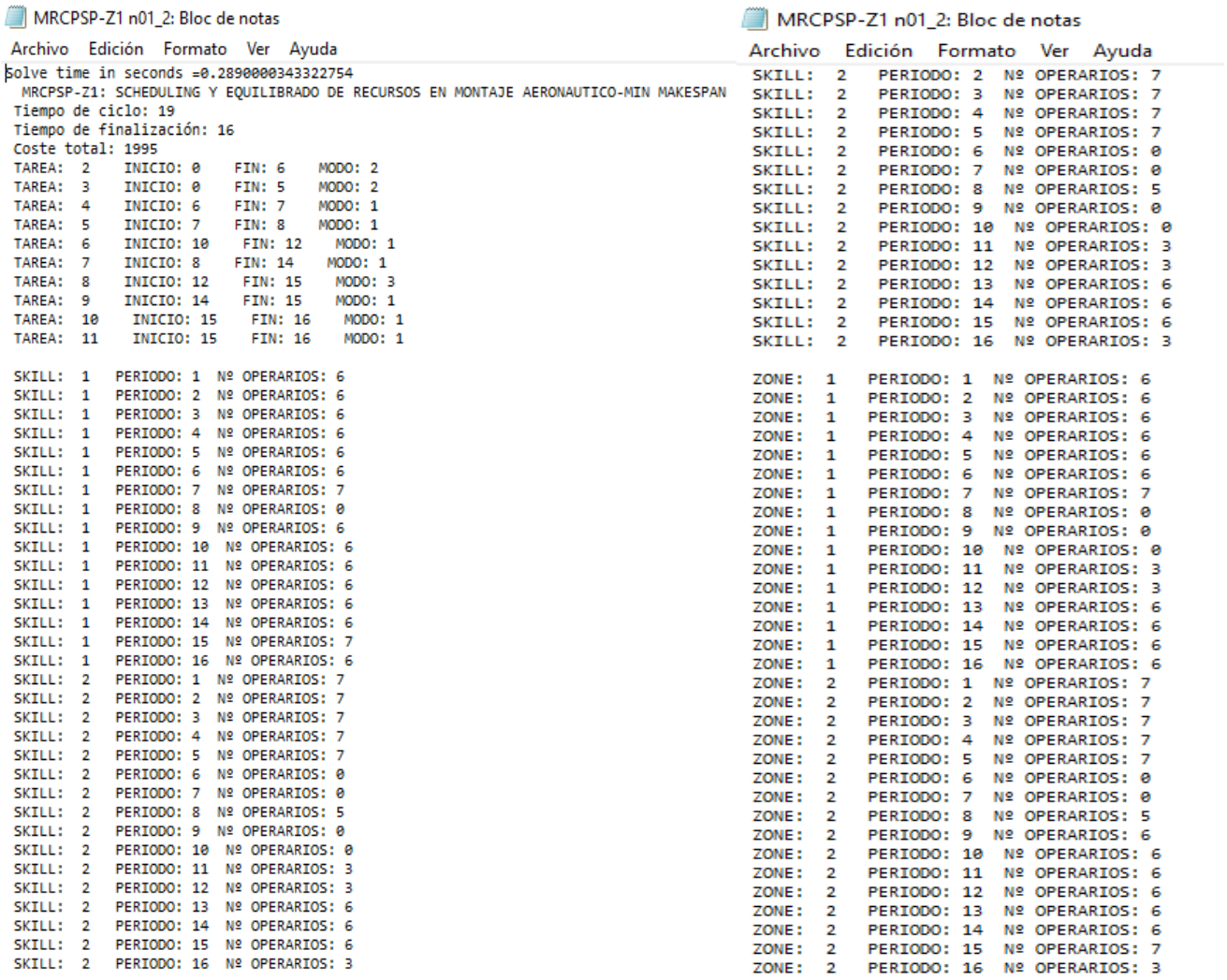
Ilustración 37: Programación de funciones objetivo minimizando los costes totales de los trabajadores del problema MRCPSP-Z

Como se anotó recientemente, este tipo de problemas crea dos soluciones. Una de ellas minimizando el *makespan* y otra minimizando los costes totales de los trabajadores.

En el primer caso, *Ilustración 38*, se ve el tiempo que ha necesitado Lingo para hallar la solución, el tiempo de ciclo asignado, el makespan óptimo y los costes totales (no minimizado). Igualmente, indica el momento de inicio y fin de cada actividad y el modo de

trabajo seleccionado; los operarios necesarios de cada especialidad en cada instante de tiempo; y los operarios necesarios en cada zona de trabajo en cada instante de tiempo.

De esta forma se obtiene los operarios necesarios de cada especialidad en cada momento y su asignación a cada una de las zonas.



MRCPSP-Z1 n01_2: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Solve time in seconds =0.2890000343322754
  MRCPSP-Z1: SCHEDULING Y EQUILIBRADO DE RECURSOS EN MONTAJE AERONAUTICO-MIN MAKESPAN
Tiempo de ciclo: 19
Tiempo de finalización: 16
Coste total: 1995
TAREA:  2    INICIO: 0    FIN: 6    MODO: 2
TAREA:  3    INICIO: 0    FIN: 5    MODO: 2
TAREA:  4    INICIO: 6    FIN: 7    MODO: 1
TAREA:  5    INICIO: 7    FIN: 8    MODO: 1
TAREA:  6    INICIO: 10    FIN: 12    MODO: 1
TAREA:  7    INICIO: 8    FIN: 14    MODO: 1
TAREA:  8    INICIO: 12    FIN: 15    MODO: 3
TAREA:  9    INICIO: 14    FIN: 15    MODO: 1
TAREA:  10    INICIO: 15    FIN: 16    MODO: 1
TAREA:  11    INICIO: 15    FIN: 16    MODO: 1

SKILL:  1   PERIODO: 1  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 2  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 3  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 4  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 5  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 6  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 7  Nº OPERARIOS: 7
SKILL:  1   PERIODO: 8  Nº OPERARIOS: 0
SKILL:  1   PERIODO: 9  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 10  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 11  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 12  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 13  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 14  Nº OPERARIOS: 6
SKILL:  1   PERIODO: 15  Nº OPERARIOS: 7
SKILL:  1   PERIODO: 16  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 1  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 2  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 3  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 4  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 5  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 6  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 7  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 8  Nº OPERARIOS: 5
SKILL:  2   PERIODO: 9  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 10  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 11  Nº OPERARIOS: 3
SKILL:  2   PERIODO: 12  Nº OPERARIOS: 3
SKILL:  2   PERIODO: 13  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 14  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 15  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 16  Nº OPERARIOS: 3
```

MRCPSP-Z1 n01_2: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
SKILL:  2   PERIODO: 2  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 3  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 4  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 5  Nº OPERARIOS: 7
SKILL:  2   PERIODO: 6  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 7  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 8  Nº OPERARIOS: 5
SKILL:  2   PERIODO: 9  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 10  Nº OPERARIOS: 0
SKILL:  2   PERIODO: 11  Nº OPERARIOS: 3
SKILL:  2   PERIODO: 12  Nº OPERARIOS: 3
SKILL:  2   PERIODO: 13  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 14  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 15  Nº OPERARIOS: 6
SKILL:  2   PERIODO: 16  Nº OPERARIOS: 3

ZONE:  1   PERIODO: 1  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 2  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 3  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 4  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 5  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 6  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 7  Nº OPERARIOS: 7
ZONE:  1   PERIODO: 8  Nº OPERARIOS: 0
ZONE:  1   PERIODO: 9  Nº OPERARIOS: 0
ZONE:  1   PERIODO: 10  Nº OPERARIOS: 0
ZONE:  1   PERIODO: 11  Nº OPERARIOS: 3
ZONE:  1   PERIODO: 12  Nº OPERARIOS: 3
ZONE:  1   PERIODO: 13  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 14  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 15  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 16  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 1  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 2  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 3  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 4  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 5  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 6  Nº OPERARIOS: 0
ZONE:  2   PERIODO: 7  Nº OPERARIOS: 0
ZONE:  2   PERIODO: 8  Nº OPERARIOS: 5
ZONE:  2   PERIODO: 9  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 10  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 11  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 12  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 13  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 14  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 15  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 16  Nº OPERARIOS: 3
```

Ilustración 38: Solución con makespan minimizado del experimento MRCPSP-Z n01_2

Del mismo modo que se ha minimizado el makespan mostrado en la ilustración anterior, en las *Ilustración 39*, se detalla exactamente lo mismo pero minimizando los costes totales de los trabajadores y no el makespan.

