Universidad de Valladolid

Simulador para la resolución de problemas de programación multiproyecto con restricción de recursos.

Pablo Álvarez-Campana Rodríguez

MÁSTER EN DIRECCIÓN DE PROYECTOS
Departamento De Organización De Empresas Y C.I.M.
Universidad De Valladolid
España

INSISOC
SOCIAL SYSTEMS ENGINEERING CENTRE
2021

Simulador para la resolución de problemas de programación multiproyecto con restricción de recursos

Pablo Álvarez-Campana Rodríguez

MÁSTER EN DIRECCIÓN DE PROYECTOS
Departamento De Organización De Empresas Y C.I.M.
Universidad De Valladolid

Valladolid, Junio 2021

Tutores
David Jesús Poza García
Félix Antonio Villafañez Cardeñoso

AGRADECIMIENTOS

Deseo expresar mi más sincero agradecimiento a Jesús Álvarez – Campana, Irene Rodríguez y Álvaro Álvarez – Campana por su apoyo, confianza y cariño incondicional.

Me gustaría también agradecer a David Poza y Félix Villafañez su esfuerzo, trabajo e implicación como tutores del presente trabajo fin de máster.

RESUMEN

El contenido del presente trabajo fin de máster se basa en la creación de un simulador para la obtención de programaciones de proyectos con restricciones de recursos en entornos multi-proyecto. Además, el funcionamiento de este simulador se ha validado mediante la incorporación de instancias de la librería MPSPLib.com y la verificación de los resultados a través de diferentes métodos. Para la obtención de las programaciones se utiliza una nueva heurística propuesta: la heurística de cajones o heurística DH (*Drawers Heuristic*).

Palabras Clave

Simulador, Heurística, Dirección de Proyectos, RCMPSP, Programación multiproyecto.

ABSTRACT

This final thesis content is based on the development of a simulator which is able to solve some problems called as RCMPSP (Resource-Constrained Multi-Project Scheduling Problem). With the help of the online platform MPSPLib.com, the simulator as well as the results have been validated using different methods. In order to obtain the scheduling of the instances, a new proposed heuristic has been used, the *Drawers Heuristic*.

Keywords

Simulator, Heuristic, Project Management, RCMPSP, Multi-project Scheduling.

INDICE

INTRODUCCIÓN

Hoy en día muchas empresas realizan su actividad basada en proyectos. Estos proyectos a su vez están formados por actividades las cuales necesitan de una serie de recursos (recursos humanos, maquinaria, etc.) para ser desarrolladas y completadas. El problema surge cuando, en un mismo proyecto, dos o más actividades pueden comenzar debido a que todas las actividades que las preceden han terminado, pero existen recursos limitados que impiden su ejecución simultánea.

Los problemas de programación de actividades con limitación de recursos se conocen como RCPSP (por sus siglas en inglés: *Resource Constrained Project Scheduling Problem*) y han sido ampliamente estudiados durante décadas. No obstante, en la vida real, las empresas no suelen realizar un proyecto aislado, sino que su actividad se desarrolla dentro de una cartera de proyectos en la cual existen una serie de recursos limitados que son compartidos por varios proyectos. Así, el problema de programación de actividades en un proyecto se ve magnificado al existir no solo uno, sino varios proyectos con sus correspondientes actividades las cuales hay que programar de acuerdo a la restricción de recursos. En este marco se añade la caracterización de "multiproyecto" a RCPSP pasando a ser RCMPSP (por sus siglas en inglés: *Resource Constrained Multi Project Scheduling Problem*).

Los recursos disponibles por una empresa para realizar un conjunto de actividades se pueden catalogar como locales (pertenecen solo a un proyecto) y globales (son compartidos por varios proyectos). Por norma general, los recursos globales suelen ser recursos caros o con un alto nivel de formación cuando hablamos de personas y, por ende, las empresas tienden a minimizar estos recursos (Villafañez *et al.*, 2014). Este es el principal motivo por el cual se ha estudiado durante años los problemas de programación de actividades.

Por otro lado, detrás de la programación de las actividades de los proyectos puede haber diferentes intereses catalogados como "funciones objetivo" tales como minimizar la duración total de un conjunto de proyectos, minimizar el coste de los proyectos o reducir la variabilidad de los retrasos de los proyectos. Dependiendo del objetivo de la empresa donde se realizan los proyectos, se utiliza diferentes métodos para cumplir con el objetivo propuesto dadas una serie de restricciones.

Las técnicas y métodos para resolver este tipo de problemas han sido y siguen siendo tema de discusión. Encontrar un método general que sea capaz de resolver todos los problemas de manera óptima supondría un gran avance y beneficio para las empresas. Desgraciadamente, a día de hoy no existe un método generalista de este tipo dado que el mejor método depende de las características del problema. Debido a la complejidad de los problemas de programación de actividades con recursos limitados se utilizan diversos softwares para conseguir una solución que satisfaga las funciones objetivo y a la vez respete las restricciones de recursos. Estos softwares, debido a la gran complejidad computacional que plantean los problemas a tratar, no son capaces de obtener una solución cercana al óptimo, sino que plantean una única solución de nivelación de recursos basándose en un algoritmo predefinido. Este es uno de los motivos por los que la comunidad investigadora trata de encontrar algoritmos más eficaces que se encuentren cerca del resultado óptimo mediante la utilización de heurísticas y metaheurísticas. En este marco se desarrolla el trabajo que expondré a continuación: un simulador capaz de obtener soluciones de programación de actividades con restricciones de recursos dentro de una cartera de proyectos utilizando técnicas heurísticas basadas en reglas de prioridad. Este simulador podría ser utilizado en un futuro por investigadores, para comprobar nuevas heurísticas o metaheurísticas; por empresas, debido a la

viabilidad de relacionar este simulador con Microsoft Project y por directores de proyecto, para saber la utilización de los recursos y tomar decisiones respecto a esta información.

Objetivo del Proyecto

El objetivo principal de este proyecto es el desarrollo de una herramienta para la programación de actividades con recursos limitados en un entorno accesible por todo el mundo (entorno Microsoft). Además, la herramienta tiene el objetivo de ser eficaz, sencilla e intuitiva para su utilización pudiendo llegar a competir en un futuro con otras herramientas para la programación de actividades las cuales necesitan un aprendizaje específico.

Alcance del Proyecto

Las tareas a realizar en este trabajo serán:

- Explicación del problema, características y métodos de resolución. Realización de una labor de revisión del estado del arte acerca de los problemas de programación (proyecto y multi-proyecto) con restricción de recursos.
- Presentación del simulador realizado detallando sus diferentes partes, funcionamiento, utilidades, ventajas y limitaciones.
- Comprobación del funcionamiento del simulador mediante los problemas de la librería MPSPLib utilizando la heurística DH (*Drawers Heuristic* – Heurística de cajones) para la resolución de estos problemas.
- Extracción de las conclusiones del trabajo y propuesta de futuras líneas de desarrollo para el perfeccionamiento de la herramienta.

Motivación del Proyecto

El motivo de realizar este simulador no es otro que crear un programa para la resolución de este tipo de problemas que sea sencillo, intuitivo de manejar y que permita comparar diferentes reglas de prioridad en la programación de actividades con limitación de recursos de cara a futuras investigaciones.

Estructura del Documento

El presente trabajo está estructurado como sigue:

- En el primer capítulo se realiza por una parte una explicación y clasificación de los problemas de programación con restricciones de recursos. También se plantean los diferentes métodos existentes para la resolución de estos problemas. Por otra parte, se lleva a cabo una revisión del estado del arte en este ámbito de la dirección de proyectos.
- En el segundo capítulo se presenta el simulador desarrollado para obtener soluciones a estos problemas y se detalla en profundidad el lenguaje utilizado, sus partes y su funcionamiento. Se presentan sus ventajas y asimismo se identifican algunas de sus limitaciones.
- El tercer capítulo está dedicado a la comprobación del simulador mediante una nueva heurística (*Drawers Heuristic* "DH" – Heurística de cajones). También se compara esta heurística con otros métodos mediante la librería MPSPLib.

- Por último, en el cuarto capítulo se recopilan las conclusiones de este trabajo y se proponen unos pasos futuros a seguir para profundizar en el desarrollo de este simulador.

Capítulo 1 : Introducción a la programación con limitación de recursos

1.1 Descripción del problema

La programación de proyectos trata de buscar una fecha de inicio y una fecha de fin para cada una de las actividades del proyecto. Sin embargo, la programación de estas actividades cuenta una serie de restricciones.

La primera restricción es la interrelación de las actividades. Las actividades de un proyecto están sujetas a relaciones de precedencia que deben ser respetadas (por ejemplo, el hecho de que una actividad del proyecto no pueda comenzar hasta que haya finalizado completamente otra). Además, estas actividades contarán con una duración (ya sea conocida o estimada), lo que, junto con la secuenciación anteriormente nombrada, nos permitirá obtener una programación completa del proyecto junto con su duración. En este espacio de trabajo se desarrollaron los primeros métodos para la programación de proyectos basados en grafos (CPM[1], PERT[2], ROY y GANTT). Estos métodos se consideraron adecuados para cuando la naturaleza del proyecto no requería un análisis de otros requerimientos, lo que rara vez se correspondía con la realidad del entorno de los mismos.

La segunda restricción es la limitación de recursos. Los métodos previamente mencionados no tuvieron en cuenta, en un principio, que estas actividades utilizan y/o consumen una serie de recursos que en la vida real pueden ser limitados debido a su coste, accesibilidad u ocupación (Villafáñez *et al.*, 2018). Por esta razón, aunque los métodos basados en grafos fueron muy útiles para una primera estimación, en su aplicación rara vez se cumplían sus programaciones, las cuales resultaban ser inviables en los proyectos reales (Araúzo, Pajares y Lopez-Paredes, 2010). Es por esto por lo que se hacía necesario elaborar métodos de resolución que tuvieran en cuenta también las restricciones de recursos que puedan existir. Algunos de estos métodos de resolución serán explicados posteriormente en la sección 1.3.

Una condición a tener en cuenta es el hecho de que pueden existir varios proyectos que compartan mismos recursos, produciéndose una sobreasignación de los mismos, lo que imposibilitaría la realización de los proyectos según su estimación inicial. Estos problemas se entienden como una variante del problema clásico de programación de actividades con restricción de recursos para un único proyecto. Al introducir varios proyectos la complejidad aumenta, disminuyendo por otra parte la probabilidad de encontrar una solución adecuada mediante los métodos de resolución exactos (ver sección 1.3.1).

Si dos actividades de diferentes proyectos comparten recursos y no existen recursos suficientes para programar las dos surge el problema de a qué proyecto dar preferencia. En la sección 1.2.2 se plantean dos enfoques a la hora de resolver este problema: centralizado y descentralizado.

[1] Critical Path Method – Método del camino crítico
[2] Program Evaluation and Review Techniques – Técnica de revisión y evaluación de proyectos

1.2 Caracterización de los problemas

1.2.1. **Dificultad de resolución**

En el momento en que a los problemas de programación de actividades les añadimos la restricción del número de recursos disponibles estos se convierten en problemas combinatorios caracterizados por la teoría computacional como problemas en tiempo no polinómico complejo (NP-Hard[3]). Los problemas NP-Hard son problemas en los cuales cuando encontramos una solución válida no podemos determinar si la solución encontrada es la óptima o no (Bartusch, Möhring y Radermacher, 1988). Por ende, es imposible determinar el tiempo requerido para encontrar la solución óptima.

La dificultad de resolución de estos problemas ha sido la principal motivación para la creación del simulador que presentaré más adelante en este trabajo. El simulador permitirá la obtención de diferentes soluciones viables a la programación de las actividades de los proyectos en cada uno de los problemas propuestos.

1.2.2. **Número de proyectos**

Como se ha mencionado anteriormente, el problema de programación de actividades con recursos limitados (RCPSP) se concibió inicialmente para actividades dentro de un solo proyecto. En estos problemas, al existir un único proyecto al cual dedicar los recursos, no había problema de asignación de recursos globales, pero sí de los recursos particulares del proyecto. Cuando introducimos problemas con dos o más proyectos la complejidad aumenta en el caso de que existan recursos globales que pueden ser asignados a diferentes actividades de diferentes proyectos. Así, el número de proyectos será relevante a la hora de caracterizar estos problemas y el enfoque con el que tratar los mismos.

En los problemas en los que existe más de un proyecto se definen dos enfoques atendiendo a la figura o ente sobre el cual recae la responsabilidad de decisión a la hora de programar las actividades:

- Centralizado, donde todas las actividades de los proyectos forman parte de un "macro proyecto". Este macro proyecto está formado por todas las actividades del proyecto y sus precedencias, así como de una actividad de duración cero que marca el inicio del macro proyecto y una actividad de duración cero que marca el final del mismo. En este enfoque la decisión recaería sobre un único ente, el director del programa de proyectos.
- Descentralizado, donde cada uno de los proyectos cuenta con libertad para programar sus actividades siempre y cuando respeten la disponibilidad global y particular de recursos (Villafañez *et al.*, 2014), recayendo la responsabilidad de decidir sobre el director de cada uno de los proyectos.

[3] Para información complementaria acerca de la complejidad computacional ver (Bartusch, Möhring and Radermacher, 1988)

1.2.3. **Recursos**

En los problemas de programación de actividades, la limitación de recursos es lo que los convierte en un problema combinatorio en el cual hay que decidir qué actividad se programa dado unos recursos limitados.

Estos recursos pueden ser por una parte renovables (una vez que la actividad que utiliza esos recursos finaliza esto se liberan y pueden ser asignados a otra actividad) o no renovables (una vez que una actividad utiliza esos recursos ya no pueden ser utilizados por otra actividad). Ejemplos de recursos renovables podrían ser las máquinas de producción de una industria, las cuales una vez terminan su labor, quedan disponibles para realizar otro proceso. Un ejemplo de recursos no renovables son las materias primas que trata esa misma máquina.

Por otra parte, los recursos utilizados pueden tener un carácter global, en el caso de que estos recursos puedan ser utilizados por actividades de diferentes proyectos, o un carácter local, en el caso de que los recursos solamente sean ocupados por actividades del propio proyecto. Un recuso global, por ejemplo, podría ser un director de proyecto el cual puede ser asignado a varios proyectos simultáneamente. Por otro lado, un recurso local podría ser las máquinas de producción especializadas que solo pueden ser utilizadas por un proyecto dentro de un programa.

Los recursos juegan un papel importante en el planteamiento de las actividades ya que, a menudo, estos recursos son costosos y la disponibilidad de los mismos es baja para realizar sus actividades. Si un recurso es demasiado caro, pero a la vez necesario para una empresa, esta tenderá a optimizar la utilización del recurso para aumentar todo lo posible los beneficios teniendo en cuenta el objetivo del proyecto (funciones objetivo).