MRCPSP-Z2 n01_2: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Solve time in seconds =0.5399999618530273
  MRCPSP-Z2: SCHEDULING Y EQUILIBRADO DE RECURSOS EN MONTAJE AERONAUTICO-MIN COSTE
 Tiempo de ciclo: 19
 Tiempo de finalización: 16
 Coste total: 1900
 TAREA:  2    INICIO: 0    FIN: 6    MODO: 2
 TAREA:  3    INICIO: 0    FIN: 7    MODO: 3
 TAREA:  4    INICIO: 6    FIN: 7    MODO: 1
 TAREA:  5    INICIO: 7    FIN: 8    MODO: 1
 TAREA:  6    INICIO: 8    FIN: 10    MODO: 1
 TAREA:  7    INICIO: 8    FIN: 14    MODO: 1
 TAREA:  8    INICIO: 10    FIN: 13    MODO: 3
 TAREA:  9    INICIO: 14    FIN: 15    MODO: 1
 TAREA:  10    INICIO: 15    FIN: 16    MODO: 1
 TAREA:  11    INICIO: 15    FIN: 16    MODO: 1

 SKILL:  1   PERIODO: 1  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 2  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 3  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 4  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 5  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 6  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 7  Nº OPERARIOS: 7
 SKILL:  1   PERIODO: 8  Nº OPERARIOS: 0
 SKILL:  1   PERIODO: 9  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 10  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 11  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 12  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 13  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 14  Nº OPERARIOS: 6
 SKILL:  1   PERIODO: 15  Nº OPERARIOS: 7
 SKILL:  1   PERIODO: 16  Nº OPERARIOS: 6
 SKILL:  2   PERIODO: 1  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 2  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 3  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 4  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 5  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 6  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 7  Nº OPERARIOS: 4
 SKILL:  2   PERIODO: 8  Nº OPERARIOS: 5
 SKILL:  2   PERIODO: 9  Nº OPERARIOS: 3
 SKILL:  2   PERIODO: 10  Nº OPERARIOS: 3
 SKILL:  2   PERIODO: 11  Nº OPERARIOS: 6
 SKILL:  2   PERIODO: 12  Nº OPERARIOS: 6
 SKILL:  2   PERIODO: 13  Nº OPERARIOS: 6
 SKILL:  2   PERIODO: 14  Nº OPERARIOS: 0
```

*MRCPSP-Z2 n01_2: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
ZONE:  1   PERIODO: 1  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 2  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 3  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 4  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 5  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 6  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 7  Nº OPERARIOS: 7
ZONE:  1   PERIODO: 8  Nº OPERARIOS: 0
ZONE:  1   PERIODO: 9  Nº OPERARIOS: 3
ZONE:  1   PERIODO: 10  Nº OPERARIOS: 3
ZONE:  1   PERIODO: 11  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 12  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 13  Nº OPERARIOS: 6
ZONE:  1   PERIODO: 14  Nº OPERARIOS: 0
ZONE:  1   PERIODO: 15  Nº OPERARIOS: 0
ZONE:  1   PERIODO: 16  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 1  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 2  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 3  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 4  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 5  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 6  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 7  Nº OPERARIOS: 4
ZONE:  2   PERIODO: 8  Nº OPERARIOS: 5
ZONE:  2   PERIODO: 9  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 10  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 11  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 12  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 13  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 14  Nº OPERARIOS: 6
ZONE:  2   PERIODO: 15  Nº OPERARIOS: 7
ZONE:  2   PERIODO: 16  Nº OPERARIOS: 3
```

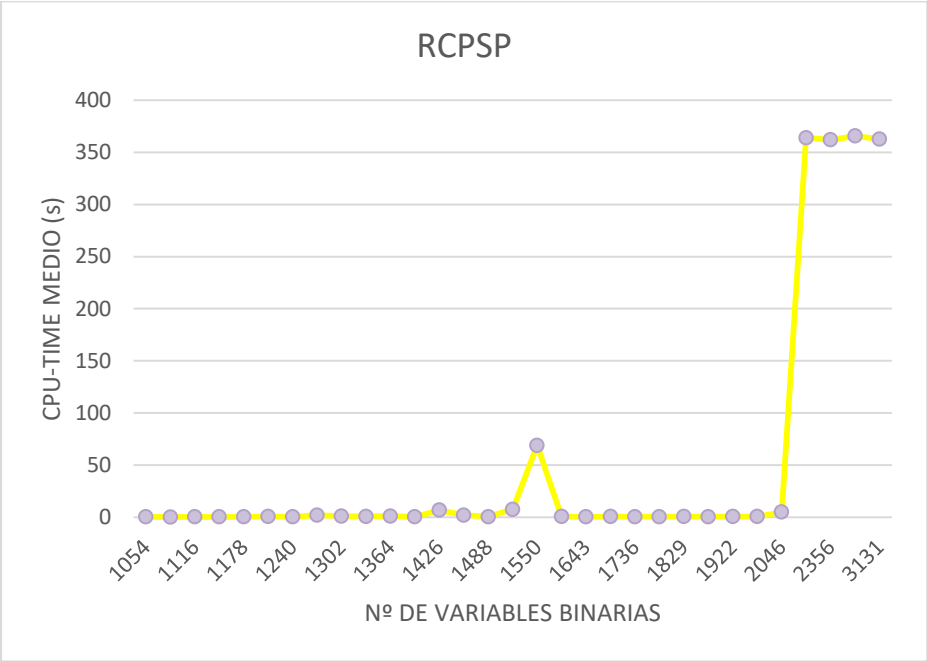
Ilustración 39: Solución con costes totales de trabajadores minimizado del experimento MRCPSP-Z n01_2

5.4 Análisis de los resultados

En este apartado se estudian los resultados obtenidos de los ejemplos de cada uno de los experimentos realizados. Primeramente, se analiza de forma individual cada experimento para, más tarde, hacerlo de forma conjunta. En ambos casos, las soluciones se representan en forma de gráfica para ver la variación de cada uno de ellos respecto a otro.

En el análisis de los resultados se tienen en cuenta diferentes variables como son los CPU-TIME medios, número de variables binario (NVB), número de trabajos, makespan y costes medios. También hay que comentar que se ha puesto un límite de tiempo para la resolución de cada uno de los experimentos siendo este de 360 segundos (6 minutos). Esto se ha realizado para facilitar la obtención de los resultados ya que de haber permitido que se hallase la solución sin limitación, podría llegar a tardar horas e incluso no obtener la solución óptima.
Otro aspecto a tener en cuenta, es que no se han realizado experimentos con 22 trabajos debido a que se ha visto que con menos trabajos los resultados obtenidos dan una solución suficiente de lo que se quiere representar.

Los primeros resultados a analizar son los relacionados con los tiempos de resolución y el número de variables binarias.

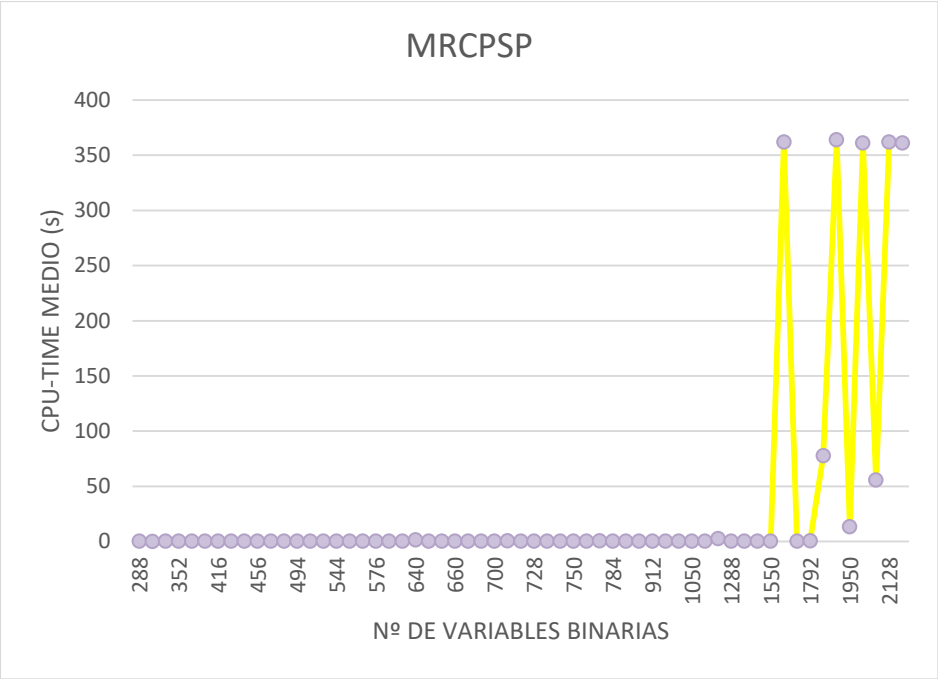


Gráfica 1: CPU-TIME Medio frente al Nº de Variables Binarias en el experimento RCPSP

En la *Gráfica 1* se expone el comportamiento del CPU-TIME Medio frente al número de variables binarias del experimento RCPSP. El CPU-TIME medio es la media entre los CPU-TIME de aquellos ejemplos con el mismo NVB.

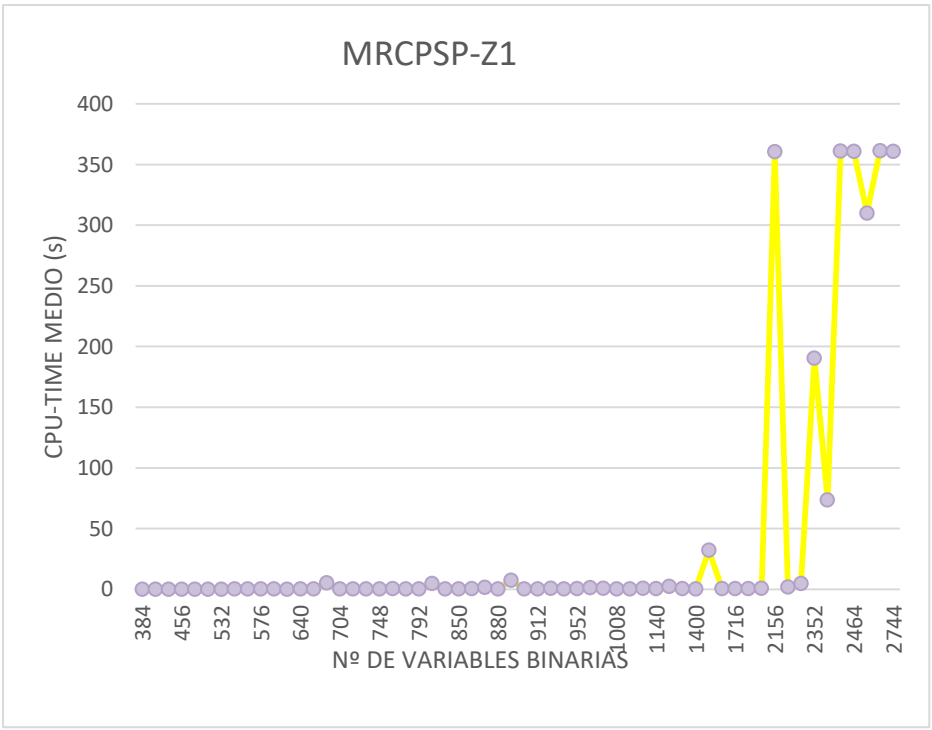
Como se puede observar, el CPU-TIME se mantiene constante, muy próximo a 0, cuando las variables tienen un valor pequeño. A medida que van creciendo, los CPU-TIME suben, e incluso en algunos ejemplos, pueden llegar a disminuir de nuevo. A partir de los 2000 NVB, el CPU-TIME asciende hasta llegar a los 360 (límite fijado) donde se mantiene constante para valores mayores de NVB. Es decir, a mayor número de variables binarias, mayor es el tiempo de resolución de los experimentos, excepto instancias puntuales.

La *Gráfica 2* muestra, de la misma forma que la anterior, el comportamiento de los CPU-TIME Medio respecto al número de variables binarias en este caso del experimento MRCPSP.
El CPU-TIME se mantiene cercano a 0 hasta que el NVB asciende hasta 1550, donde a partir de este momento el CPU-TIME realiza una serie de subidas y bajadas que termina a partir de las 2000 variables binarias que se mantiene constante, más o menos, en los 360 segundos.

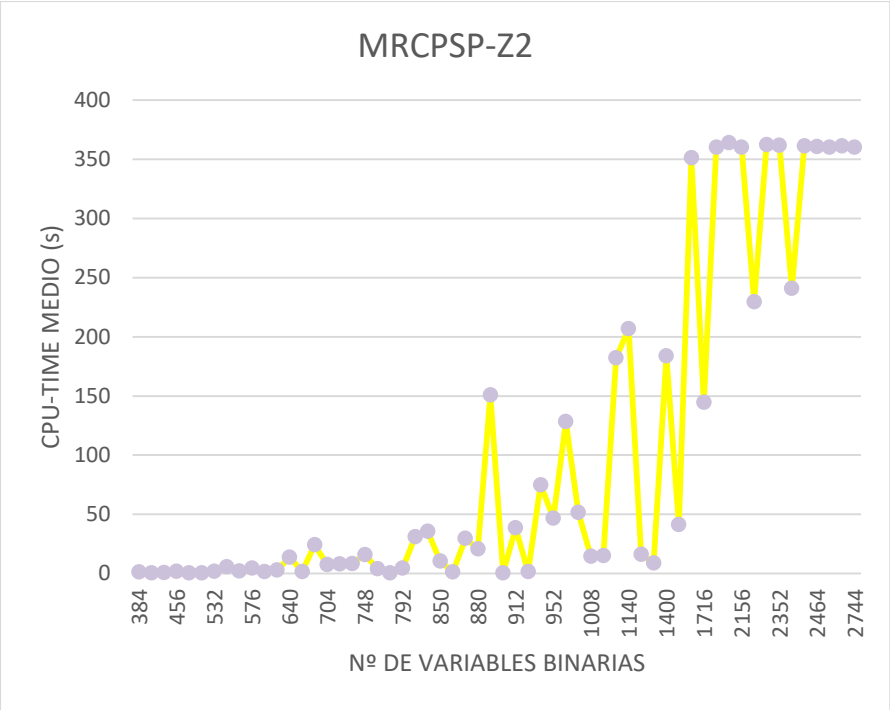


Gráfica 2: CPU-TIME Medio frente al Nº de Variables Binarias en el experimento MRCPSP

Las *Gráficas 3 y 4* representan lo mismo que las anteriores pero, en este caso, respecto a los experimentos propios de MRCPSP con zonas y diferenciando las dos funciones objetivos que se nombraron en capítulos anteriores (MRCPSP-Z1 y MRCPSP-Z2).



Gráfica 3: CPU-TIME Medio frente al Nº de Variables Binarias en el experimento MRCPSP-Z1

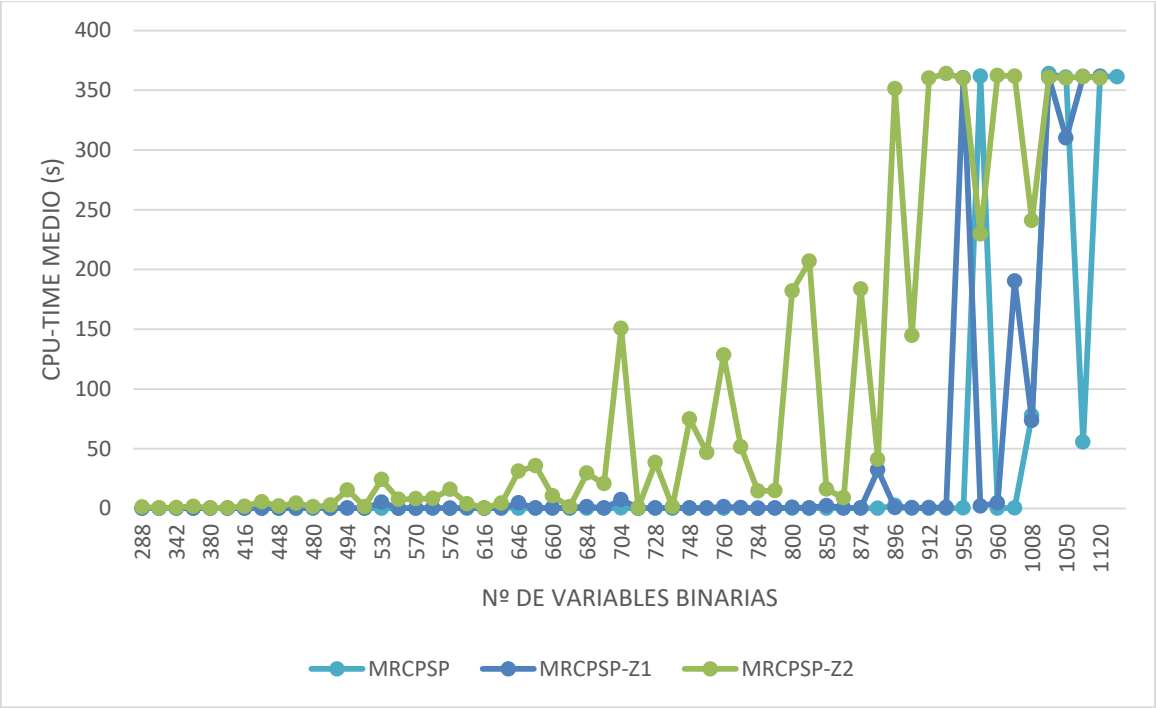


Gráfica 4: CPU-TIME Medio frente al Nº de Variables Binarias en el experimento MRCPSP-Z2

En la gráfica de MRCPSP-Z1 se observa que los CPU-TIME se mantienen muy cerca de 0 a medida que el NVB asciende teniendo, en algunos casos, subidas pero que rápidamente bajan. A partir de las 2000 variables binarias el CPU-TIME llega al límite de 360 segundos pero aparece una fase de ascensos y descensos irregulares hasta las 2500 variables aproximadamente que se mantiene de forma regular en los 360 segundos.

Algo diferente ocurre en los MRCPSP-Z2 donde los CPU-TIME crecen con un número de variables binarias menor que en los otros casos. A partir de las 650 variables empieza a haber un gran número de subidas y bajadas con CPU-TIME más elevados que en los ejemplos ya expuestos. Alrededor de las 2400 variables se encuentra la estabilidad en límite del tiempo.

La *Gráfica 5* representa los resultados de los experimentos tanto de MRCPSP como de MRCPSP-Z en una misma gráfica para así compararlos mejor. En ella, es fácil de comprobar que el inicio del MRCPSP junto con MRCPSP-Z1 es similar, mientras que en MRCPSP-Z2 muy pronto empiezan las irregularidades respecto a las otras dos hasta que llega a las 2000 variables que se regula. En ningún momento, esta última tiene alguna semejanza con las demás, que, por ejemplo, una vez llegadas a un número de variables similar empiezan a aparecer los crecimientos y decrecimientos manteniéndose en el límite de tiempo al mismo número de variables.



Gráfica 5: Comparativa CPU-TIME Medio frente al Nº de Variables Binarias en todos los experimentos

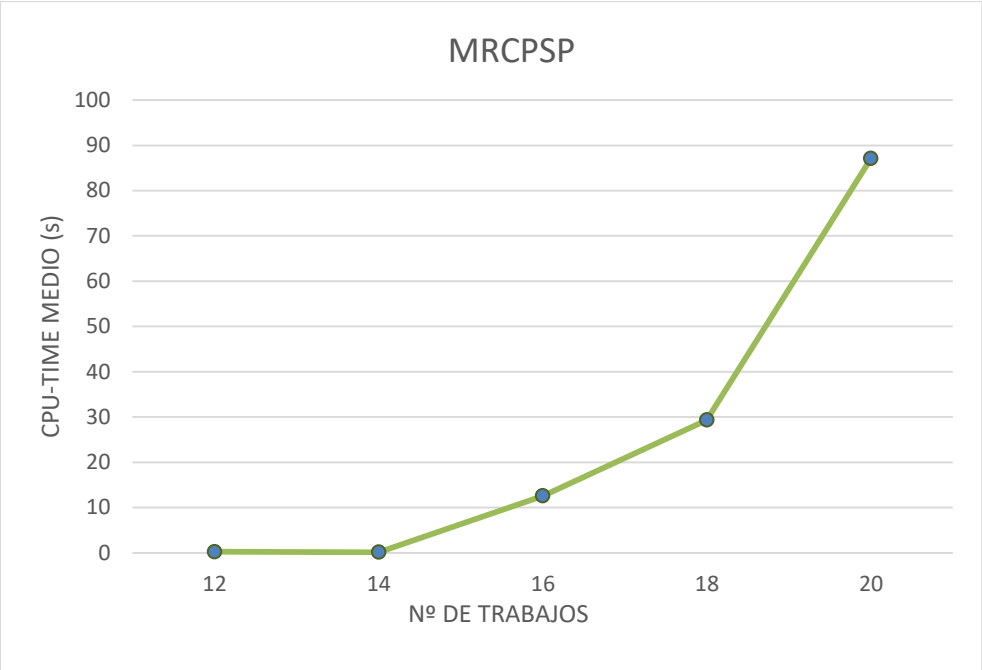
De las últimas gráficas se puede llegar a las siguientes conclusiones:

- Como es lógico, el número de variables binaria es una medida principal de la dificultad de la resolución de problemas mediante software de optimización como LINGO, donde se utilizan métodos de Branch & Bound.