1.2.4. **Función objetivo**

La función objetivo es aquel propósito que se quiere conseguir con la programación de las actividades. Estas funciones pueden ser muy diferentes dependiendo del entorno en el que se esté desarrollando el proyecto en cuestión. Según los objetivos de la empresa o institución que esté llevando a cabo el programa de proyectos, esta puede buscar objetivos como:

- Minimizar la duración del proyecto o proyectos, denominado en ingles TMS (*Total Makespan*), hace referencia a la distancia en tiempo desde que se inicia un proyecto o conjunto de proyectos hasta que terminan.
- Minimizar el retraso medio de los proyectos, denominado por sus siglas en ingles APD (*Average Project Delay*), refleja el número de unidades temporales medio que se retrasa un conjunto de proyectos respecto a su camino critico cuando no existe restricción de recursos

Otras funciones objetivo que pueden considerarse son minimizar el retraso total del proyecto, minimizar la penalización por retraso del proyecto o programa, minimizar el coste total del proyecto o maximizar la utilización de recursos entre muchas otras (Browning y Yassine, 2010).

No obstante, Villafáñez *et al.* (2019) señalan que la función objetivo más útil para una compañía genérica sería la minimización de la duración (TMS) ya que esta buscaría reducir la duración antes que, por ejemplo, todos los proyectos obtuvieran un retraso similar.

1.3 Métodos de resolución

Los métodos de resolución de los problemas de programación de actividades han sido ampliamente estudiados desde la década de los 50, cuando surgió la necesidad de realizar secuenciación de actividades en proyectos para obtener una programación viable. Estos métodos han ido evolucionando y perfeccionando a lo largo de los años, desde sus inicios (GANTT, PERT, CPM, ROY) hasta el día de hoy (heurísticas y metaheurísticas).

A día de hoy no se ha podido encontrar un método que sea mejor que los demás en todos y cada uno de los problemas planteados debido a que, dependiendo de las características del problema a abordar, un método puede ser más eficiente que otro.

A continuación, se realizará una clasificación y descripción de los principales métodos de resolución atendiendo a las técnicas utilizadas. Para ello me apoyaré en la publicación de Morillo, Moreno y Díaz (2014) en la cual realizan una revisión del estado del arte acerca de estos métodos.

1.3.1. **Métodos exactos**

Dentro de los métodos exactos se agrupan aquellos procedimientos que usan técnicas analíticas o matemáticas las cuales garantizan la convergencia a una solución óptima. No obstante, estos solo son útiles bajo unas condiciones determinadas como la continuidad del problema, un espacio de búsqueda de soluciones reducido y condiciones de linealidad. La razón por la cual existen diferentes métodos exactos para la búsqueda de soluciones a estos problemas es debido a que ninguno de ellos es lo suficientemente robusto. A continuación, se explican algunos de ellos debido a su importancia en la evolución de los métodos de resolución.

1.3.1.1 Búsqueda exhaustiva

La búsqueda exhaustiva es uno de los métodos más antiguos para solucionar los problemas de programación de actividades. Este método se basa en generar y evaluar todas las posibles soluciones a un problema dentro de un espacio de búsqueda, lo cual es una ventaja respecto a otros métodos ya que este puede ser aplicado a todos los problemas. En clara contraposición está el tiempo que puede llegar a tardar en encontrar la solución óptima debido a la explosión combinatoria generada cuando existen un gran número de actividades y proyectos. Debido a esto, el método de resolución de búsqueda exhaustiva solo se considera útil para los denominados problemas P, cuyo tiempo de cómputo crece de manera polinómica (Nievergelt, 2000). En la imagen (Figura 1.3-1) podemos observar todos los posibles resultados de una operación denominada triangulación de un hexágono. En este ejemplo tenemos una figura con seis nodos, en la programación de actividades podríamos encontrarnos con muchos más generando una explosión combinatoria que dificultaría la utilización de este método.

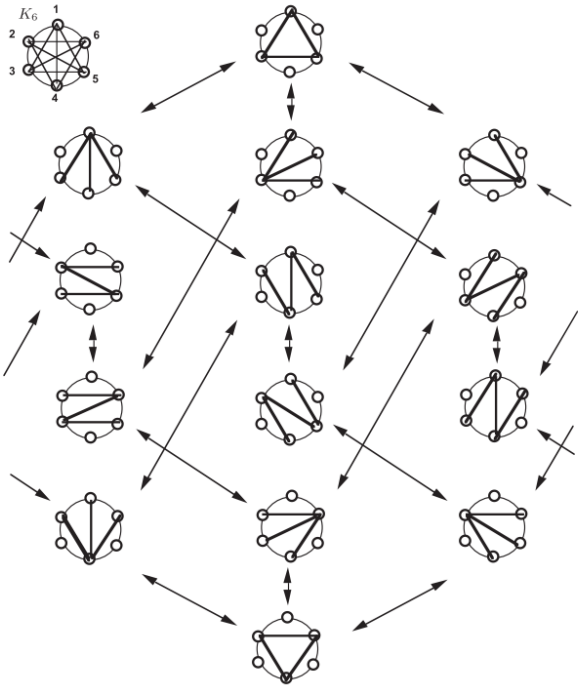

Figura 1.3-1: Espacio S de triangulación de un hexágono. Fuente: (Nievergelt, 2000)

1.3.1.2 Programación lineal

La programación lineal se basa en la hipótesis de que todas las funciones de un problema son combinaciones lineales de las variables de decisión del problema siendo la función objetivo el resultado que se desea maximizar o minimizar y las restricciones las limitaciones de recursos y las relaciones de precedencias. Estas restricciones son las que limitan el espacio de búsqueda de soluciones óptimas.

No obstante este método no se considera eficaz para un número grande de actividades (solo se considera eficaz en problemas de menos de 60 actividades (Valls, Ballestín y Quintanilla, 2005) debido a que el tiempo de búsqueda es demasiado grande incluso para los ordenadores modernos.

1.3.1.3 Ramificación y acotamiento

Este método se considera igual de robusto que la búsqueda exhaustiva pero más eficiente, ya que se basa en dividir el espacio factible donde puede encontrar la posible solución óptima. Dentro del espacio de posibles soluciones, este método determina dónde es más probable que se encuentre la posible solución mediante la aplicación de una regla determinada y limita su búsqueda a este espacio. Esto permite acotar el número total de posibles soluciones. En la Figura 1.3-2 podemos observar un ejemplo de ramificación y acotamiento a las soluciones.

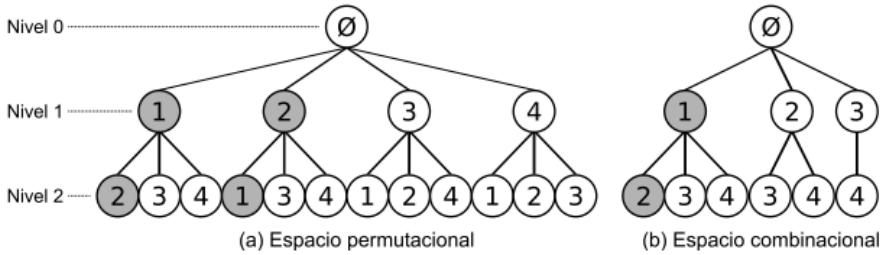

Figura 1.3-2: Árbol de soluciones de dos niveles. Fuente: (Ángel-Martínez *et al.*, 2019)

1.3.2. **Métodos heurísticos**

Los métodos heurísticos se caracterizan por obtener una solución lo suficientemente buena (válida) en un tiempo de computación razonable. Estos métodos surgieron de la necesidad de mejorar los métodos exactos, los cuales, si bien llegan a obtener un resultado óptimo, el tiempo que tardan en conseguirlo es demasiado grande debido a la explosión combinatoria que se genera al existir varios proyectos con numerosas actividades. En contrapartida, los métodos heurísticos son incapaces de asegurar que la solución que se ha obtenido sea el óptimo. No obstante, estos métodos han sido ampliamente utilizados y estudiados en la literatura debido a su gran utilidad en problemas en los que los métodos exactos no son viables.

Dado que no se puede conocer si el resultado obtenido por las heurísticas es el óptimo, existe la necesidad de elaborar un método para comparar como de buenas son estas heurísticas. Para ello se utilizan repositorios de problemas de programación de actividades los cuales proporcionan un soporte web para poder subir los resultados de la heurística y así poder compararlo con otras. En este trabajo se utilizará la librería de problemas MPSPLib (Homberger, 2007).

Los métodos heurísticos básicamente se pueden dividir en métodos heurísticos primitivos (técnicas de programación basada en reglas de prioridad, procedimientos de ramificación y acotamiento truncados, conceptos de arcos disyuntivos, búsqueda local, etc.) y métodos heurísticos basados en metaheurísticas (temple simulado, búsqueda tabú, algoritmos genéticos, programación evolutiva, colonia de hormigas, etc.). Dado la relevancia posterior en este trabajo del método heurístico basado en reglas de prioridad, este será el único procedimiento explicado dentro de los métodos heurísticos primitivos. Para mayor información sobre los métodos heurísticos primitivos se remite al lector a la publicación de Morillo, Moreno y Díaz (2014). También se explicará brevemente los métodos de resolución basados en metaheurísticas debido a su gran utilidad y buenos resultados.

1.3.2.1 Métodos heurísticos basados en reglas de prioridad

Estos métodos utilizan diferentes reglas de prioridad para decidir el orden en el que se realizarán las actividades y por lo tanto a cuál de ellas serán asignados los recursos en caso de que exista limitación en los mismos (Kolisch, 2012).

Los métodos heurísticos basados en reglas de prioridad están formados por un esquema de generación de programación (*Schedule Generation Scheme* – SGS) y una regla de prioridad. Por una parte, los esquemas de generación se pueden dividir en:

- Esquemas de generación en serie (S-SGS): consiste en una serie de fases en las cuales se programa una sola actividad. Dentro de cada fase existen tres conjuntos de actividades: actividades que ya han sido completadas, actividades elegibles cuyas predecesoras ya han terminado y actividades que no cumplen ni la primera ni la segunda condición (Kolisch, 2012). En cada fase, se selecciona una actividad del conjunto de actividades elegibles mediante una regla de prioridad y se programa en la fecha en la que todas sus predecesoras han terminado y existen recursos disponibles para realizarla. Esto provoca que cambiemos la actividad del grupo de elegibles a completadas, actualizando fase a fase los tres grupos de actividades.
- Esquemas de generación en paralelo (P-SGS): consiste en una serie de fases en cada cual se programan un conjunto de actividades en vez de una sola como en los esquemas de generación en serie. Estos esquemas están estrechamente relacionados con el tiempo. Cada fase dentro del esquema es una unidad temporal t en la cual existen cuatro conjuntos de actividades: actividades que ya han sido realizadas en tiempo t, actividades elegibles cuyas predecesoras han terminado en tiempo t y están disponibles los recursos necesarios para llevarlas a cabo, actividades que se están realizando en tiempo t y actividades que todavía no han sido programadas y no son elegibles (Kolisch, 2012). En cada fase se elige una serie de actividades del conjunto de actividades elegibles y se programan en ese periodo de tiempo. Al final de cada fase se aumenta la unidad temporal pasando a la siguiente fase y se actualizan los grupos de actividades.

El segundo elemento clave de estos métodos heurísticos es la regla de prioridad mediante la cual se decidirá qué actividades se programan antes y qué actividades, por el contrario, tienen que aplazarse al menos una unidad temporal. Existe un gran número de reglas de prioridad propuestas en la literatura. La regla de prioridad asigna un valor a cada una de las actividades que se encuentran dentro del grupo de elegibles y, una vez se asigne ese valor, se seleccionará el máximo o el mínimo valor para programar esa actividad. En el caso que ese valor sea igual, entra en juego los criterios de desempate, criterios mediante los cuales una actividad se programará antes que otra.

Según Kolisch (2012) las reglas de prioridad se pueden clasificar según la información procesada en reglas de prioridad de diagrama, de tiempo y de recursos. Las reglas de prioridad de diagrama atienden a la información relacionada con la secuenciación de actividades como puede ser el número de sucesoras totales, la pertenencia al camino crítico o el número de actividades sucesoras inmediatas. Por otro lado, las reglas de prioridad clasificadas como reglas de tiempo hacen referencia a las unidades temporales en las que se programan las actividades como por ejemplo la actividad que empiece o termine más tarde. Por último, las reglas relacionadas con los recursos sugieren un orden de actividades basado en cómo y en qué medida las actividades utilizan los recursos, por ejemplo, la utilización media de todos los recursos por parte de una actividad.

Por otra parte, en este mismo documento se clasifican las reglas de prioridad en estáticas, si para cualquier unidad de tiempo el valor asignado a una actividad se mantiene constante, o dinámicas, si este valor varía dependiendo del momento temporal de programación en el que nos encontremos.

1.3.2.2 Métodos heurísticos basados en técnicas metaheurísticas

Las técnicas metaheurísticas permiten por una parte la exploración de soluciones factibles para los problemas y por otra parte procedimientos para concentrar la búsqueda en el espacio más probable. A su vez, estas técnicas suelen incorporar herramientas para escapar de óptimos locales los cuales provocarían que el método se quedase atrapado en una solución lejana al optimo global. La gran versatilidad y los buenos resultados obtenidos por estos métodos hacen que sean una gran opción a

la hora de programar actividades con restricciones de recursos. A continuación, se resumirán los métodos más utilizados.

- Temple simulado (*Simulated Anneling*): este algoritmo empieza desde la generación de una solución aleatoria para la cual evalúa la función objetivo. A continuación, se genera una solución perteneciente a la vecindad de soluciones de la primera y se evalúa la nueva solución con la función objetivo. Si la segunda solución es mejor que la primera, esta se toma como buena y se repite el procedimiento hasta que se cumplan las iteraciones programadas o el criterio de parada. Se introduce un parámetro T (temperatura) el cual va disminuyendo cada iteración y sirve para explorar un rango más amplio de soluciones en las etapas iniciales del algoritmo escapando de posibles óptimos locales. En la Figura 1.3-3 se muestra como el parámetro T es relevante para no quedar atrapado en un mínimo local.

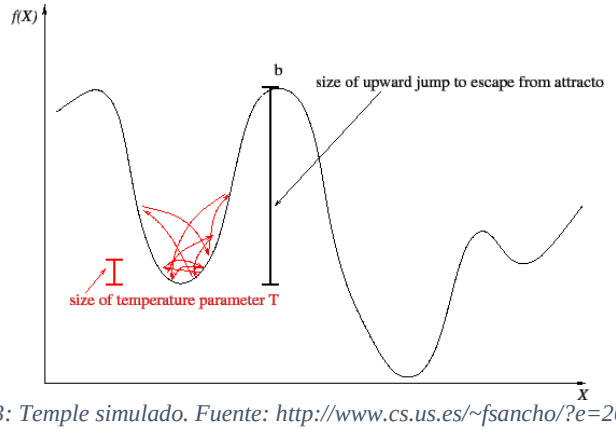

Figura 1.3-3: Temple simulado. Fuente: http://www.cs.us.es/~fsancho/?e=205

- Búsqueda tabú (*Tabú Search*): se genera una solución inicial y se determina las posibles soluciones adyacentes. Cada vez que se genera una solución y esta es peor que la solución anterior se incorpora a una lista de soluciones que no se pueden volver a generar (lista tabú). De esta manera nos aseguraremos de que no se vuelven a explorar soluciones ya valoradas y no quedarnos atrapados en óptimos locales.
- Algoritmos genéticos (*Genetic Algorithm*): los algoritmos genéticos se basan en el principio natural de mutación. Primero se generan una serie de soluciones mediante métodos heurísticos primitivos que se denominaran "padres". De esas soluciones se seleccionan aquellos que mejor solución han obtenido y se les asigna una probabilidad alta de obtener la misma solución en la siguiente iteración. A las soluciones que han obtenido peores resultados se les asigna una probabilidad menor. En la siguiente iteración a partir de las mejores soluciones (padres) se generan nuevas soluciones introduciendo "mutaciones", cambios aleatorios en la programación de actividades que pueden derivar en soluciones mejores. En el caso de que una mutación de una solución mejor que de la que deriva, esta se convierte en padre y se repite el proceso.
- Optimización de la colonia de hormigas (*Ant Colony Optimization*): este procedimiento se basa en la naturaleza de las hormigas las cuales buscan el camino más corto entre su colonia y el alimento dejando una serie de feromonas por el camino las cuales hacen que el resto de hormigas las sigan. Llevado al problema que nos atañe, este proceso parte de determinar una solución aleatoria dando un valor a la solución encontrada. Para la siguiente iteración, esta solución tendrá más probabilidades de ser obtenida que el resto, aunque se producirán ciertas variaciones. Si estas variaciones conducen a un resultado mejor el camino seguido recibe un valor. De esta manera los caminos menos transitados (peores soluciones) perderán importancia y los caminos más transitados (mejores soluciones) ganarán importancia (Figura 1.3-4).

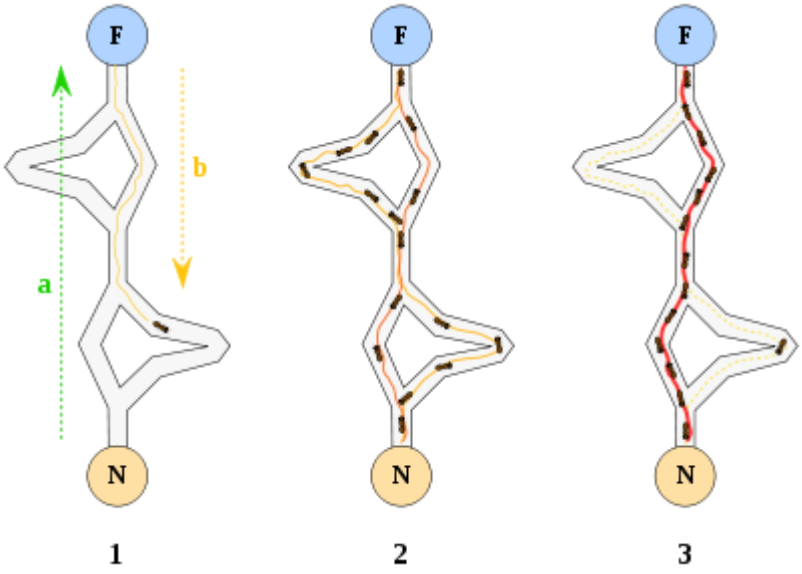

Figura 1.3-4: Representación colonia de hormigas. Fuente: Wikipedia.

1.4 Definición del problema

Como se ha visto en el apartado 1.2, los problemas de programación de actividades pueden caracterizarse de diferentes maneras atendiendo al número de proyectos con los que cuenta el problema, los recursos de los que dispone y los objetivos que se desean alcanzar dentro de esos problemas.

En este trabajo nos centraremos en los problemas de programación de actividades que cuentan con las siguientes características:

- Existencia de más de un proyecto. Así, los problemas a tratar serán los denominados RCMPSP, dejando de lado los problemas que solo cuentan con un proyecto (RCPSP). La motivación para tratar solamente los problemas con varios es proyectos es la dificultad incrementada que ofrecen estos problemas respecto a sus homólogos de tan solo un proyecto. Por otra parte, los problemas con varios proyectos se asemejan en mayor medida a los problemas actuales con los que cuentan las empresas.
- Actividades no interrumpibles. Plantearemos problemas formados por actividades las cuales una vez son iniciadas estas no pueden ser interrumpidas en ningún caso para comenzar otras actividades. Si bien en la realidad estas actividades pueden ser interrumpidas, no se aborda este problema debido al tratamiento del problema desde un enfoque de programación inicial previo al inicio de los proyectos.
- Actividades con relaciones de precedencia fin – inicio. Las actividades solo serán programadas una vez hayan finalizado todas las actividades que las preceden.
- Los recursos, tanto globales como locales, se asumen como renovables. Esta característica se asume por simplicidad de estudio, aun sabiendo que en la realidad se usan tanto recursos renovables como no renovables.

Con estas premisas y haciendo uso del trabajo planteado por Villafáñez *et al.* (2018), en la cual se propone el uso de una nomenclatura unificada para el planteamiento de los problemas RCMPSP, se plantea el problema.

Cada problema está formado por un número de proyectos I = {1, …, i, …, m} y un conjunto de recursos globales $K_I = \{1, \ldots, k, \ldots, kI\}$ los cuales pueden ser utilizados por todos los proyectos con la limitación de la capacidad total en el tiempo t del recurso $R_{kt} = R_k(t) = R_k$ que se mantiene constante a lo largo del proyecto.

Para cada proyecto i existen una serie de actividades $J_i = \{1, \ldots, j, \ldots, ni\}$ y un conjunto de recursos locales $L_i = \{1, \ldots, l, \ldots, k_{il}\}$. Estos recursos locales solo pueden ser asignados a las actividades del proyecto i y cuentan a su vez con la limitación de capacidad global en tiempo t del recurso $R_{ilt} = R_{il}(t) = R_{il}$ la cual se mantiene constante a lo largo del proyecto.

Cada actividad j del proyecto i tiene una duración d_{ij}, un requerimiento de recursos locales r_{ijl} y un requerimiento de recursos globales r_{ijk}. Estos requerimientos de recursos globales y particulares se mantienen constantes durante toda la duración de la actividad. A su vez, cada actividad también cuenta con un conjunto de predecesoras $P_{ij} = \{1, \ldots, j, \ldots, q\}$ que tienen que terminar antes de poder comenzar la actividad.

Dentro de cada proyecto, el conjunto de actividades J_i cuenta con una actividad ficticia de inicio de duración igual a cero ($d_{i1} = d_{ini} = 0$) y que no consume recursos ($r_{i1k} = r_{i1l} = r_{inil} = r_{inik} = 0$) y con una actividad ficticia de fin con las mismas características. La actividad de inicio marca el momento en el cual se pueden empezar a programar las actividades del proyecto I (ST_I) y la actividad de fin marca la finalización completa de todas las actividades y, por consiguiente, del propio proyecto (ED_i). La duración de la actividad (d_{ij}), el requerimiento de recursos globales y particulares por parte de las actividades (r_{ijl}, r_{ijk}) y el límite de los recursos particulares y globales (R_{il}, R_k) se asumen como números enteros no negativos.

La función objetivo de este problema es encontrar las fechas de inicio de las actividades (ST_{ij}) tal que se minimice el tiempo de realización del conjunto de proyectos (TMS) cumpliendo las restricciones de precedencia entre actividades y las limitaciones de recursos globales y particulares.

Con estas definiciones el problema puede plantearse formalmente como un modelo de programación lineal entera (Figura 1.4-1)

Objective Function:

$$\text{find } S_I = \{ST_{I\,start}, ST_{11}, \ldots, ST_{1n_1}, \ldots, ST_{ij}, \ldots, ST_{m1}, \ldots, ST_{mn_m}, ST_{I\,end}\}$$

such that

$$minimize\ TMS_I \qquad (\approx min.\ ST_{I\,end})\ (\approx min.\ (\max_{j \in J_i, i \in I}(ST_{ij} + d_{ij}))) \quad (5)$$

Subject to

$$ST_{ij} \geq ST_{iq} + d_{iq} \qquad \forall i \in I;\ j,q \in J_i\ /\ q \in P_{ij} \quad (6)$$

$$ST_I = \min_{i \in I} ST_i = \min_{i \in I} ST_{i\,start} = AD_I \qquad / \ AD_I \geq 0 \quad (7)$$

$$\sum_{j \in J_{it}} r_{ijlt} \leq R_{ilt} = R_{il}(t) = R_{il} = const. \qquad \forall l \in L_i;\ i \in I;\ t \in \{AD_I, \ldots, \overline{T}_I\} \quad (8)$$

$$\sum_{i \in I, j \in J_{it}} r_{ijkt} \leq R_{kt} = R_k(t) = R_k = const. \qquad \forall k \in K_I;\ t \in \{AD_I, \ldots, \overline{T}_I\} \quad (9)$$

Figura 1.4-1: Planteamiento formal del problema RCMPSP. Fuente: (Villafáñez *et al.*, 2018)

1.5 Revisión

En este apartado se realizará un resumen de la evolución en la programación de proyectos y carteras de proyectos desde sus inicios. Para ello esta sección se dividirá en cuatro apartados: En el apartado 1.5.1 se muestra los inicios de la programación de proyectos en el siglo XIX y los diferentes métodos desarrollados; en el apartado 1.5.2 se presentarán las herramientas basadas en grafos las cuales fueron de mucha utilidad y que, de hecho, a día de hoy se siguen utilizando; el apartado 1.5.3 trata los problemas de programación de proyectos con restricciones de recursos y su desarrollo y; por último, el apartado 1.5.4 versará sobre la programación de actividades dentro de carteras de proyectos.

1.5.1. **Inicios de la programación de proyectos**

La gestión de proyectos tal y como la conocemos hoy en día surge en la primera mitad del siglo XIX en el seno de la primera revolución industrial cuando se produce un auge de empresas que llevan a cabo procesos productivos. Estos procesos productivos requieren de una organización y gestión para su correcta ejecución por lo que se diseñaron métodos para poder programar su realización. La productividad de las empresas pasaba por ser capaces de generar sus productos con los recursos necesarios y en el mínimo tiempo posible lo que suponía una correcta programación de las actividades a desarrollar coordinando logística, trabajadores, maquinaria y materias primas.

Frederick W. Taylor, promotor de la organización científica del trabajo, fue el primero que propuso la división del trabajo en tareas y la especialización de las mismas como método para mejorar la eficiencia en el trabajo. Planificación, estandarización y control son los conceptos en los que se basaba la propuesta de Taylor para una mejora de los sistemas productivos y logísticos.

El primer método de gestión de tiempos que se conoce se denomina diagrama de Gantt (Figura 1.5-1), por su creador Henry Gantt el cual propuso a principios del siglo XX, entre muchas otras cosas, este diagrama temporal en el que representaba la secuencia de las tareas a realizar, su duración y las relaciones de precedencia entre las mismas. En este diagrama el eje horizontal representa el tiempo y en el eje vertical figuran las diferentes actividades en forma de barras, cuya longitud es proporcional a su duración y los extremos de la barra señalan el inicio y fin respectivamente de la actividad. Así mismo, utiliza flechas las cuales unen dos barras diferentes para denotar la relación de precedencia que existe entre ambas. En las versiones posteriores se introdujo nuevos datos acerca de las actividades como su holgura total, su holgura libre o fechas tardías y tempranas de inicio y fin de las actividades. Este diagrama fue la técnica más utilizada en la primera mitad del siglo XX y a día de hoy se sigue utilizando en la planificación de actividades debido a su carácter visual y analítico.

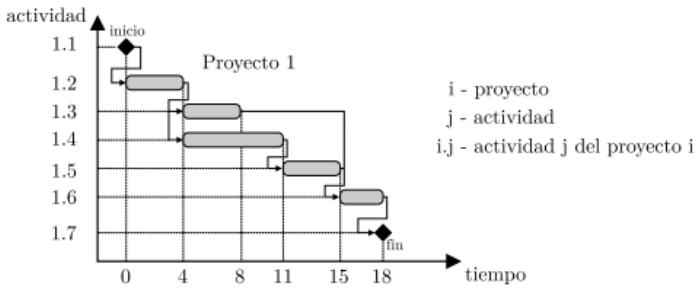

Figura 1.5-1: Diagrama de Gantt. Fuente: (Villafañez, 2014)

Otro recurso utilizado para la programación de actividades fue el diagrama de hitos (Figura 1.5-2), el cual es una versión más sencilla del diagrama de Gantt en la cual se representa el cronograma en función de unos hitos. Son diagramas muy sencillos en los cuales no existen relaciones de precedencia entre las actividades por lo que se utiliza en fases preliminares de planificación de los proyectos.

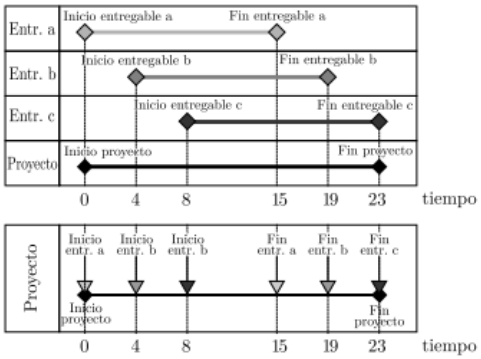

Figura 1.5-2: Diagrama de hitos. Fuente: (Villafañez, 2014)

Otras técnicas utilizadas que cayeron en desuso son la línea base de balance (LOB), el armonígrafo de Karol Adamiecki o la programación con líneas de flujo (Villafañez, 2014).

1.5.2. **Métodos basados en diagramas de red**

A partir del siglo XX es cuando se intensifica el estudio de las herramientas para la planificación de proyectos, más específicamente en los proyectos militares donde se empezaron a utilizar herramientas basadas en grafos o diagramas de red para coordinar grandes equipos. La falta de flexibilidad de los elementos utilizados hasta el momento (diagramas de Gantt) hace que estas nuevas herramientas tomen especial relevancia dado el ambiente de constante cambio e incertidumbre.

En este marco surgen en Estados Unidos las técnicas PERT[4] y CPM[5] y en Francia el método de los potenciales o ROY[6]. Estas técnicas perduran en el tiempo hasta el día de hoy debido a su gran utilidad y versatilidad. El motivo de la utilización de estos métodos es la búsqueda de un control de plazos efectivo, aunque, tiempo después, se incorporaron funcionalidades como control de costes, riesgos y nivelación de recursos a estas técnicas.

La principal diferencia entre los métodos PERT/CPM y ROY es el método de representación de los grafos. La representación de las relaciones en los métodos PERT y CPM se realiza mediante una representación de actividades en los arcos. En su representación existen diferentes nodos o sucesos a los cuales llegan las diferentes actividades. Por el contrario, en el método ROY son las propias actividades las que marcan los nodos y las relaciones de precedencia vienen señaladas por los arcos que relacionan unos nodos con otros (Figura 1.5-3).

[4] *Program Evaluation and Review Techniques* - Técnica de Revisión y Evaluación de Programas.
[5] *Critical Path Method* – Método del Camino Crítico.
[6] Nombre que se corresponde con el apellido de su creador, Bernard Roy.