- A partir de 2000 variables binarias aproximadamente empieza un crecimiento exponencial en este tipo de problemas, existiendo instancias puntuales donde se resuelve más rápido de lo habitual debido a los valores propios de esas instancias.

- Los problemas más sencillos son los RCPSP y los más complejos son los MRCPSP-Z y concretamente los que incluyen la función objetivo de minimizar costes (MRCPSP-Z2).

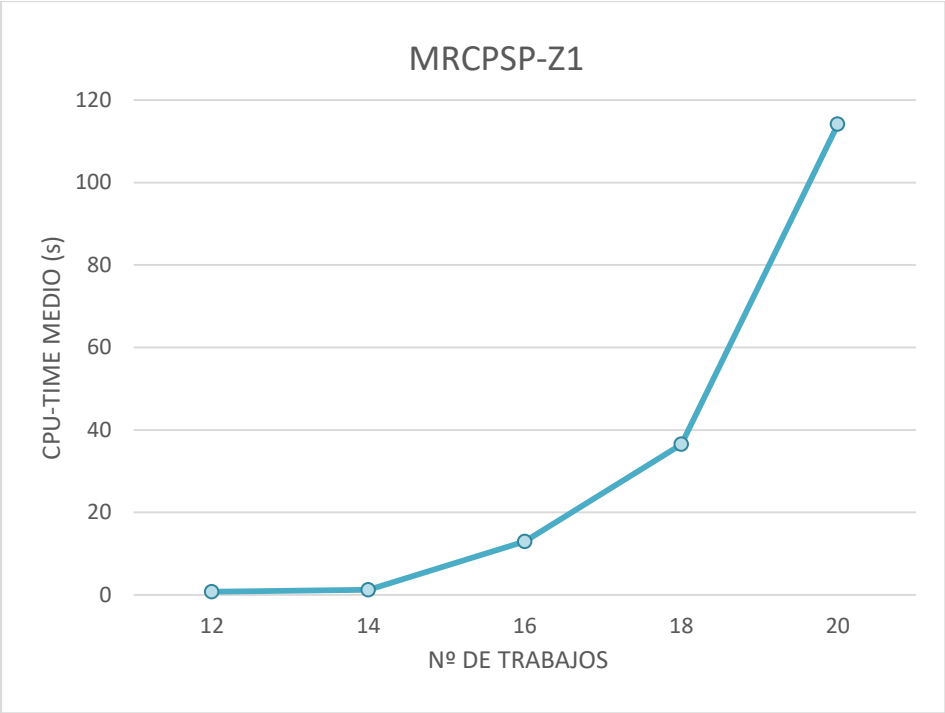
A continuación, se presentan los resultados obtenidos de CPU-TIME Medios respecto al número de trabajos de cada ejemplo. En este caso, el experimento RCPSP no se tendrá en cuenta ya que en él siempre se trabaja con 32 tareas.



Gráfica 6: CPU-TIME Medio frente al Nº de Trabajos en el experimento MRCPSP

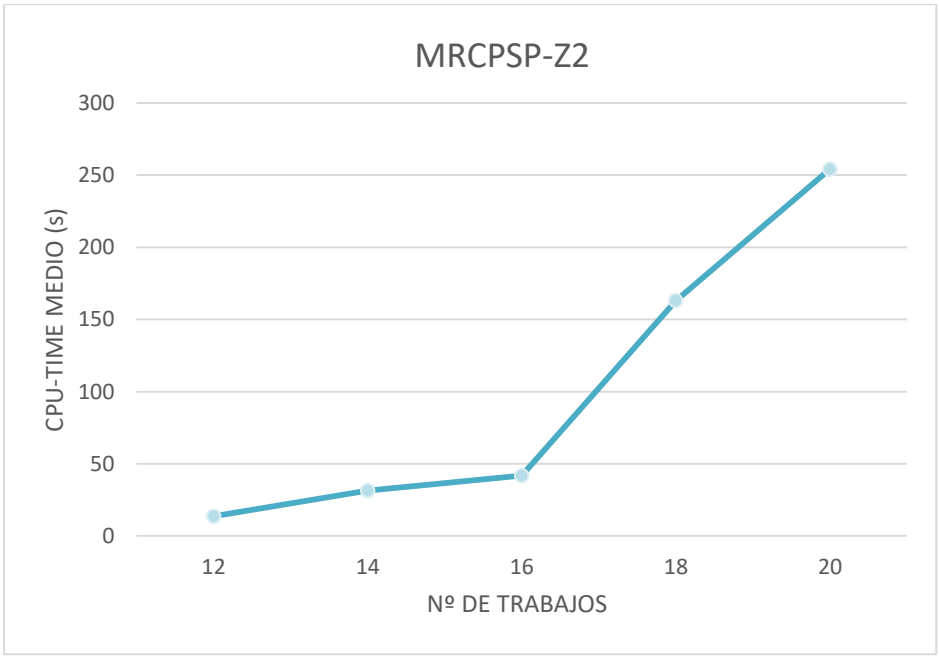
Como se puede observar claramente en la *Gráfica 6*, resulta una gráfica exponencial, es decir, que a mayor número de trabajos mayor es el tiempo de resolución. En este caso, los CPU-TIME son cercanos a 0 en los ejemplos de 12 trabajos pero va ascendiendo al igual que ascienden el número de tareas hasta llegar a casi 90 segundos para la resolución de los experimentos con 20 trabajos.

La *Gráfica 7* muestra el CPU-TIME Medio respecto al número de trabajos del experimento MRCPSP-Z1. La solución obtenida es similar a la del MRCPSP pero la media de los CPU-TIME es mayor. De igual modo, a mayor número de trabajo mayor es el CPU-TIME donde los ejemplos con 20 trabajos se resuelven en 115 segundos.



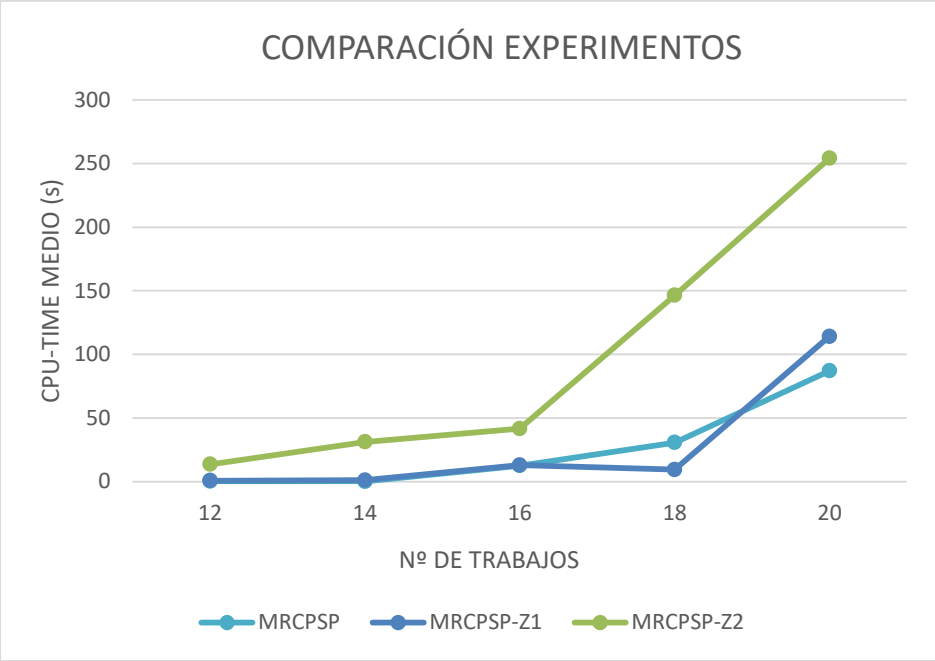
Gráfica 7: CPU-TIME Medio frente al Nº de Trabajos en el experimento MRCPSP-Z1

La *Gráfica 8* expone los valores de resolución del experimento de MRCPSP-Z2, los cuales son un poco diferentes a los anteriores ya que en los casos de 12,14 y 16 trabajos la media del CPU-TIME asciende ligeramente a medida que suben las tareas (no más de 50 segundos) y no es hasta que se trabaja con 18 trabajos que el CPU-TIME crece en mayor magnitud hasta llegar a los 254 segundos de media de los ejemplos con 20 tareas.