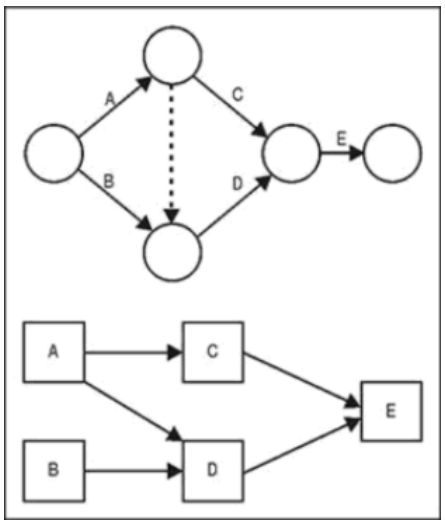

Figura 1.5-3: Diagramas PERT/CPM y ROY. Fuente: (Prado, 2015)

El método CPM y PERT utiliza el mismo tipo de gráfico con la diferencia del tratamiento de la duración de las actividades. Por un lado, el método CPM entiende la duración de las actividades como determinista mientras que el método PERT modela una distribución probabilística de la duración de las actividades.

1.5.3. **Programaciones de proyecto con recursos limitados (RCPSP)**

Los métodos introducidos en los apartados 1.5.1 y 1.5.2 (Gantt, PERT, CPM, ROY) fueron los métodos más utilizados hasta la década de los 50 dando excelente resultado en la programación de actividades dentro de un proyecto. Sin embargo, estos métodos solo eran capaces de resolver eficazmente los problemas más básicos (Villafañez, 2014), los cuales se caracterizaban por:

- Optimizar la duración del proyecto completo.
- Las actividades una vez iniciadas no podían ser interrumpidas[7].
- Las actividades tenían un consumo de recursos definido y constante a lo largo del tiempo.
- Los recursos eran renovables, bien limitados o ilimitados.
- Las actividades tenían relaciones de precedencia fin-inicio[8].

Fue a partir de los años 50 con el avance de la computación cuando se empezaron a aplicar técnicas de programación lineal para la resolución de estos problemas. A partir de ese momento se generalizó la nomenclatura RCPSP (*Resource-Constrained Project Scheduling Problem*) para referirse a aquellos problemas que cumplían las condiciones anteriormente mencionadas.

Los problemas RCPSP se convirtieron en problemas estándar y, aunque su aplicabilidad estaba muy limitada, sirvió de base para planteamientos de nuevos problemas modificando alguna de sus características como la función objetivo, los tipos de recursos, etc. Por ejemplo, Valls, Martí y Lino (1996) plantean que el problema RCPSP puede transformarse en un problema G-RCPSP (*Generalized Resource-Constrained Scheduling Problem*), problema que permite más relaciones de precedencia que las de fin – inicio, mediante la subdivisión de la actividad en tantas actividades como unidades de tiempo conlleve esa actividad unidas por relaciones de fin – inicio. Esto permitirá

[7] Denominadas en inglés como "*non-preemtable activities"*
[8] Hasta que no finalice una actividad no puede comenzar su sucesora.

crear relaciones de diferente tipo, así como plantear soluciones a problemas con actividades interrumpibles.

Los problemas RCPSP se catalogaron por Blazewicz, Lenstra y Kan (1983) como problemas NP-Hard (ver sección 1.2.1), problemas con una gran explosión combinatoria de lo cual provoca que el espacio de posibles soluciones sea extremadamente grande. Así, si añadimos complicaciones variando algunas características de los problemas como por ejemplo las relaciones de precedencia, estos problemas se volverán incluso más complicados de resolver.

La complejidad computacional planteada justifica la utilización de métodos de resolución heurísticos y metaheuristicos, métodos que son capaces de obtener resultados relativamente buenos en un tiempo razonable, aunque no se pueda determinar si se ha encontrado el resultado óptimo. El primer método heurístico aplicado a estos problemas fue propuesto por primera vez por Kelley (1959). El autor propone por primera vez ordenar las actividades por una secuencia y dentro de esa secuencia por un orden relacionado con la importancia de las actividades, como puede ser su holgura.

Pero no fue hasta aproximadamente la década de los 70 cuando se produce un auge de las heurísticas aplicadas a los problemas RCPSP. Al final de esta década se habían generado más de cien heurísticas, además de otros métodos exactos (Villafañez *et al.*, 2014), para resolver estos problemas que cada vez se consideraban más importantes para el desarrollo de los proyectos.
El interés, así como el estudio y el desarrollo de nuevas metodologías crece en la década de los 80 con la introducción de variantes al problema RCPSP, introducidas en esta misma sección. En la vida real no se producían las condiciones propuestas por el problema estándar RCPSP, por lo que se vio necesaria la introducción de variantes al problema estándar. Esto, junto con las metodologías desarrolladas hasta el momento fue un hito relevante ya que el problema pasaba de un marco teórico a un marco eminentemente práctico aplicable en la vida real.

En este periodo también se crearon los primeros repositorios de problemas para la comparación de los diferentes algoritmos. Patterson (1984) reunió en un mismo repositorio diferentes problemas de proyectos que habían sido usados para testear los algoritmos. El repositorio contaba con 110 problemas RCPSP cada cual estaba compuesto por 51 actividades. Este método de comparación de algoritmos fue usado por numerosos investigadores por lo que pronto se convirtió en un estándar para testear algoritmos heurísticos y exactos.

Desde entonces hasta el día de hoy se han ido perfeccionando las técnicas heurísticas y metaheurísticas aplicadas a todo tipo de variantes del problema original

1.5.4. **Programaciones de multiproyecto con recursos limitados (RCMPSP)**

El problema de programación de actividades con restricciones de recursos para carteras de proyectos es una especialización de problema RCPSP. En este caso se añade el término "multiproyecto" al conjunto de siglas representando la existencia de dos o más proyectos los cuales pueden compartir un recurso global a parte de contar con los recursos particulares o locales. Los problemas RCMPSP cumplen las mismas condiciones que los problemas RCPSP con la diferencia del tratamiento de la disponibilidad de recursos debido al posible carácter general de los mismos.

Asimismo, a partir de este problema, se han ido desarrollando nuevos problemas variando alguna de las características o tratamientos del problema y respetando la existencia de varios proyectos. Por ejemplo, según la responsabilidad de decisión a la hora de la asignación de recursos el problema podría ser CRCMPSP (*Centralized Resource-Constrained Multiproject Scheduling Problem)* si la

decisión recae sobre un ente global que controla todos los proyectos (Figura 1.5-4) o DRCMPSP (*Decentralized Resource-Constrained Multiproject Scheduling Problem)* si se otorga a los diferentes proyectos cierto libre albedrío a la hora de programar sus actividades (Figura 1.5-5).

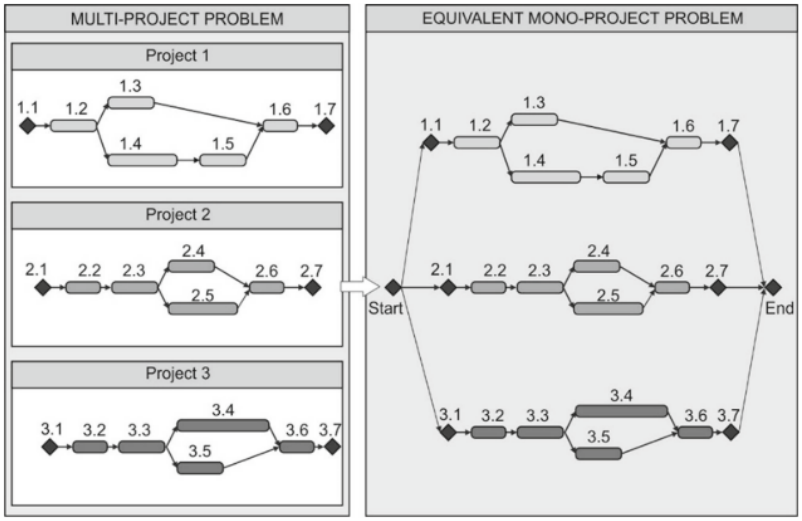

Figura 1.5-4: C - RCMPSP. Fuente:(Villafañez, 2014)

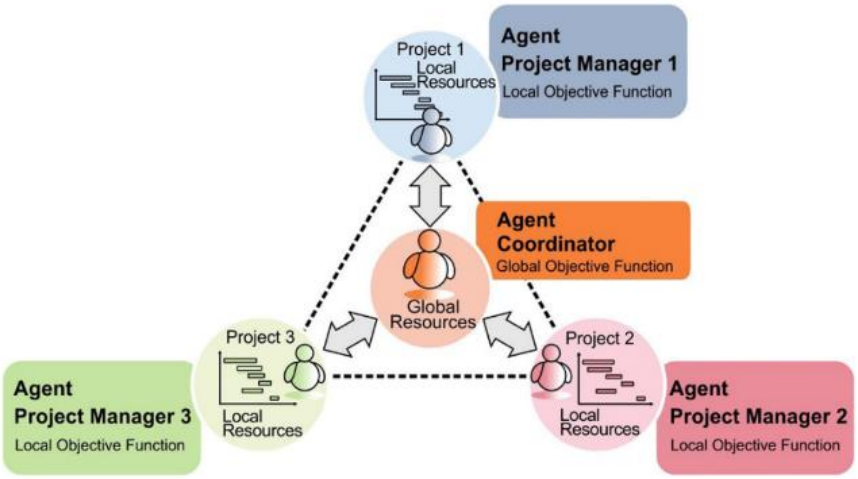

Figura 1.5-5: D - RCMPSP. Fuente:(Villafañez, 2014)

Como con los problemas RCPSP, para comprobar la eficacia de los diferentes algoritmos en estos problemas se crearon numerosas librerías de problemas entre las cuales destaca la librería MPSPLib (Homberger, 2007) por ser una de las más completas contando con 140 problemas con diferente número de proyectos y actividades[9].

[9] Esta librería cuenta con 140 problemas entre los cuales hay problemas de 2, 5, 10 y 20 proyectos con un número de actividades que puede ser 30, 90 y 120 actividades sin contar con actividades de inicio y fin.

Capítulo 2 SIMULADOR

En este capítulo presentaré el núcleo del presente trabajo fin de máster: un simulador de soluciones para problemas de programación de actividades con restricción de recursos. El simulador ha sido desarrollado en el entorno de Microsoft Excel mediante la utilización de macros utilizando el método de generación de esquemas de programación en paralelo (P-SGS). Esto conlleva que, para cada unidad de tiempo, se reproduzca todo el procedimiento lineal de programación estableciendo tanto el inicio como el final de una o varias actividades en cada fase. Así, una vez terminadas todas las fases (días) de los que consta cada problema, se obtendrá una programación completa de todas las actividades de cada proyecto habiendo respetado las restricciones de precedencia y recursos.

El capítulo se ha dividido como sigue: en el apartado 2.1 hago una introducción y breve explicación de la librería MPSPLib, librería utilizada para conformar el simulador y hacer la comprobación de las heurísticas implementadas; en el apartado 2.2 muestro todas las partes de las que consta el simulador (pestañas, tablas y gráficos); el apartado 2.3 hace referencia a los procedimientos externos que el usuario puede realizar y cómo estos activan una serie de instrucciones que hacen que el simulador despliegue todas sus aplicaciones; en el apartado 2.4 se realiza un análisis de la herramienta teniendo en cuenta sus ventajas y limitaciones y, por último, en el apartado 2.5 se realiza una comparación con otras herramientas actuales del mercado utilizadas para la programación de actividades con recursos limitados.

Con el fin de establecer un límite para el simulador, este se ha diseñado para utilizar como fuente problemas de la librería MPSPLib: problemas de un máximo de 20 proyectos, 120 actividades y 4 recursos (entre recursos globales y particulares). No obstante, debido a la flexibilidad del código programado, estos máximos podrían aumentar en un futuro mediante una ligera modificación del código en cuestión.

2.1 Problemas de la librería MPSPLib

Para el diseño de este simulador se ha tomado como referencia los problemas de la librería MPSPLib, un conjunto de 140 problemas RCMPSP cada cual es un portfolio de proyectos. El número de proyectos de los que consta cada problema varía entre 2, 5, 10 y 20. Por otro lado, las actividades de las que consta cada proyecto también varían entre 30, 90 y 120 actividades. Con la combinación de número de proyectos y el número de actividades se generan los 140 problemas los cuales tienen las siguientes características:

- Todos los problemas constan de cuatro recursos (R1, R2, R3 y R4) los cuales pueden ser globales si pueden ser utilizados por todos los proyectos o particulares si solo pueden ser utilizados por un proyecto.
- En todos los problemas existe al menos un recurso que es de carácter global, ya que si todos fuesen particulares el problema podría tratarse como varios problemas RCPSP independientes.
- Todos los recursos son de tipo renovable y su disponibilidad máxima se mantiene constante a lo largo del tiempo.

Cada problema está denominado de manera diferente. En primer lugar, figuran las letras "*mp*" comunes a todos los proyectos y que hacen referencia a la existencia de varios proyectos (multi-

project). En segundo lugar, se representa el número de actividades de las que consta cada proyecto que forma el problema mediante los grupos de caracteres "*j30*", "*j90*" o "*j120*"[10]. En tercer lugar, figura el número de proyectos por los que está compuesto el problema con los grupos de caracteres "*a2*", "*a5*", "*a10*" o "*a20*". En cuarto lugar, se encuentra las letras "*nr*" seguidas de un ordinal cuya misión es diferenciar los problemas que cuentan con las mismas características anteriores. Por último, en quinto lugar, puede figurar las letras "*AgentCopp*" las cuales indican que ese problema está compuesto por cuatro recursos todos ellos globales (Villafañez, 2014).

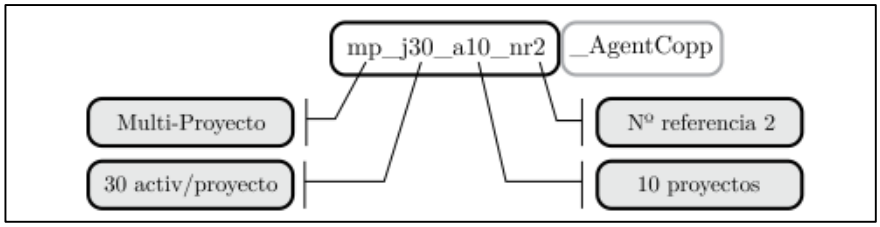

Figura 2.1-1: Codificación de los problemas MPSPLib. Fuente: (Villafañez, 2014).

Para cada problema existe un fichero con terminación .xml en el cual se indican las características del problema: nombre del problema (Figura 2.1-2I), nombre de cada uno de los proyectos (Figura 2.1-2II), fecha de inicio de cada uno de los proyectos (Figura 2.1-2III) y limitación de cada recurso global (Figura 2.1-2IV)[11].

```
<mp-list>
  <mp>
I   <name>mp_j30_a2_nr1</name>
    <project-list>
      <project>
II      <filename>KolischInstanzen/j30/j3015_5.sm</filename>
III     <start>0</start>
      </project>
      <project>
        <filename>KolischInstanzen/j30/j3015_5.sm</filename>
        <start>0</start>
      </project>
    </project-list>
    <resources>
IV      <resource>32</resource>
        <resource>62</resource>
        <resource>0</resource>
        <resource>0</resource>
    </resources>
  </mp>
</mp-list>
```

Figura 2.1-2: Ejemplo del fichero .xml.