Gráfica 8: CPU-TIME Medio frente al Nº de Trabajos en el experimento MRCPSP-Z2

En la *Gráfica 9* se muestra una representación conjunta de los tres casos. En todos los casos, la representación final de la gráfica es una exponencial ya que a mayor número de trabajos en el experimento el tiempo para hallar la solución es mayor ya que es más difícil hacerlo.
En el experimento MRCPSP-Z2 se ve claramente que una vez que se alcanzan las 18 tareas el tiempo asciende rápidamente mientras antes lo hacía ligeramente en los ejemplos con menores trabajos. En los otros dos casos, el inicio es similar teniendo un CPU-TIME muy parecido. Una vez que se superan los 16 trabajos, MRCPSP-Z1 disminuye los tiempos de resolución mientras el MRCPSP los aumenta. Lo contrario ocurre cuando los ejemplos superan los 18 trabajos donde a MRCPSP-Z1 tarda más en resolverlos.



Gráfica 9: Comparativa CPU-TIME Medio frente al Nº de Trabajos en todos los experimentos

De las últimas gráficas se puede llegar a las siguientes conclusiones:

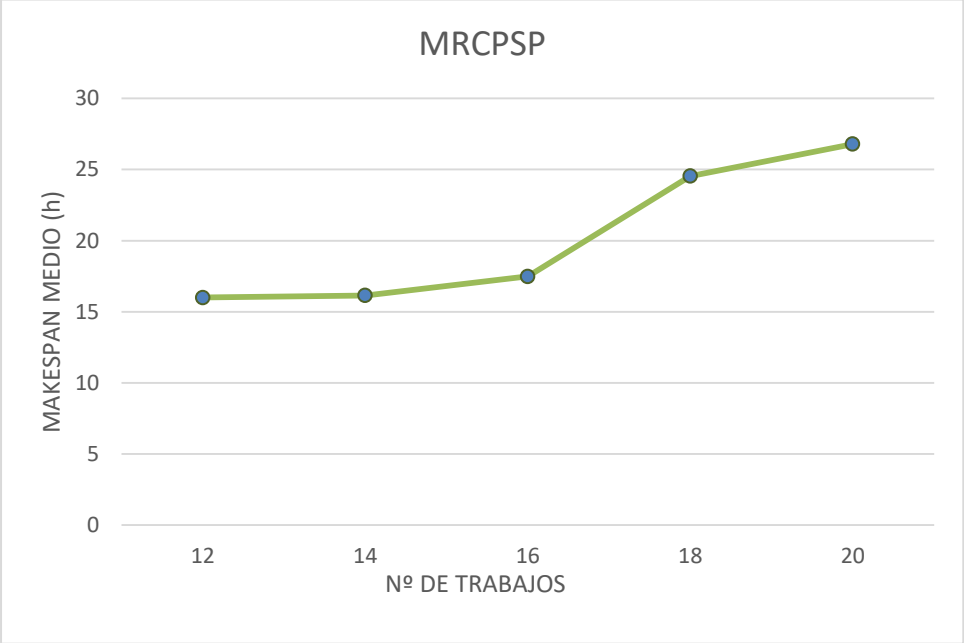
- El número de trabajos de los problemas *n_0* (MRCPSP) de la librería PSPLIB es otra medida principal de la dificultad de la resolución de problemas mediante software de optimización como LINGO al estar directamente relacionado con el número de variables binarias.

- Se observa claramente el crecimiento exponencial de los problemas con el número de trabajos al ser estos problemas de Clase NP.

- Se confirma, con el número de trabajos como referencia, que dentro de los problemas MRCPSP los más complejos son los MRCPSP-Z2 por la definición de las variables asociadas a la función objetivo de costes (es necesario calcular el número máximo de trabajadores usados de cada tipo).

Otro método de estudio de los experimentos realizados es el análisis en función del makespan respecto al número de trabajo de cada uno de ellos. En el experimento de MRCPSP el makespan que se obtiene es idéntico al makespan obtenido por métodos propios. En el caso

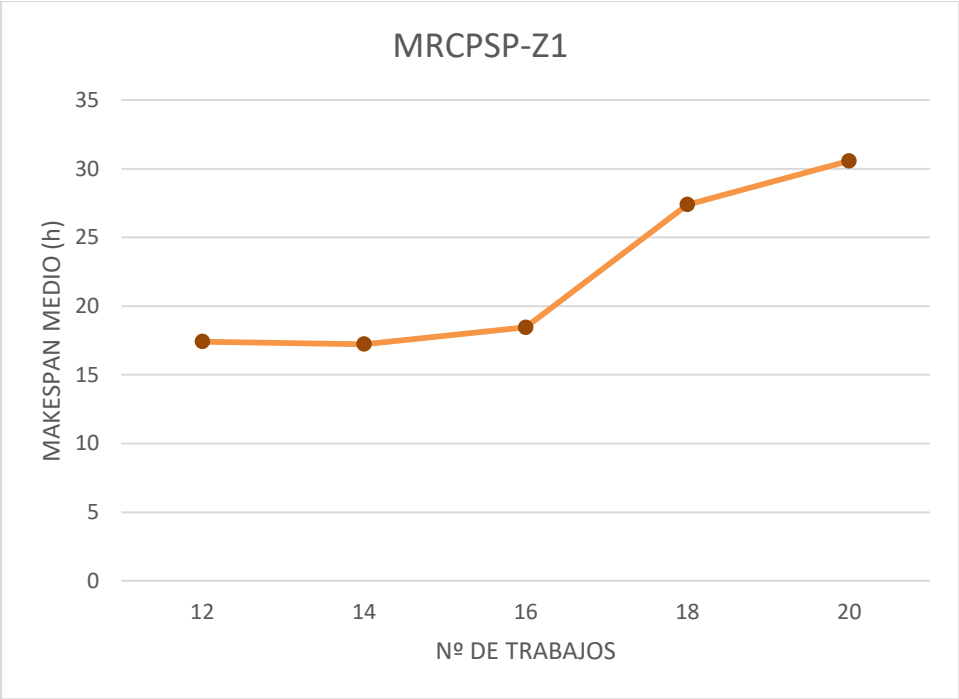
de MRCPSP-Z1 el makespan es igual o menor al calculado como se explicó en puntos anteriores. En MRCPS-Z2 el makespan obtenido es igual o superior al obtenido en MRCPSP-Z1. Hay que recordar que en los problemas con zonas la primera función objetivo minimiza el makespan (de ahí que sea menor o igual que en Z2) y la segunda función objetivo es minimizar los costes.



Gráfica 10: Makespan Medio frente al Nº de Trabajos en el experimento MRCPSP

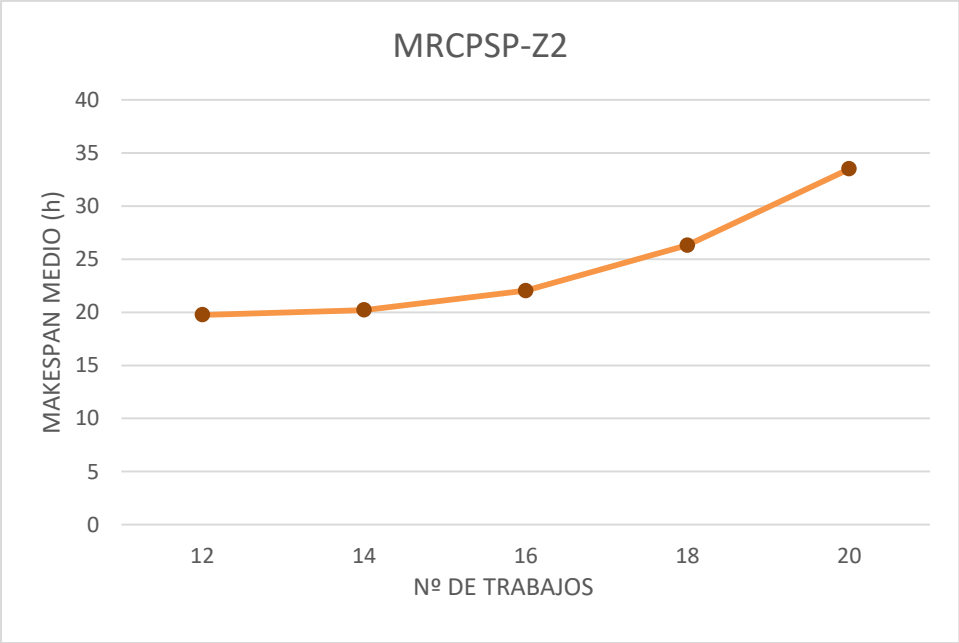
En la *Gráfica 10* muestra cómo asciende el makespan en función del número de trabajo creciendo levemente desde los 12 trabajos a los 16 y de una forma más gradual cuando supera estos últimos y más aún con 20 tareas. A mayor número de trabajo, mayor es el makespan.

Con la Gráfica 11 se demuestra una cierta similitud con la forma de representación del MRCPSP pero donde los valores de cada experimento son mayores.



Gráfica 11: Makespan Medio frente al Nº de Trabajos en el experimento MRCPSP-Z1

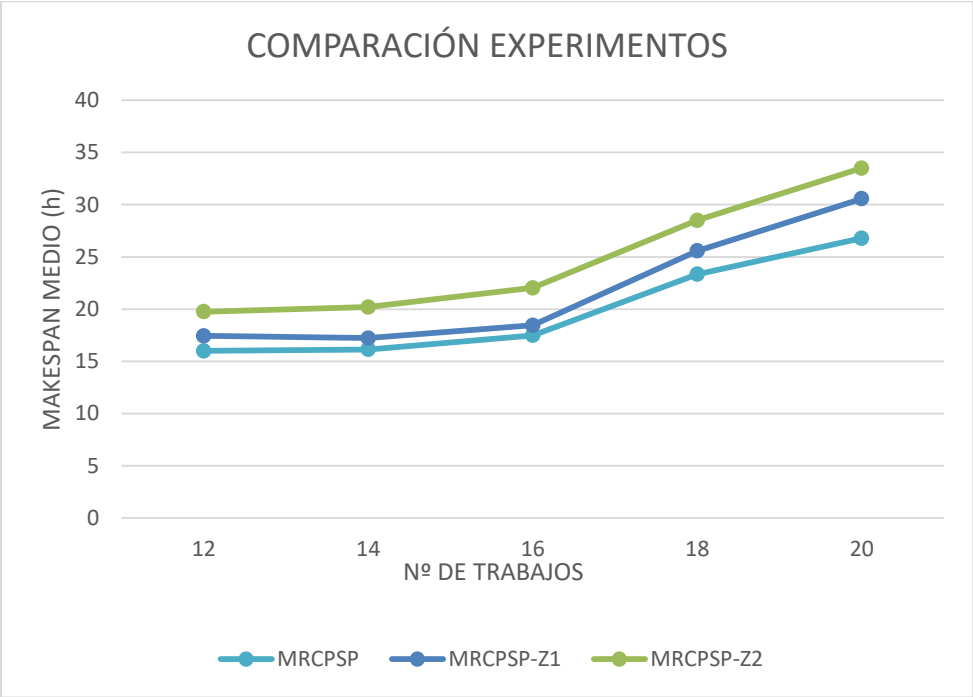
La *Gráfica 12* expone la gráfica del MRCPSP-Z2 donde se puede observar un crecimiento exponencial en los resultados.



Gráfica 12: Makespan Medio frente al Nº de Trabajos en el experimento MRCPSP-Z2

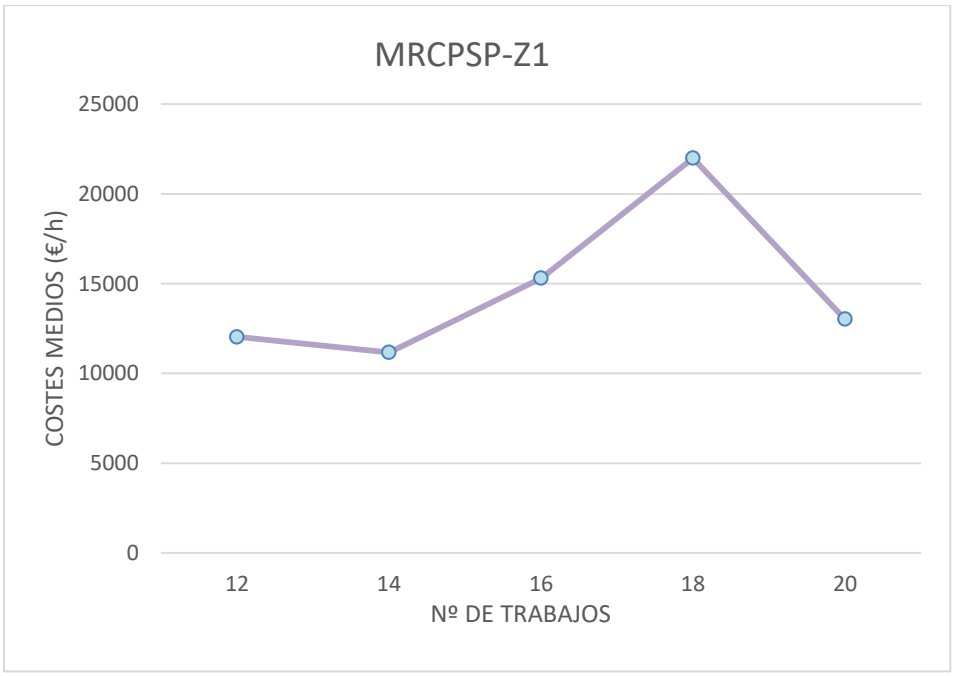
La *Gráfica 13* representa el conjunto de soluciones del makespan respecto al número de trabajos de los tres experimentos. Como es de imaginar, en todos los casos el makespan es mayor mientras más tareas hay que realizar pero más lo es aún en MRCPSP-Z2 ya que no se

está minimizando. En MRCPSP-Z1 es menor ya que en ese caso sí se minimiza pero mayor que el MRCPSP.



Gráfica 13: Comparativa Makespan Medio frente al Nº de Trabajos en todos los experimentos

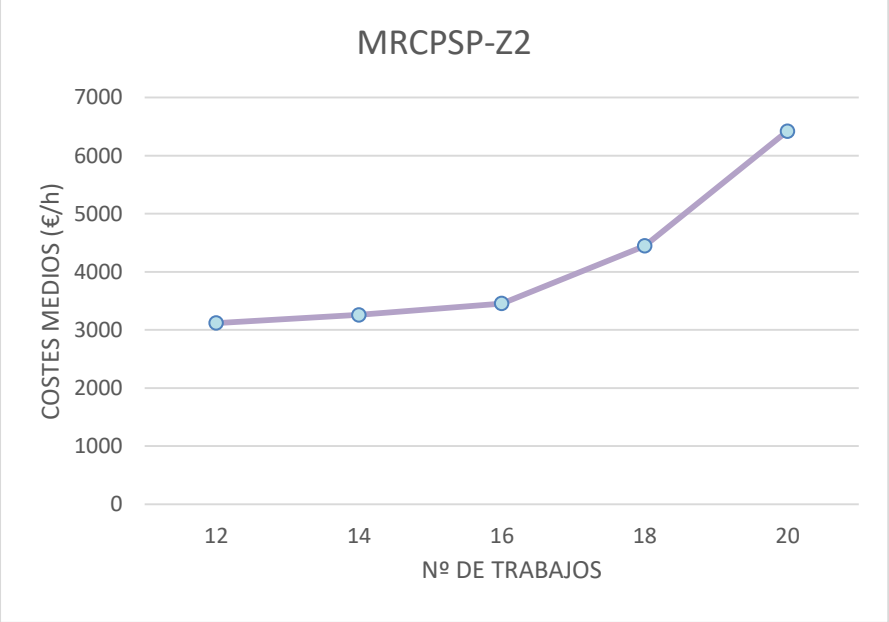
Para terminar el análisis de los resultados, se hace una comparativa de los MRCPSP con zonas en función de sus costes medios totales y el número de trabajos de cada experimento. En MRCPSP-Z1 los costes no se minimizan, por lo tanto, son mucho mayores a los de MRCPSP-Z2 donde sí se minimizan.



Gráfica 14: Coste Medio frente al Nº de Trabajos en el experimento MRCPSP-Z1

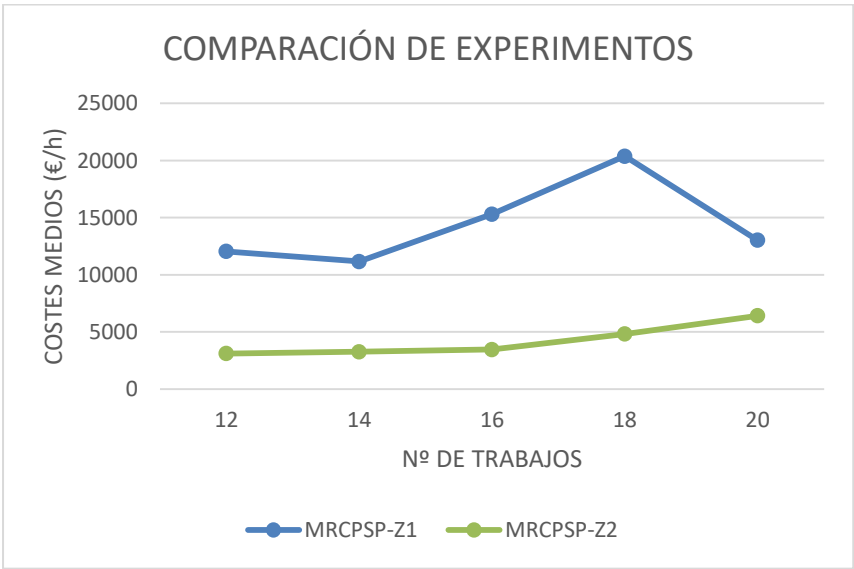
La *Gráfica 14* muestra el número de costes a lo largo de los diferentes números de trabajos de los experimentos. En este caso, los costes pueden ser mayores o menores en función tanto de las tareas como del número de trabajadores necesarios, y es por eso por lo que pueden aumentar o disminuir ya que no son óptimos.

En la *Gráfica 15* los costes están minimizados, por lo tanto, a medida que vayan creciendo los trabajos los costes serán mayores. Se observa que la representación es de forma exponencial.



Gráfica 15: Coste Medio frente al Nº de Trabajos en el experimento MRCPSP-Z2

La *Gráfica 16* es una representación conjunta de ambos casos en los que MRCPSP-Z1 está por encima ya que los costes son superiores al no estar optimizados; mientras que MRCPSP-Z2 está por debajo porque sí lo está. Por lo tanto, se muestra la diferencia de la optimización.



Gráfica 16: Comparativa Costes Medios frente al Nº de Trabajos en todos los experimentos

De las últimas gráficas se puede llegar a las siguientes conclusiones:

- Como era lógico, se comprueba que en los problemas *n_0* (MRCPSP) de la librería PSPLIB el Makespan aumenta con el número de trabajos. Esto se confirma tanto en los problemas estándares como cuando se incorporan las zonas.

- Se confirma, que en todos los casos, el Makespan obtenido en los problemas estándares MRCPSP es menor que cuando se añaden restricciones de zona (MRCPSP-Z).

- Además en los mismos problemas con zonas, si se minimiza el Makespan (MRCPSP-Z1) los valores del mismo son menores que si se minimiza el Coste Total (MRCPSP-Z2).

- De igual forma, en los mismos problemas con zonas, si se minimiza el Coste Total (MRCPSP-Z2) los valores del mismo son menores que si se minimiza el Makespan (MRCPSP-Z1).