[10] Nótese que dentro de un mismo problema los proyectos están formados por el mismo número de actividades.

[11] Si este número es igual a cero significa que el recurso en ese problema es de carácter local

2.2 Partes del simulador

El simulador realizado consta principalmente de tres partes claramente diferenciadas relacionadas cada una con una pestaña del libro de trabajo:

- Tratamiento de datos (pestaña nº 1): Esta es la hoja principal del simulador. En ella se encuentran todos los datos necesarios para trabajar con el problema de programación de actividades (tablas de recursos globales y particulares, tablas de información general de cada proyecto, tablas de precedencias y tablas de recursos y duración) así como los principales gráficos (gráficos de utilización de recursos y gráficos Gantt).
- Recopilación de resultados (pestaña nº2): En esta hoja figurarán todos los resultados obtenidos durante las sucesivas simulaciones, así como los datos resumidos de los mejores resultados obtenidos.
- Mejor resultado obtenido (pestaña nº3): esta pestaña será una copia de la primera, pero en este caso figurará el mejor resultado obtenido atendiendo a la duración total del problema (mínimo *total makespan*, TMS o duración total). También existirá la opción de obtener más de un resultado y por consiguiente esta pestaña estará replicada tantas veces como resultados encuentre.

2.2.1. **Tratamiento de datos (Hoja 1 – "Datos Excel")**

En esta primera hoja figuran todos los datos necesarios para el cálculo de soluciones del programa. El simulador utiliza estos datos en sus procedimientos para, secuencialmente, realizar todas las acciones necesarias. La información está dispuesta en forma de tablas para una ejecución más rápida y eficiente. Así, para cada proyecto perteneciente al problema a tratar existirán las siguientes tablas:

- Tabla de información general del proyecto "*InfoProyecto*". En esta tabla figurarán el nombre del proyecto, el número del problema (*pronr*)[12], el número de actividades de las que está formado el proyecto (*#jobs*), la fecha en la que el proyecto se inicia (*rel.date*), la fecha en la que el proyecto terminaría en el caso de que no hubiese restricción de recursos (*duedate*), los costes por retraso (*tardcost*)[13] y la duración de calculada mediante el Método Roy (*MPM-Time*).

Project 1					
pronr.	#jobs	rel.date	duedate	tardcost	MPM-Time
1	120	0	99	97	99

Figura 2.2-1: Tabla "InfoProyecto"

[12] Nótese que, al tratar los proyectos como individuales, todos los proyectos figurarán como problema 1.
[13] El dato de los costes por retraso se utiliza para calcular las penalizaciones. En este caso no se tendrá en cuenta para la búsqueda de soluciones.

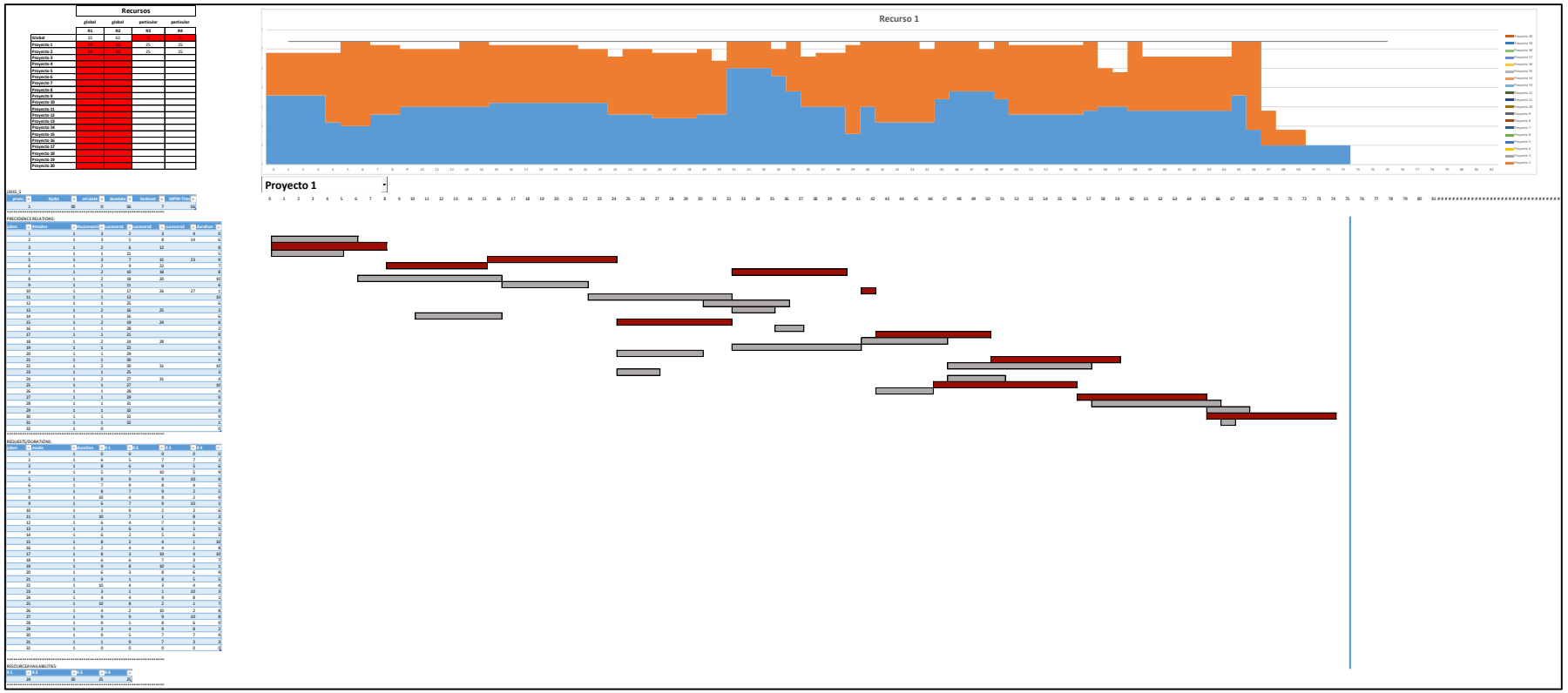

Figura 2.2-2: Vista general del simulador.

- Tabla de precedencias de las actividades "Precedencias". Los datos sobre las relaciones de precedencia que han de respetarse entre las actividades figuran en esta tabla. La presente tabla consta del número de cada actividad (*jobnr.*), el modo en el que esta implementado esta actividad[14] *(#modes*), el número de sucesoras que tiene esa actividad (*#successors*) y cuál es su número de actividad (*successor1, successor2, successor3*) y finalmente la duración de la actividad (*duration*).

PRECEDENCE RELATIONS:

jobnr.	#modes	#successors	successor1	successor2	successor3	duration
1	1	3	2	3	4	0
2	1	2	9	20		5
3	1	1	64			10
4	1	3	5	11	13	9
5	1	1	6			4

Figura 2.2-3: Tabla "Precedencias"

- Tabla de consumo de recursos "RecursosDuración". En la siguiente tabla figurará el requerimiento de cada recurso por parte de la actividad (*R1, R2, R3, R4*) así como el número de la actividad (*jobnr.*), el modo de ejecución (*mode*) y la duración de la actividad (*duration*)[15].

REQUESTS/DURATIONS:

jobnr.	mode	duration	R 1	R 2	R 3	R 4
1	1	0	0	0	0	0
2	1	5	0	10	2	0
3	1	10	1	0	0	0
4	1	9	6	4	2	9
5	1	4	5	0	10	9

Figura 2.2-4: Tabla "RecursosDuración"

- Tabla de disponibilidad de recursos "Disponibilidad". La disponibilidad de recursos parciales por parte de cada proyecto quedará reflejada en esta tabla constando en la misma cuatro columnas correspondientes a la disponibilidad de cada recurso (*R1, R2, R3, R4*). En el caso de que un recurso de los cuatro fuese global, este dato no aplicaría para el problema.

RESOURCEAVAILABILITIES:

R 1	R 2	R 3	R 4
21	26	25	25

Figura 2.2-5: Tabla "Disponibilidad"

Aparte de las tablas anteriormente mencionadas existe una extensión a la tabla denominada "Precedencias" en la que se incluyen diferentes datos relativos a la programación. Esta parte de la tabla de carácter general se mantiene oculta en el simulador ya que no se ha considerado de utilidad el ser mostrada a la hora de obtener una vista más amigable para el usuario. En esta parte de la tabla se encuentran las columnas que designan características de cada actividad como pueden ser el valor a la hora de ordenar las actividades para la heurística (*valor*), la fecha de inicio temprana (*FechaInicio*) y fecha de fin tardía (*FechaFinTardia*) de cada actividad, fecha de finalización de la actividad (*FechaFin*), su estado (*candidata*) que puede ser sin iniciar si está en blanco,

[14] El modo en el que se desarrolla una actividad puede marcar el carácter de los recursos (renovables/no renovables). En el presente trabajo se valorará únicamente un modo de ejecución de la actividad, recursos renovables.

[15] Por motivos de simplicidad a la hora de programar se repite este dato ya presente en la tabla "Precedencias".

"Programada", "No recursos/retrasada" o "Finalizada", el número de unidades que se ha retrasado la actividad (*Retraso*) y su holgura total (*HolguraTotal*).

PRECEDENCE RELATIONS:

jobnr.	Fecha inicio	Fecha fin	candidata	valor	Retraso	FechaFinTardia	HolguraTotal
1	0	0				0	0
2	4	11	No Recursos	0,208531111	5	8	4
3	4	7	No Recursos	0,436157167	5	4	0
4	0	5	Finalizada			0	0
5	5	9	Programada	1,386068612		5	0

Figura 2.2-6: Extensión de la tabla "Precedencias"

El último conjunto de tablas corresponde al tratamiento de los recursos. La primera de esas tablas no es más que un resumen de la disponibilidad de recursos globales y particulares del problema, teniendo que señalar por parte del usuario que tipo de recursos es. La segunda tabla hace referencia a la cantidad de recursos que tenemos disponibles para asignar a las actividades en el tiempo t y, por último, la tercera tabla nos señala la cantidad de recursos que se han quedado sin asignar después de programar las actividades en t. La segunda y tercera tabla se encuentran ocultas en la visión principal del simulador.

	Recursos			
	global	global	particular	particular
	R1	R2	R3	R4
Global	45	42		
Proyecto 1	30	28	26	33
Proyecto 2	30	28	26	33
Proyecto 3	30	28	26	33
Proyecto 4	30	28	26	33
Proyecto 5	30	28	26	33
Proyecto 6	30	28	26	33
Proyecto 7	30	28	26	33
Proyecto 8	30	28	26	33
Proyecto 9	30	28	26	33
Proyecto 10	30	28	26	33

Figura 2.2-7: Recursos globales (fila "Global" con la casilla en blanco) y recursos particulares (fila de cada proyecto con la casilla del recurso correspondiente en blanco)

Disponibilidad de recursos antes de prg			
R1	R2	R3	R4
43	40		
30	28	26	33
30	28	26	33
30	28	26	33
28	26	20	31
30	28	26	33
30	28	26	33
30	28	26	33
30	28	26	33
30	28	26	33
30	28	26	33

Figura 2.2-8: Disponibilidad de recursos antes de programar las actividades

Disponibilidad de recursos después de pgr			
R1	**R2**	**R3**	**R4**
1	18		
20	23	19	26
30	28	26	33
20	23	19	26
18	21	13	24
30	28	26	33
30	28	26	33
30	28	26	33
18	21	13	24
30	28	26	33
30	28	26	33

Figura 2.2-9: Disponibilidad de recursos después de programar las actividades

Para una sencilla visualización de los resultados se han añadido dos representaciones gráficas: un diagrama de Gantt en el cual figuran en forma de rectángulos todas las actividades con su correspondiente duración y una representación gráfica de la utilización de recursos globales por parte de cada proyecto.

- Gantt: descrito en este mismo documento (sección 1.5.1) el diagrama de Gantt es una representación gráfica de la programación de actividades en la que, por cada actividad, se genera un rectángulo desde la fecha de inicio hasta la fecha de finalización. En este simulador se ha querido representar a su vez el estado de propia actividad (sin programar o programada) mediante un código de colores en el que el color azul indica que todavía no se ha definido una fecha de comienzo de la actividad, el color gris significa que se ha fijado la actividad con una fecha de inicio y una de fin, el color rojo significa que la actividad es crítica para la duración total del proyecto y además no se encuentra programada y el color rojo oscuro para aquellas actividades que se han programado siendo críticas. También se ha incorporado una barra de progreso que indica por qué fase t (día) se llega la simulación. A medida que avanzan los días, la barra de progreso también avanza, desplazando en el diagrama de Gantt a aquellas actividades que no han sido programadas por falta de disponibilidad de recursos.

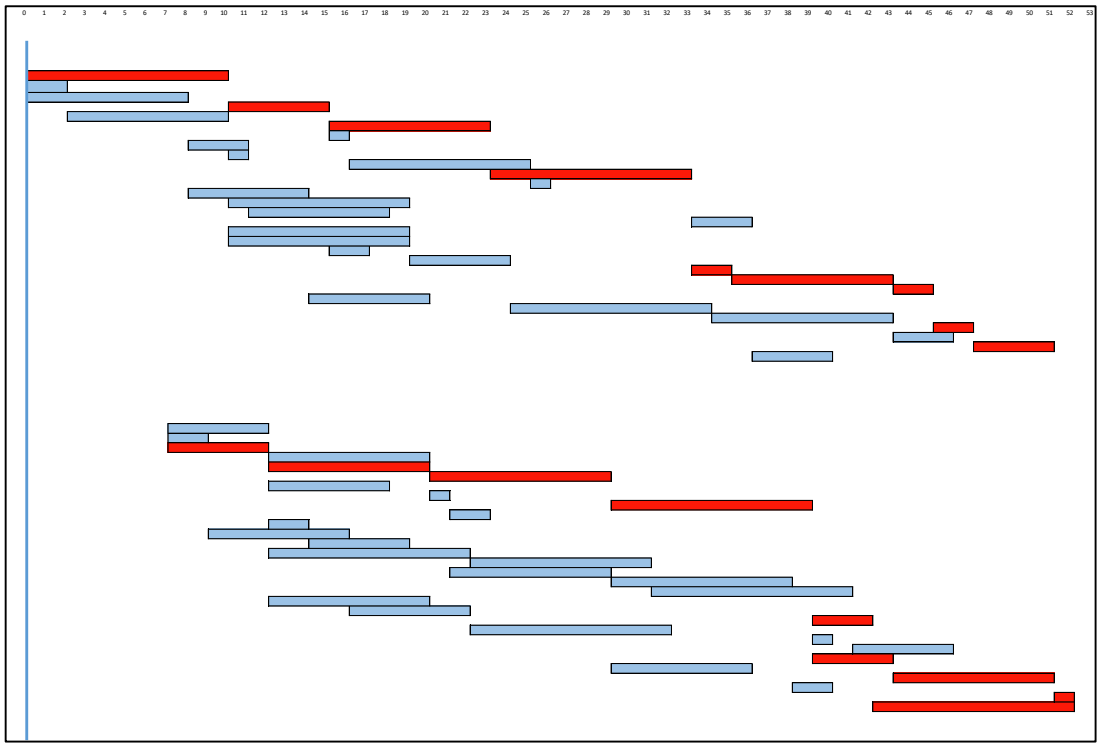

Figura 2.2-10: Diagrama Gantt al inicio del problema en el que se muestran las actividades criticas (rojo) y las actividades no criticas (azul) sin programar.