- Este estudio con la función objetivo de minimizar los costes totales de mano de obra es muy poco habitual pero de enorme importancia en estaciones de trabajo aeronáuticas donde las actividades son intensivas en mano de obra. Con los resultados de los problemas MRCPSP-Z2 se alcanzan reducciones importantes en costes respecto a los estudios más frecuentes que se hacen usando como referencia el Makespan (MRCPSP-Z1).

6. Conclusión y futuras líneas

En este Trabajo Fin de Grado se ha abordado el problema de Programación de Proyectos con Recursos Limitados en sus versiones estándares tanto con un único modo de ejecución, RCPSP, como con más de un modo o multi-modo, MRCPSP. De la misma forma e introduciendo aspectos del sector aeronáutico, se han propuesto una serie de problemas propios basándose en el MRCPSP estándar añadiéndole restricciones de zonas, lo cual hace que la dificultad del problema aumente. Estos problemas con zonas de trabajo se han estudiado con dos funciones objetivos: minimizar el *makespan* (MRCPSP-Z1) que es la habitual en los estudios, y minimizar los costes totales de mano de obra (MRCPSP-Z2) que es innovador.

Estos problemas RCPSP y MRCPSP se han estudiado a lo largo de la literatura desarrollándose una serie de técnicas para hallar su solución, las cuales se citan en este trabajo. Estos métodos pueden ser exactos o aproximados siendo estos primeros no aplicables a todos los problemas y dando mejores resultados en los segundos.
En este caso, en los problemas propios, también se ha llevado a cabo un modo de resolución para su estudio.

Una vez analizados este tipo de problemas, se han planteado modelos de programación lineal de ellos. Se han resuelto baterías de experimentos de estos problemas basados en las librerías PSPLIB de muchos de ellos. Finalmente se ha realizado un estudio en función de una serie de variables que permiten llegar a conclusiones finales.
Para analizar los resultados se han tenido en cuenta variables como el tiempo de resolución o CPU-TIME, el número de trabajos, número de variables, makespan o costes totales.

En el experimento de RCPSP, donde en todos los ejemplos siempre había 32 trabajos, se puede decir que a medida que el número de variables ascendía los tiempos para encontrar una solución óptima también lo hacían. Esto se debe a que mientras más variables binarias existen en el problema, mayor es su dificultad, y esto exige mayor tiempo para su resolución. Se ha validado el modelo lineal planteado pues se han obtenido los mismos resultados que los óptimos conocidos.

Con los problemas MRCPSP y MRCPSP con Zonas se puede hacer una comparativa conjunta. Ya de por sí se entiende que los problemas con restricciones de zonas son más complicados ya que tienen un mayor número de limitaciones, y por lo tanto, se tardará más en alcanzar su óptimo. Con todo ello, mientras que en MRCPSP y MRCPSP-Z1 (minimiza el makespan) los CPU-TIME son pequeños hasta que no alcanzan un número de variables binarias considerable, en los MRCPSP-Z2 (minimiza los costes) desde muy pronto los CPU-TIME aumentan. Esto se debe a la dificultad de encontrar una solución con mínimo coste al ser necesario la definición de nuevas variables que calculen el número máximo de trabajadores usados de cada tipo, frente a encontrar una solución de makespan mínimo donde no es necesario.

Lo mismo ocurre al comparar los CPU-TIME con el número de trabajos donde los MRCPSP-Z2 tardan más ya que su dificultad es siempre mayor que la del resto y los MRCPSP-Z1 mayor que los MRCPSP.

En estos tipos de problemas, también se puede sacar conclusiones en función del número de trabajos. Tanto en el MRCPSP como en los MRCPSP-Z, el makespan de todos ellos es superior mientras mayor sea el número de trabajos a realizar. Está claro que a mayor número de tareas a realizar, el tiempo para realizarlas todas debe ser mayor. Dependiendo de las tareas y de la dificultad del experimento, el makespan será mayor siendo el MRCPSP-Z2 (que no minimiza el makespan) el que más tiempo consuma.
Si se comparan los costes, claramente se sabe que los de MRCPSP-Z1 son mayores porque es una función objetivo que no se está minimizando (sí lo hace el makespan). De forma viceversa ocurre en MRCPSP-Z2.

Como ya se ha anotado con anterioridad, la resolución de estos experimentos y su tiempo para obtener el óptimo dependen de la dificultad de cada uno de ellos en función del número de trabajos, variables binarias y costes. Mientras mayor sea el número de tareas a realizar y variables binarias, mayor será el CPU-TIME. Los costes dependerán de lo que se quiera minimizar.

El estudio llevado a cabo en este TFG se ha centrado en la obtención del makespan óptimo en los tres tipos de experimentos, además de minimizar los costes en MRCPSP-Z2. Este último estudio con la función objetivo de minimizar los costes totales de mano de obra es muy poco habitual en la literatura. Sin embargo son de enorme importancia en estaciones de trabajo aeronáuticas donde las actividades son principalmente manuales. Con los resultados de los problemas MRCPSP-Z2 se alcanzan reducciones importantes en costes respecto a los estudios más frecuentes que se hacen usando como referencia el Makespan (MRCPSP-Z1) y alcanzando soluciones dentro del tiempo de ciclo que marca el instante final disponible en la estación de montaje.

Estos tipos de problemas también pueden ser útiles para obtener otros objetivos diferentes y ampliarse con nuevas restricciones como pueden ser paradas entre las actividades, tiempos de set-ups o diferentes relaciones de precedencia.

También sería muy interesante añadir más de una función objetivo a los problemas de RCPSP, aparte de minimizar el tiempo de ejecución del proyecto.

Los resultados obtenidos se han conseguido en un tiempo corto de ejecución, lo cual permite la incorporación de estos métodos a software profesionales de gestión de proyectos, siendo esto una posible dirección de trabajo en el futuro.

7. Bibliografía

- Alcaraz, J., & Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of operations Research*, *102*(1), 83-109.

- Alcaraz, J., & Maroto, C. (2006). A hybrid genetic algorithm based on intelligent encoding for project scheduling. In *Perspectives in modern project scheduling* (pp. 249-274). Springer, Boston, MA.

- Alcaraz, J., Maroto, C., & Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, *54*(6), 614-626.

- Baar, T., Brucker, P., & Knust, S. (1999). Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem. In *Meta-Heuristics* (pp. 1-18). Springer, Boston, MA.

- Boctor, F. F. (1996). A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research*, *90*(2), 349-361.

- Boctor, F. F. (1996). Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, *34*(8), 2335-2351.

- Boctor, F. F. (1993). Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *The international journal of production research*, *31*(11), 2547-2558.

- Bouleimen, K. L. E. I. N., & Lecocq, H. O. U. S. N. I. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European journal of operational research*, *149*(2), 268-281.

- Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European journal of operational research*, *107*(2), 272-288.

- Christofides, N., Alvarez-Valdés, R., & Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*(3), 262-273.

- Debels, D., & Vanhoucke, M. (2005, May). A bi-population based genetic algorithm for the resource-constrained project scheduling problem. In *International Conference on Computational Science and Its Applications* (pp. 378-387). Springer, Berlin, Heidelberg.

- Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management science*, *38*(12), 1803-1818.

- Demeulemeester, E. L., & Herroelen, W. S. (2006). *Project scheduling: a research handbook* (Vol. 49). Springer Science & Business Media.

- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, *13*(5), 533-549.

- Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, *49*(5), 433-448.

- Hartmann, S., & Drexl, A. (1998). Project scheduling with multiple modes: A comparison of exact algorithms. *Networks: An International Journal*, *32*(4), 283-297.

- Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *127*(2), 394-407.

- Igelmund, G., & Radermacher, F. J. (1983). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, *13*(1), 1-28.

- Józefowska, J., Mika, M., Różycki, R., Waligóra, G., & Węglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, *102*(1-4), 137-155.

- Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, *34*(5), 975-986.

- Klein, R. (2000). Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects. *European Journal of Operational Research*, *127*(3), 619-638.

- Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project scheduling* (pp. 147-178). Springer, Boston, MA.

- Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, *174*(1), 23-37.

- Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science*, *41*(10), 1693-1703.

- Lova, A., Tormos, P., & Barber, F. (2006). Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, *10*(30), 69-86.

- Mori, M., & Tseng, C. C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, *100*(1), 134-141.

- Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In *Essays and surveys in metaheuristics* (pp. 557-588). Springer, Boston, MA.

- Pinson, E., Prins, C., & Rullier, F. (1994). Using tabu search for solving the resource-constrained project scheduling problem. In *International Workshop on Project Management and Scheduling, PMS'94*.

- Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, *16*(1), 93-108.

- Słowiński, R., Soniewicki, B., & Węglarz, J. (1994). DSS for multiobjective project scheduling. *European Journal of Operational Research*, *79*(2), 220-229.

- Sprecher, A. (2012). *Resource-constrained project scheduling: Exact methods for the multi-mode case* (Vol. 409). Springer Science & Business Media.

- Sprecher, A., Hartmann, S., & Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum*, *19*(3), 195-203.

- Stinson, J. P., Davis, E. W., & Khumawala, B. M. (1978). Multiple resource–constrained scheduling using branch and bound. *AIIE Transactions*, *10*(3), 252-259.

- Talbot, F. B. (1981). An Algorithm for a general class of precedence and resource constrained scheduling problems.

- Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management science*, *28*(10), 1197-1210.

- Thomas, P. R., & Salhi, S. (1998). A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics*, *4*(2), 123-139.

- Valls, V., Ballestín, F., & Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, *165*(2), 375-386.

- Wall, M. B. (1996). *A genetic algorithm for resource-constrained scheduling* (Doctoral dissertation, Massachusetts Institute of Technology).