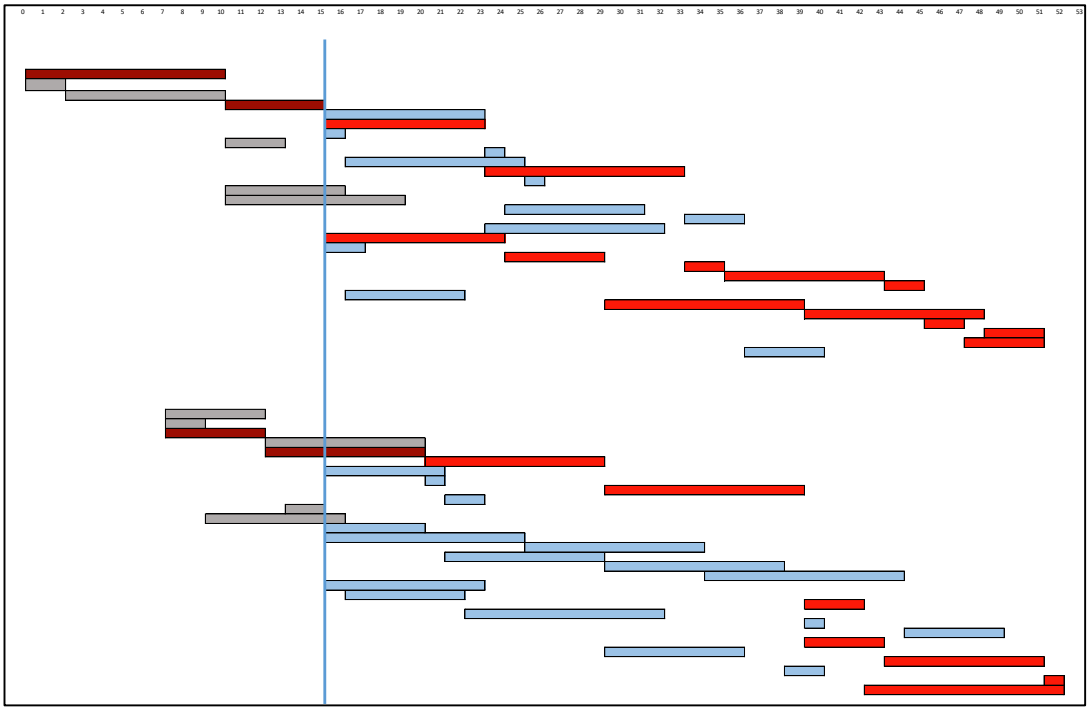

Figura 2.2-11: Diagrama Gantt después de 15 días simulados en el que se muestran las actividades criticas (rojo), las actividades no criticas (azul), las actividades programadas criticas (granate) y las actividades programadas no criticas (gris).

- Gráficos de recursos. Estos gráficos representan qué cantidad de recursos consume cada proyecto. Gracias a este gráfico podemos ver si existe o no sobreasignación en algún

periodo de tiempo, así como los periodos de tiempo en los cuales el recurso está menos demandado. Este gráfico se considera de utilidad en situaciones en las cuales el usuario tiene que gestionar los recursos, ya sea dedicandolos a otras tareas cuando están desocupados o aumentando la capacidad de los recursos (si fuera posible) cuando estos se encuentan más limitados. El gráfico se puede mostrar como desee el usuario, con la barra de opciones puede señalar el recurso que desea visualizar y mediante la lista desplegable puede seleccionar el proyecto del cual quiere ver los recursos en el caso de que este recurso sea particular de cada proyecto (ver Figura 2.2-14). Nótese que si se selecciona un recurso global y se cambia de proyecto no cambia el grafico, ya que solo representa los recursos globales de toda la cartera (ver Figura 2.2-13).

2.2.2. **Recopilación de resultados (Hoja 2 – "Resultados")**

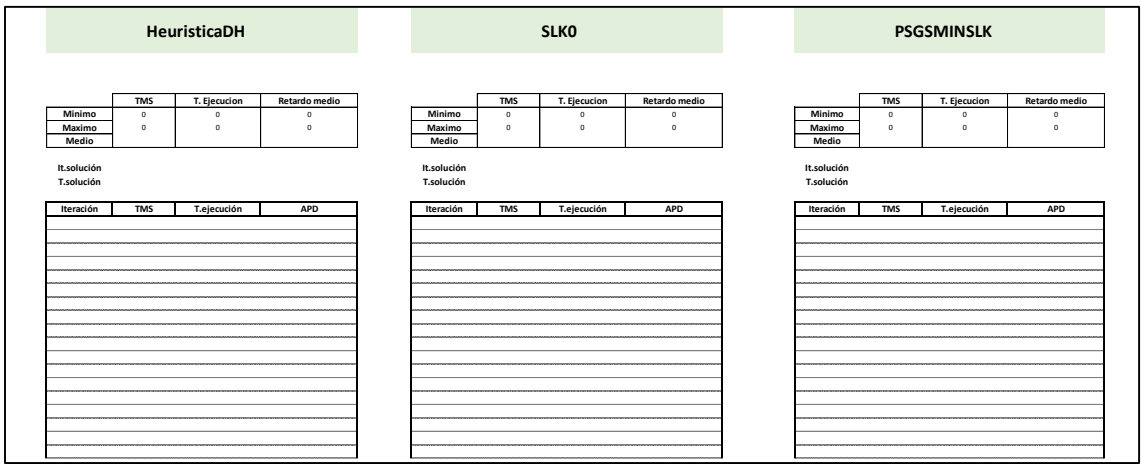

Figura 2.2-12: Vista parcial de la segunda hoja del simulador.

La segunda hoja de este simulador está dedicada al resumen de los resultados obtenidos por la simulación. En el uso de las heurísticas y metaheurísticas es habitual la repetición del proceso de obtención de resultados para poder valorar la mejor opción obtenida. Por este motivo se presenta en esta parte del simulador una tabla general para cada regla de prioridad implementada en el simulador (seis en total[16]) en la que figuran los resultados de cada simulación.

Con el fin de acotar la información obtenida, esta tabla contiene cuatro columnas correspondientes a:

- Iteración. Esta cifra identifica el número de simulación en la que se han obtenido los resultados que figuran a continuación. Es de especial relevancia en aquellas técnicas metaheurísticas que buscan llegar al mejor resultado posible en el menor número de iteraciones.

[16] Las seis heurísticas implementadas en este simulador serán explicadas en la sección 2.3.

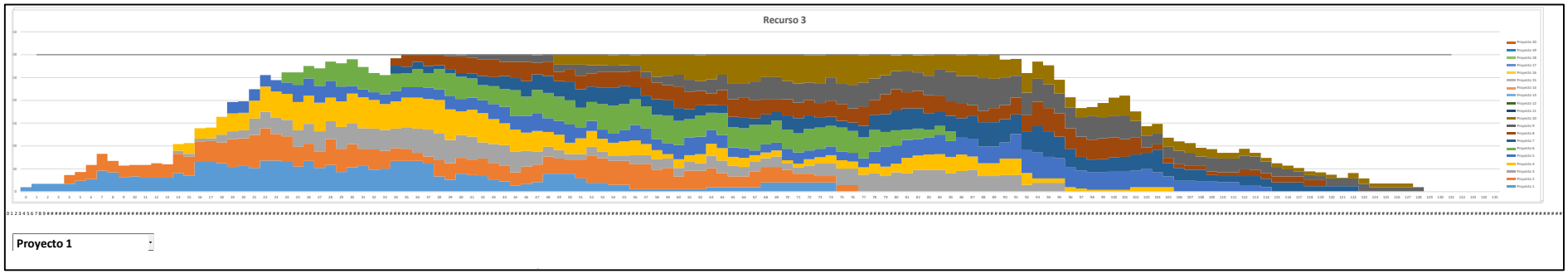

Figura 2.2-13: Gráfico de utilización de recurso global R3. Problema 22 de la librería MPSPLib con diez proyectos.

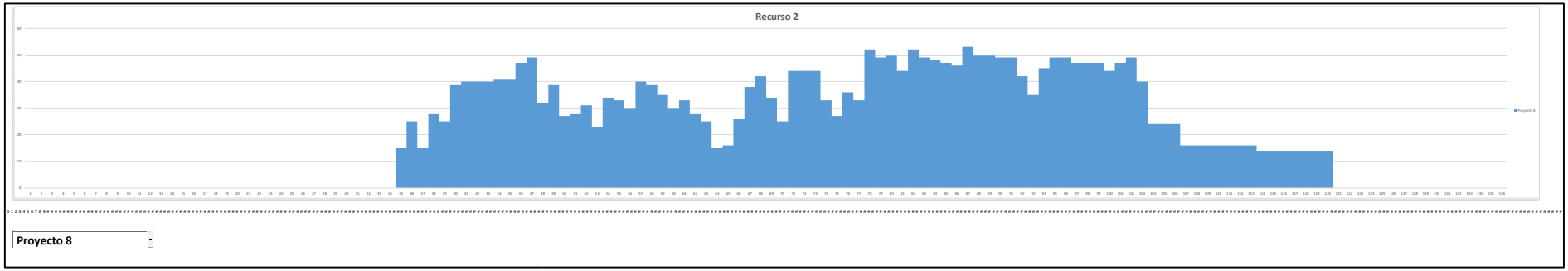

Figura 2.2-14: Gráfico de utilización de recurso local R2. Proyecto 8 del problema 22 de la librería MPSPLib.

- TMS (*Total Makespan*) o duración total de la cartera de proyectos. La minimización de esta función objetivo es una de las más estudiadas y equivale a la fecha de finalización más tardía entre todos los proyectos menos la fecha de inicio más temprana. En resumen, TMS sería el número de unidades temporales que transcurren desde el inicio del primer proyecto hasta la finalización del último proyecto dentro de una cartera.
- Tiempo de ejecución. El tiempo de ejecución es el número de segundos que tarda el simulador en obtener una simulación completa. Este número se considera relevante ya que muestra la velocidad con la que se ejecuta la simulación. Cuanto mayor es la rapidez de simulación, mayor es el número de resultados obtenidos y, por consiguiente, la probabilidad de obtener un mejor resultado en el caso de utilizar técnicas heurísticas es mayor.
- APD (*Average Project Delay*) o retraso medio de finalización de los proyectos. Este factor es la media entre los retrasos de cada proyecto, entendiendo retraso como el número de unidades temporales empleadas a mayores de la duración inicial del camino crítico para programar todas las actividades de ese proyecto.

Iteración	TMS	T.ejecución	APD
1	193	74,046875	143,3
2	196	69,53515625	147
3	194	69,73828125	144
4	198	74,25390625	132
5	195	71,44140625	140,3
6	195	68,39453125	148,7
7	196	68,83203125	142,9
8	192	70,95703125	137,3
9	197	68,63671875	138,5
10	199	67,3671875	132,8
11	195	70,57421875	138,5
12	195	68,75390625	144,2
13	193	68,14453125	143,1
14	193	69,0234375	132,2
15	196	70,91796875	141,3
16	194	68,59765625	141,8
17	195	68,9921875	140,5
18	195	69,30078125	136,3
19	197	69,296875	148,2
20	195	69,4140625	153,9
21	197	69,546875	132,1
22	197	68,5078125	145,7
23	194	68,6328125	142,9
24	194	69,37109375	139
25	199	69,30078125	151,3

Figura 2.2-15: 25 resultados de un problema con su información correspondiente.

Aparte de esta tabla en la que figuran todos los resultados, existe en la misma hoja una tabla que resume los mejores resultados obtenidos de entre todas las simulaciones, esto es, el menor y mayor valor obtenido de entre todas las simulaciones, así como su promedio para cada una de las tres salidas de cada simulación (TMS, Tiempo de ejecución y APD). A su vez, se han añadido dos datos que hacen referencia al número de iteración en el que se ha encontrado la mejor solución y cuanto ha tardado en segundos en conseguirla.

	TMS	T. Ejecución	APD
Mínimo	192	67,3671875	132
Máximo	199	74,25390625	153,9
Medio	195,36	69,663125	141,512
It.solución	8		
T.solución	567,1992188		

Figura 2.2-16: Tabla resumen de los resultados obtenidos.

Estas tablas se encuentran replicadas dentro de la misma hoja seis veces, una para cada regla de prioridad implementada en este simulador.

2.2.3. **Mejor resultado obtenido (Hoja 3 en adelante)**

Esta última parte del simulador hace referencia a las mejores soluciones encontradas, entendiendo como mejor solución aquella que minimiza el tiempo total de ejecución de la cartera (TMS).

Esta última hoja se denomina "Mejor resultado. TMS =" seguido del tiempo mínimo de programación encontrado (TMS). Así, cuando el simulador encuentre la mejor solución, creará una copia de la primera hoja "Datos Excel" con los datos de la mejor solución encontrada. Asimismo, en el caso de que el usuario haya establecido un número de TMS a partir del cual quiere obtener los datos de esa simulación, se crearan tantas hojas como resultados encuentre menores que el tiempo total establecido. Esta funcionalidad se explicará más en detalle en el apartado 0.

2.3 Funcionamiento

El funcionamiento del presente simulador se basa en la programación en lenguaje Visual Basic, una programación orientada a objetos en la cual se leen secuencialmente las líneas de código programadas. Estas líneas de código forman parte de diferentes "macros" las cuales ejecutan las funcionalidades mediante las cuales se obtienen los resultados de programación de las actividades.

En la sección 2.2 se han introducido las tres partes del simulador, las cuales conforman la estructura de tablas y gráficos que darán lugar a las soluciones finales. Sin embargo, este apartado se centra en explicar los procesos que lleva a cabo el simulador, así como las diferentes opciones de las que puede hacer uso la persona que interactúa con el mismo.

Con el fin de simplificar la interacción usuario-simulador se ha reducido las pestañas de la cinta de opciones del archivo Excel (la cual habitualmente cuenta con diferentes opciones) a una única opción, el simulador. Esta pestaña se denomina "RCMPSP" y dentro de ella se encuentran todas las opciones que el usuario puede realizar y que a continuación se presentan (Figura 2.3-1, Figura 2.3-2).

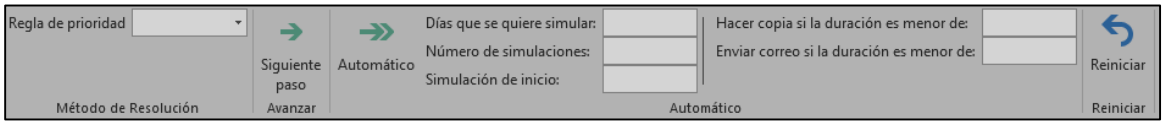

Figura 2.3-1: Cinta de opciones "RCMPSP"

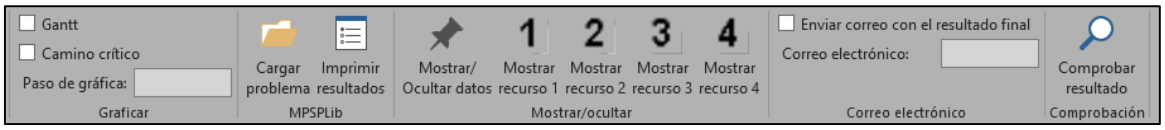

Figura 2.3-2: Cinta de opciones (2) "RCMPSP"

2.3.1. **Relación Simulador – Librería MPSPLib**

Como se ha comentado en la sección **¡Error! No se encuentra el origen de la referencia.**, este s imulador utiliza los problemas de la librería MPSPLib para su funcionamiento. Debido a esto, se considera necesario insertar un grupo de opciones dedicado a esta librería.

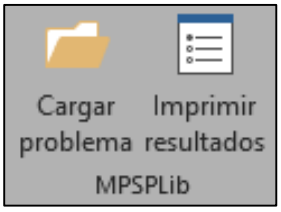

Figura 2.3-3: Opciones relacionadas con la librería MPSPLib.

Dentro de este grupo podemos encontrar en primer lugar la opción de cargar un problema de la librería para su estudio en el simulador. Al seleccionar esta opción se nos abrirá una ventana en la cual se pide que el usuario guarde el archivo pudiendo elegir tanto el nombre como el destino donde quiere guardarlo (Figura 2.3-4). Una vez guardado, el programa pedirá al usuario que introduzca el número del problema de la librería MPSPLib que desee utilizar en el simulador (Figura 2.3-5).

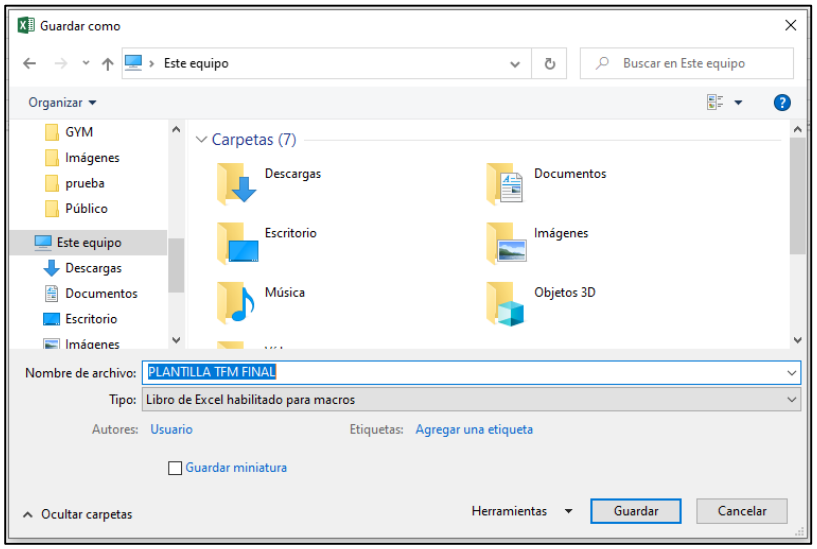

Figura 2.3-4: Paso 1 para introducir un nuevo problema.

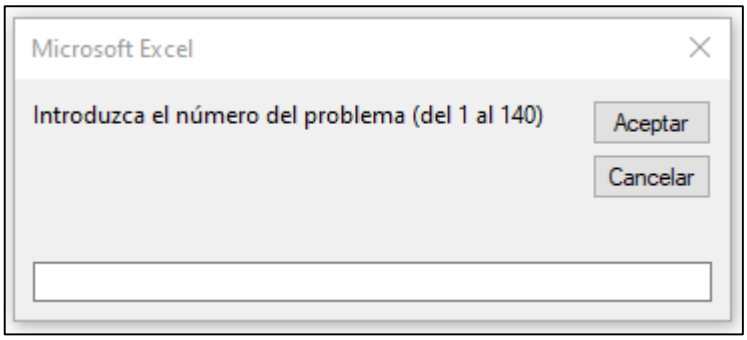

Figura 2.3-5: Paso 2 para introducir un nuevo problema.

Los problemas de la librería van del número 1 hasta el número 140. Por eso, si el usuario introduce un número que no se encuentra en ese rango, el programa devolverá un mensaje de error y se le pedirá de nuevo que introduzca un numero valido (Figura 2.3-6).

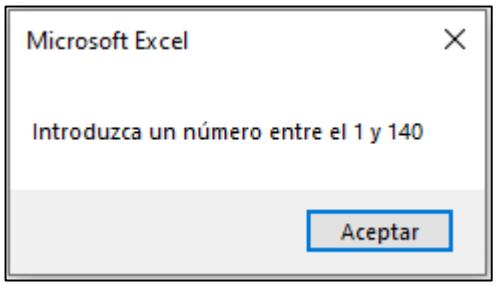

Figura 2.3-6: Mensaje de error al introducir un problema fuera del rango numérico.

Por otro lado, dentro de este mismo grupo "MPSPLib" se encuentra la opción de imprimir los resultados. Al presionar este botón, el programa recogerá los resultados obtenidos en la simulación, los pondrá en el formato específico de soluciones de la librería MPSPLib y finalmente abrirá una pestaña de Internet Explorer proporcionando la web para subir resultados de MPSPLib y copiando el resultado obtenido en la casilla correspondiente. Este resultado también estará copiado en el portapapeles para poder utilizarlo para otros fines.

2.3.2. **Regla de prioridad utilizada**

Este simulador está dedicado a la utilización de métodos heurísticos basados en reglas de prioridad. Es por ello por lo que se da la opción al usuario de seleccionar que regla de prioridad prefiere utilizar para ver cómo esta afecta al problema en cuestión. Para ello se ha introducido la opción de seleccionar en una lista desplegable una de las seis opciones de reglas de prioridad. Estas reglas de prioridad proporcionan un valor a cada actividad que puede programarse en función de unos parámetros. La actividad con mayor valor será elegida primera y en el caso de que dos actividades cumpliesen las mismas características, un sistema aleatorio decidirá cual programar primero.

- "HeuristicaDH". Esta regla de prioridad asigna un valor en función de si la actividad pertenece al camino crítico o no y si la actividad pertenece al proyecto que más tarde termina o no. La heurística se explicará más en detalle en la sección 3.2.
- "SLK0". *SLK0* es una variación de la anterior regla de prioridad en la cual solo se tiene en cuenta la pertenencia al camino crítico de cada proyecto. Al generarse múltiples empates, el sistema aleatorio decide qué actividad programar primero.

- "PSGSMINSLK". Esta heurística fue presentada por Villafáñez *et al.* (2019) y se basa en la ordenación del valor de las actividades en función de su holgura.
- "LPT – *Longest Processing Time*". Con esta regla de prioridad se programarán antes aquellas actividades que cuenten con mayor duración.
- "SPT – *Shortest Processing Time*". Al contrario que la regla de prioridad anterior, en este caso se programarán primero aquellas actividades cuya duración es menor.
- "Numero de actividad". Con esta regla se programará primero aquella actividad que tiene menor número de referencia y, en caso de empate, se programará la actividad que pertenece al proyecto con menor número de referencia.

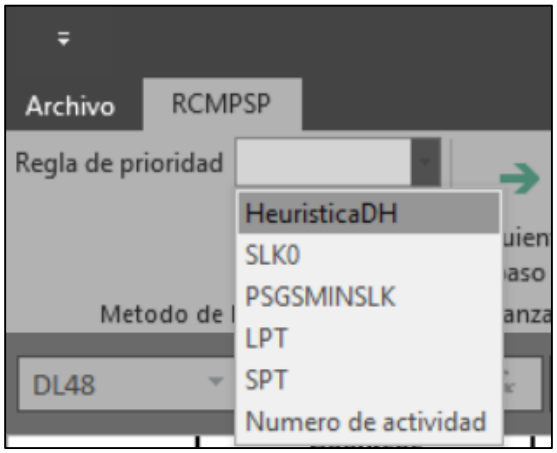

Figura 2.3-7: Desplegable con todas las reglas de prioridad disponibles.

2.3.3. **Simulación de una unidad temporal**

Esta opción permite avanzar una unidad de tiempo la simulación utilizando un sistema de generación de programación en paralelo (P-SGS). De manera secuencial el simulador realizará las siguientes acciones:

1. Se aumenta una unidad temporal y se desplaza la barra de señalización del día.
2. Se retorna a 0 todos los valores de las actividades suministrados por la regla de prioridad utilizados en la anterior programación.
3. Para cada actividad de cada proyecto se obtiene la fecha más temprana en la que puede iniciar teniendo en cuenta tanto las relaciones de precedencia como el número de días retrasados por la no disponibilidad de recursos. Si la actividad ya ha finalizado el simulador pasará a la siguiente.
4. Se calcula para cada actividad la fecha más tardía de finalización. Con este dato y la fecha de comienzo más temprana se calcula la holgura de cada actividad.
5. Se determinan las actividades que forman parte del conjunto de elegibles, esto es, que en el tiempo de programación en el que se encuentra la simulación, todas sus actividades predecesoras hayan terminado. Después mediante la regla de prioridad seleccionada se les asigna un valor.
6. Se dibujan las actividades en el diagrama Gantt señalando con diferentes colores atendiendo a si es crítica o no y a si se ha programado o no.[17]

[17] Si la actividad se programa en tiempo t no cambiará de color hasta el tiempo t +1.

7. A continuación, se valoran los recursos que hay disponibles para la programación en tiempo t. Para ello se restan de los recursos totales (globales y locales) todos los recursos utilizados por las actividades que actualmente se están llevando a cabo.
8. Después se selecciona aquella actividad que ha obtenido el mayor valor según la regla de prioridad aplicada dentro del conjunto de elegibles. Si no existen recursos suficientes para programar esa actividad, esta se retrasa una unidad temporal. Si por el contrario sí que existen suficientes recursos, esta actividad se programa para que empiece en esa unidad temporal t y se actualizan los recursos disponibles.
9. Se repite el paso 8 hasta que se hayan valorado todas las actividades del conjunto de elegibles.
10. Finaliza la simulación.

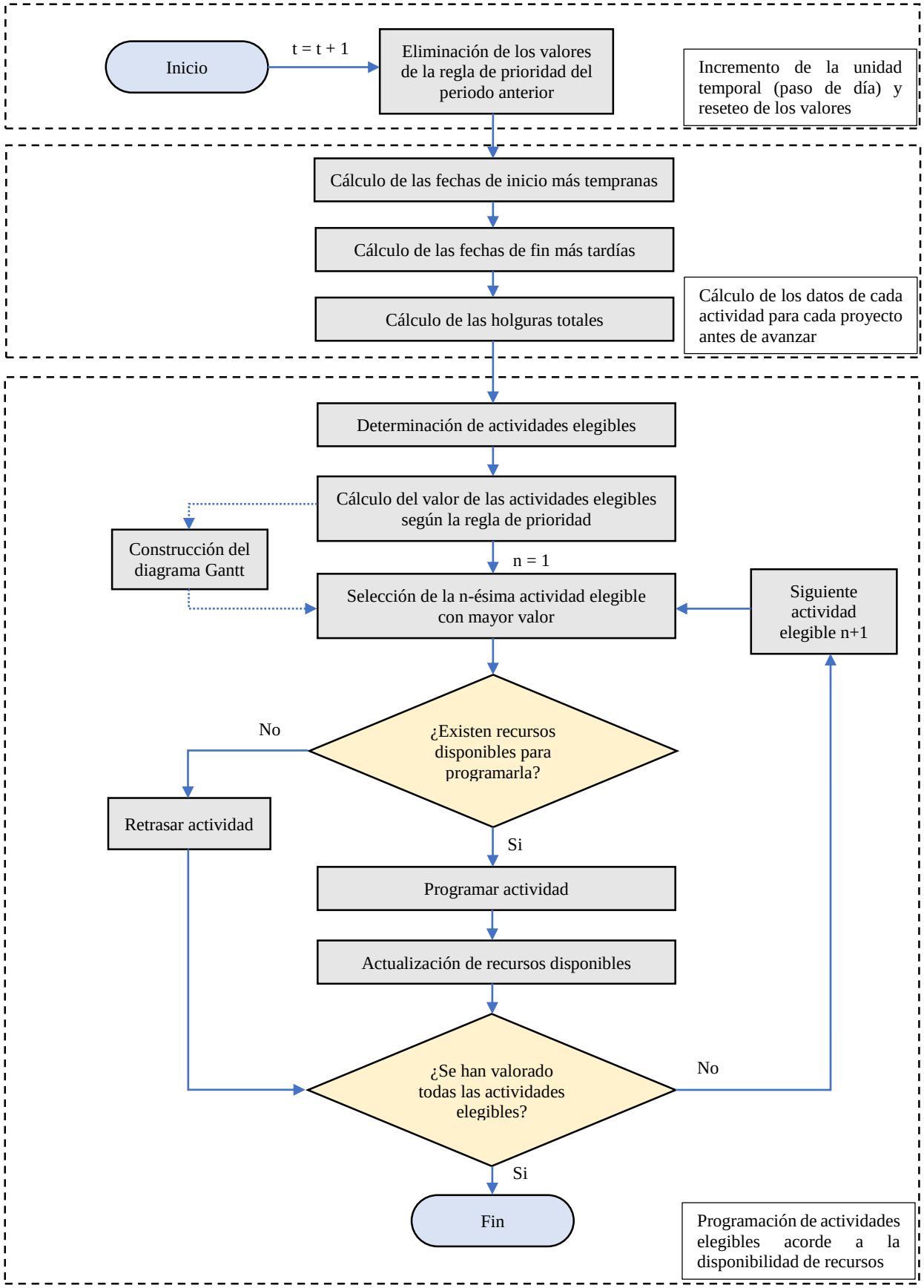

Figura 2.3-8: Diagrama de flujo de la simulación de una unidad temporal.

2.3.4. **Simulación automática**

Esta opción del simulador es una de las más útiles a la hora de generar uno o varios resultados de programación factibles para el problema que estamos tratando. Si bien la simulación de una unidad temporal permitía pasar el tiempo de uno en uno, en esta ocasión se realizarán todas las unidades temporales que sean necesarias hasta que la totalidad de las actividades estén programadas y finalizadas. En este apartado podemos encontrar diferentes opciones programables a la hora de generar resultados:

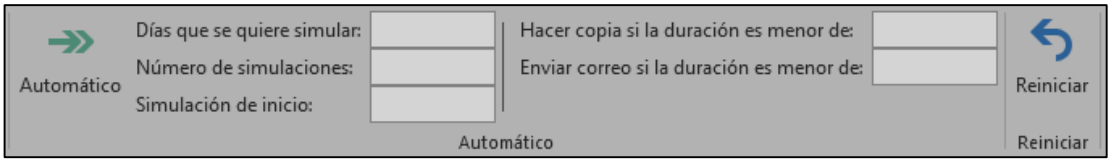

Figura 2.3-9: Opciones de automatizado.