8. Anexos

1. Representación en tablas de las soluciones de los experimentos de RCPSP.

					RCPSP	
ORDEN	Parameter	Instance	Makespan	CPU-Time [sec.]	Nº Var Bin	CPU-Time-LINGO [sec.]
1	16	7	34	0,01	1054	0,258
2	12	8	35	0,01	1085	0,254
3	32	7	35	0,01	1085	0,231
4	7	6	35	0,02	1085	0,218
5	14	6	35	0,75	1085	0,271
6	16	3	36	0,01	1116	0,262
7	11	7	36	0,5	1116	0,32
8	20	10	37	0	1147	0,21
9	12	3	37	0,01	1147	0,228
10	6	6	37	0,04	1147	0,387
11	4	9	38	0,01	1178	0,227
12	19	9	38	0,01	1178	0,226
13	24	8	38	0,01	1178	0,254
14	2	1	38	0,02	1178	0,248
15	11	10	38	0,02	1178	0,282
16	8	9	39	0,01	1209	0,268
17	19	4	39	0,02	1209	0,218
18	6	8	39	0,36	1209	0,324

2. Representación en tablas de las soluciones de los experimentos de MRCPSP.

						MRCPSP		
ORDEN	Paramter	Instance	Makespan	CPU-Time [sec.]	Nº Trabajos	Nº Var Bin	Makespan	CPU-Time-LINGO [sec.]
1	3	9	9	0,03	12	288	9	0,077
2	7	8	9	0,03	12	288	9	0,105
3	4	7	10	0	12	320	10	0,076
4	15	3	9	0	14	342	9	0,078
5	3	1	11	0	12	352	11	0,08
6	8	3	11	0	12	352	11	0,086
7	4	3	11	0,04	12	352	11	0,082
8	12	8	10	0,03	14	380	10	0,078
9	3	4	12	0	12	384	12	0,081
10	4	8	12	0	12	384	12	0,082
11	4	9	12	0,03	12	384	12	0,079
12	8	9	12	0,03	12	384	12	0,085
13	6	8	12	0,04	12	384	12	0,089
14	2	3	13	0	12	416	13	0,088
15	3	2	13	0	12	416	13	0,094
16	3	6	13	0	12	416	13	0,086
17	8	5	13	0	12	416	13	0,092
18	4	2	13	0,03	12	416	13	0,088
19	6	3	13	0,03	12	416	13	0,135
20	12	2	11	0,03	14	418	11	0,119

3. Representación en tablas de las soluciones de los experimentos de MRCPSP-Z1

						MRCPSP-Z		MRCPSP-Z1		
ORDEN	Paramter	Instance	Makespan	CPU-Time [sec.]	Nº Trabajos	CYCLE	Nº Var Bin	CPU-Time-LINGO [sec.]	Makespan	Coste Total
1	3	9	9	0,03	12	12	384	0,185	9	6000
2	7	8	9	0,03	12	12	384	0,134	12	6600
3	4	7	10	0	12	13	416	0,17	10	10270
4	15	3	9	0	14	12	456	0,151	11	10260
5	3	1	11	0	12	14	448	0,156	11	10150
6	8	3	11	0	12	14	448	0,178	11	17080
7	4	3	11	0,04	12	14	448	0,141	11	12600
8	12	8	10	0,03	14	13	494	0,141	10	5460
9	3	4	12	0	12	15	480	0,159	12	3750
10	4	8	12	0	12	15	480	0,136	12	26325
11	4	9	12	0,03	12	15	480	0,131	12	21750
12	8	9	12	0,03	12	15	480	0,166	13	16125
13	6	8	12	0,04	12	15	480	0,154	15	3675
14	2	3	13	0	12	17	544	0,302	15	2720
15	3	2	13	0	12	17	544	0,186	13	6800
16	3	6	13	0	12	17	544	0,229	13	7905
17	8	5	13	0	12	17	544	0,191	13	20315
18	4	2	13	0,03	12	17	544	0,23	13	10540

4. Representación en tablas de las soluciones de los experimentos de MRCPSP-Z2

					MRCPSP-Z			MRCPSP-Z2		
ORDEN	Paramter	Instance	Makespan	CPU-Time[sec.]	Nº Trabajos	CYCLE	Nº Var Bin	CPU-Time-LINGO [sec.]	Makespan	Coste Total
1	3	9	9	0,03	12	12	384	2,421	12	1680
2	7	8	9	0,03	12	12	384	0,147	12	2760
3	4	7	10	0	12	13	416	0,478	12	1690
4	15	3	9	0	14	12	456	1,907	12	2880
5	3	1	11	0	12	14	448	1,062	13	1680
6	8	3	11	0	12	14	448	0,559	14	3220
7	4	3	11	0,04	12	14	448	0,72	13	1890
8	12	8	10	0,03	14	13	494	0,563	13	1625
9	3	4	12	0	12	15	480	0,462	15	1875
10	4	8	12	0	12	15	480	0,582	15	2475
11	4	9	12	0,03	12	15	480	0,322	15	1575
12	8	9	12	0,03	12	15	480	0,644	15	4050
13	6	8	12	0,04	12	15	480	0,144	15	3375
14	2	3	13	0	12	17	544	4,537	17	1870
15	3	2	13	0	12	17	544	0,689	17	1445
16	3	6	13	0	12	17	544	1,015	16	1360
17	8	5	13	0	12	17	544	15,431	17	4335
18	4	2	13	0,03	12	17	544	6,532	15	2210

5. Datos utilizados para la representación de las gráficas de RCPSP.

RCPSP	
Nº Var Bin	CPU-Time medio
1054	0,258
1085	0,244
1116	0,291
1147	0,275
1178	0,247
1209	0,795
1240	0,311
1271	2,015
1302	0,989
1333	0,820
1364	1,130
1395	0,386
1426	6,816
1457	1,993
1488	0,510
1519	7,503
1550	69,020
1581	0,592
1643	0,390
1705	0,559
1736	0,411
1798	0,420
1829	0,615
1860	0,407
1922	0,637
1984	0,582
2046	4,973
2139	363,967
2356	362,032
2604	365,769
3131	362,867

6. Datos utilizados para la representación de las gráficas de MRCPSP

MRCPSP	
Nº Var Bin	CPU-Time medio
288	0,091
320	0,076
342	0,078
352	0,083
380	0,078
384	0,083
416	0,097
418	0,119
448	0,093
456	0,092
480	0,201
484	0,085
494	0,088
512	0,158
532	0,161
544	0,096
570	0,114
572	0,117
576	0,109
608	0,110
616	0,100
640	1,496
646	0,142
650	0,113
660	0,109
672	0,135
684	0,245
700	0,098
704	0,475
722	0,119

Nº Var Bin	CPU-Time medio
728	0,158
736	0,137
748	0,126
750	0,119
760	0,142
768	0,441
784	0,114
792	0,135
850	0,163
912	0,141
952	0,143
968	0,171
1050	0,181
1120	0,160
1200	2,544
1288	0,239
1364	0,223
1450	0,273
1550	0,346
1716	361,745
1736	0,236
1792	0,459
1848	77,743
1900	363,793
1950	13,198
1960	360,900
2072	55,533
2128	361,809
2184	361,151
2300	0,404

Nº Trabajos	Makespan
12	16,000
14	16,143
16	17,483
18	24,538
20	26,786

Nº Trabajos	CPU-Time medio
12	0,254
14	0,139
16	12,595
18	29,347
20	87,074

7. Datos utilizados para la representación de las gráficas de MRCPSP-Z

MRCPSP-Z		
Nº Var Bin	CPU-Time medio F1	CPU-Time medio F2
384	0,16	1,284
416	0,17	0,478
448	0,158	0,78
456	0,151	1,907
480	0,149	0,431
494	0,141	0,563
532	0,186	1,797
544	0,228	5,641
570	0,269	2
576	0,247	4,516
608	0,309	1,48
616	0,184	2,82
640	0,333	13,795
646	0,236	1,589
684	5,223	24,261
704	0,26	7,589
722	0,275	8,077
736	0,333	8,442
748	0,304	16,022
760	0,669	3,953
768	0,264	0,441
792	0,28	4,44
800	4,687	31,13
836	0,425	35,737
850	0,328	10,51
864	0,598	1,398
874	1,584	29,621
880	0,368	20,773
896	7,426	150,783

Nº Var Bin	CPU-Time medio F1	CPU-Time medio F2
900	0,211	0,485
912	0,328	38,486
928	0,839	1,503
950	0,416	74,859
952	0,48	46,853
960	1,488	128,333
968	0,828	51,562
1008	0,236	14,593
1012	0,383	15,005
1100	0,9	182,106
1140	0,552	206,827
1232	2,403	16,195
1350	0,545	8,881
1400	0,418	183,69
1500	32,23	41,21
1624	0,709	351,26
1716	0,663	144,69
1850	0,723	360,209
1950	0,843	363,869
2156	360,534	360,236
2184	1,973	229,644
2240	4,824	362,297
2352	190,421	361,794
2400	73,661	240,941
2450	361,045	361,189
2464	360,754	360,768
2632	310,008	360,216
2688	361,394	361,23
2744	360,972	360,264
2900	1,962	362,328

Nº Trabajos	CPU-Time medio F1	CPU-Time medio F2
12	0,771	13,525
14	1,212	31,224
16	12,929	41,695
18	36,492	162,977
20	114,127	254,014

Nº Trabajos	Makespan F1	Makespan F2
12	17,414	19,754
14	17,224	20,204
16	18,448	22,034
18	27,385	26,308
20	30,571	33,5

Nº Trabajos	Coste Total F1	Coste Total F2
12	12030,286	3117,536
14	11166,224	3258,673
16	15302,586	3454,31
18	21990,385	4446,538
20	13028,571	6420