- "Días que se quiere simular": Hace referencia al número de unidades temporales que queremos que el sistema avance. Si no se introduce ningún número, el simulador avanzará hasta que todas las actividades hayan finalizado.
- "Numero de simulaciones": Es el número de simulaciones completas que queremos que el programa realice, entendiendo por simulación completa la programación y finalización de todas las actividades de todos y cada uno de los proyectos.
- "Simulación de inicio": es el número de la simulación por el cual queremos que empiece. Por ejemplo, si se han realizado 50 simulaciones y se quieren hacer otras 50, tendremos que poner en esta casilla el valor 51 para que no borre los datos de las 50 primeras.
- "Hacer copia si la duración es menor de": en esta casilla se introducirá la duración total por debajo de la cual queremos obtener los datos completos. Si no se introduce ningún valor, el simulador solo obtendrá la mejor solución encontrada (solución con la mínima duración total).
- "Enviar correo si la duración es menor de": es el número de duración total por debajo del cual el simulador nos avisará mediante un correo electrónico con la solución encontrada. Para introducir la opción de mandar correo y la dirección de correo a la cual queremos que nos avise se ha introducido fuera de esta cinta de opciones las siguientes funcionalidades:

    - *Tick* para decir si queremos que nos envíe el correo con la solución obtenida.
    - Casilla para introducir la dirección de correo electrónico deseada.

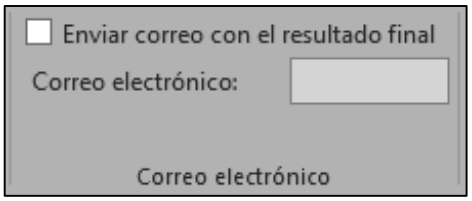

Figura 2.3-10: Opciones de correo.

Con estos datos y presionando el botón "Automático" la simulación comenzará y aparecerá un cuadro el cual señala el progreso a tiempo real del simulador. La barra superior nos indicará el

porcentaje de días que se han realizado en la simulación actual sobre el total[18] y la barra inferior nos indicará porcentaje de simulaciones que se han realizado (Figura 2.3-11). Por debajo de esta última barra figurará el tiempo estimado que falta para que la simulación termine.

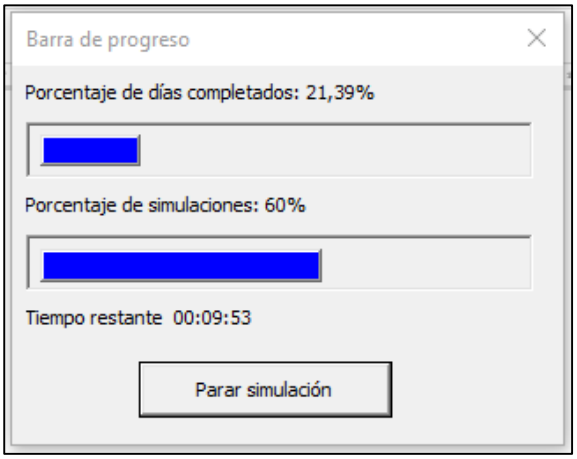

Figura 2.3-11: Cuadro de progreso.

Cada vez que se termina una programación completa, el programa guarda los valores de la duración total (TMS), el retraso medio de los proyectos (APD) y el tiempo que ha tardado en realizar la simulación en la columna correspondiente de la hoja de resultados. Así, una vez finalizada la automatización, por una parte, obtendremos los valores que ha ido generando y, por otra, en las pestañas figurarán la mejor solución y las soluciones viables con el TMS mínimo que hemos introducido.

2.3.5. **Representación gráfica**

Este grupo de opciones está dedicado a dar la opción al usuario de graficar o no graficar las soluciones obtenidas, tanto el Gantt como el camino crítico. El motivo por el que se incluye esta opción no es otro que agilizar los resultados obtenidos. Cuando el programa tiene que simular y a la vez dibujar el gráfico se ralentiza, haciéndose fastidiosamente largo cuando son programaciones de muchas unidades temporales. Por otro lado, y para facilitar un punto medio entre agilidad e información visual, se ha añadido la opción de, en el caso que se desee graficar el Gantt, que este no se actualice en cada unidad temporal si no que se actualice cada cierto tiempo (paso de gráfica).

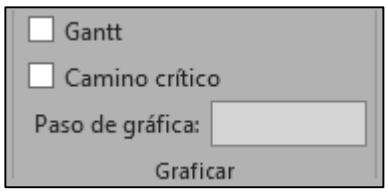

Figura 2.3-12: Opciones de gráfico.

[18] Nótese que esta función solo será real en el caso de que introduzcamos el número de días que queremos que transcurran.

2.3.6. **Mostrar información y recursos**

Los datos que utiliza el programa como las fechas de inicio y fechas de finalización son relevantes para el simulador, aunque no aportan mayor información de cara al usuario ya que la representación gráfica del diagrama de GANTT y los gráficos de los recursos ya ofrecen esta información.

Por este motivo se ha añadido un botón de "Mostrar/Ocultar información" (ver Figura 2.3-13) para que, cuando este botón se pulse, se oculten o se muestren los diferentes datos relativos a la simulación. Esta opción hace que la vista del usuario quede más "limpia" y no interfieran los datos en obtener una información clara, concisa y de manera ágil.

Por otro lado, en esta misma cinta de opciones existen cuatro botones correspondientes a los cuatro recursos disponibles en la simulación (Figura 2.3-13). Al pulsar cada uno de estos botones se mostrará el gráfico de recursos correspondiente, diferenciando si el recurso seleccionado es global o local. En el caso de que el recurso sea local tendremos que seleccionar en la lista desplegable situada en el gráfico el número del proyecto del cual queremos ver ese recurso particular.

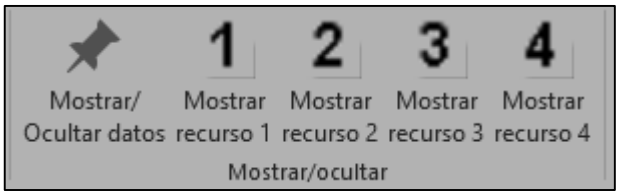

Figura 2.3-13: Opciones de mostrar/ocultar información.

2.3.7. **Comprobación de resultados**

El simulador da la opción de comprobar si se produce en cualquier periodo de tiempo alguna sobreasignación y, por consiguiente, que esa programación no sea válida. Esta funcionalidad está pensada para realizar una comprobación extra sobre la programación realizada. Si bien no se produce una sobreasignación una vez hemos puesto en marcha el simulador, con esta opción podemos ver respecto a la programación inicial, en que puntos de tiempo se produce sobreasignación de recursos, que recurso está sobre asignado y, en el caso de que el recurso sea local, a que proyecto pertenece el recurso que se encuentra sobre asignado. Esta información se podría obtener directamente de las gráficas, aunque es más intuitivo si, como en este caso, el simulador nos arroja una lista con todas las sobreasignaciones de recursos.

Otro motivo por el cual se ha introducido esta opción es el hecho de que el repositorio de soluciones de MPSPLib no detecta ningún error si la sobreasignación se da en los recursos globales, mientras que si esta se da en los recursos locales sí que arroja un mensaje de error y no permite que la solución quede publicada.

Cuando se presiona el botón de "Comprobar resultado" el simulador recorre cada una de las unidades temporales en las que existe al menos una actividad programada y recoge los recursos utilizados en esa unidad temporal. Si en algún periodo de tiempo existe sobreasignación se mostrará un mensaje de error en el que figurará el recurso que está sobre asignado, su carácter global o local, en el caso de que sea local figurará el número de proyecto al cual pertenece el recurso sobre asignado y, finalmente, su periodo de tiempo (Figura 2.3-14). En el caso de que no exista sobreasignación en ninguno de los periodos el programa lanzará un mensaje diciendo que no existe sobreasignación (Figura 2.3-15).

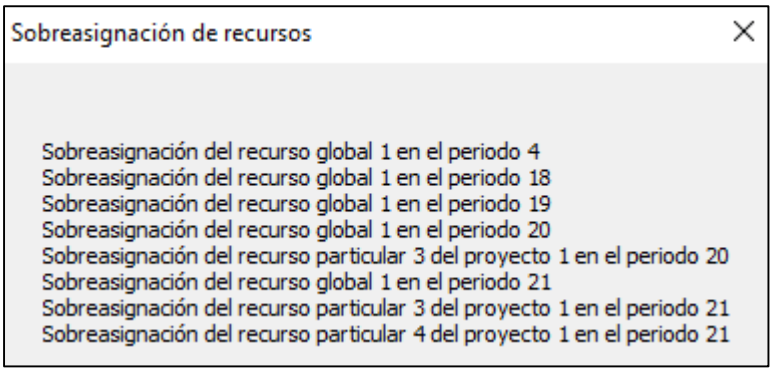

Figura 2.3-14: Información de sobreasignación de recursos.

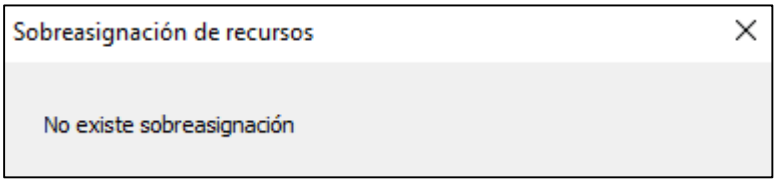

Figura 2.3-15: Mensaje de no sobreasignación de recursos.

2.4 Ventajas y limitaciones

Cabe destacar que el simulador implementado en este trabajo permite encontrar soluciones factibles a cualquier problema que se encuentre dentro de la librería MPSPLib. Esto puede facilitar el trabajo en futuras investigaciones ya que se trata de un simulador intuitivo el cual esta implementado mediante una herramienta de uso habitual como es Microsoft Excel. No obstante, el simulador cuenta con una serie de limitaciones a la hora de obtener resultados debido a diferentes características del mismo. A continuación, se mostrarán las principales ventajas y limitaciones de este simulador.

2.4.1. **Ventajas**

Dentro de este apartado se señalarán los principales puntos fuertes de este simulador:

1. Capacidad para programar varios proyectos. El uso de este simulador permite gestionar varios proyectos de forma simultánea y sencilla. Esto es especialmente útil cuando existen recursos globales o compartidos.
2. Obtención de programaciones sin sobreasignación.
3. Posibilidad de obtener más de una programación factible y encontrar aquella que satisface los objetivos del usuario y/o empresa.
4. Posibilidad de aplicar y estudiar diferentes reglas de prioridad a la hora de programar las actividades. Esto puede resultar útil para investigadores y académicos que quieran valorar diferentes métodos.
5. Estudio de problemas de la librería MPSPLib: el programa está pensado para obtener los problemas del repositorio MPSPLib, obteniendo información de los problemas automáticamente. Por otro lado, también da la opción de subir los resultados obtenidos. Este repositorio es uno de los más utilizados por los investigadores en el ámbito de programación de proyectos con recursos limitados.

6. Visualización rápida y detallada de los recursos. Mediante sus gráficos de recursos podemos ver en qué periodo de tiempo se produce la sobreasignación y en qué proyecto.
7. Desarrollo en entorno Microsoft. Al estar programado con leguaje VBA es más fácil en un futuro relacionarlo con herramientas utilizadas habitualmente como MS Project, MS Word u otros programas de MS Excel.

2.4.2. **Limitaciones**

Aunque las ventajas del simulador son numerosas, este también cuenta con una serie de limitaciones en su versión original que podrían llegar a ser suplidas en futuras versiones del mismo.

La primera limitación a tener en cuenta es el tiempo que tarda el simulador en ofrecer una programación completa. Este tiempo puede oscilar entre 4 segundos para los problemas más sencillos (2 proyectos de 30 actividades cada uno) hasta los 15 minutos para los problemas más complicados (20 proyectos de 120 actividades cada uno). Otro dato que afecta a la duración de la simulación son los días que se requieran simular. Las simulaciones de menos días, lógicamente, tardarán menos en realizarse. El dato de las duraciones también depende de las características de cada procesador del ordenador que, en este caso, se ha realizado con un ordenador cuyo modelo de procesador es Intel® Core™ i5-4210U CPU @ 1.70GHz – 2.40GHz.

Otra limitación del simulador presentado hace referencia a los datos tratados. Los datos tratados gestionan la información obtenida del repositorio de problemas MPSPLib. Por ello, existen limitaciones respecto al número de actividades sucesoras que puede tener cada actividad (3 actividades sucesoras como máximo), el número de recursos globales o locales que pueden existir (4 recursos en total como máximo) y el número de actividades que puede tener cada proyecto (120 actividades sin contar las de inicio y fin como máximo). Si bien estas limitaciones condicionan los problemas a tratar, en futuras versiones se podrían aumentar estos límites hasta el punto deseado con una modificación del código mínima.
La última limitación que se plantea es la restricción por parte de este simulador a incluir únicamente relaciones de precedencia del tipo fin-inicio entre las actividades. Asimismo, las actividades una vez programadas no pueden ser interrumpidas para utilizar los recursos en otra actividad.

2.5 Comparación con MS Project

MS Project es un conocido programa de Microsoft mediante el cual se pueden gestionar, programar y obtener diferente información sobre los proyectos como plazos, costes y recursos, además de contar con sistemas de actualización y evaluación del desempeño del proyecto.

En honor a la verdad, en este aspecto la herramienta MS Project sobrepasa con creces las funcionalidades del simulador presentado en este trabajo. No obstante, a la hora de nivelar los recursos, el programa MS Project busca aquellas unidades temporales en las que existe sobreasignación de recursos y simplemente aplaza el resto de actividades hasta que la sobreasignación desaparece. Esta forma de proceder del programa MS Project lleva a que, resolviendo la sobreasignación, la fecha de finalización del proyecto no se encuentre cerca del óptimo.

Si en vez de tener un proyecto tenemos varios proyectos, supone un trabajo tedioso el método que ofrece MS Project para gestionar varios proyectos a la vez. Además, dado su método de nivelación de recursos expuesto anteriormente, tampoco obtiene una programación cercana al óptimo.

Es en este punto en el que el simulador presentado sería de utilidad ya que, ante una serie de actividades programables, el usuario podría elegir las características de aquella que se debe programar primero. Dependiendo de la regla de prioridad utilizada se obtendrán diferentes duraciones, pero estas siempre estarán alineadas con las decisiones de prioridad implantadas por el usuario y no por un sistema predefinido como es el de MS Project.

Los dos programas no son excluyentes. El estar programados en el mismo entorno (entorno Microsoft) le da un valor añadido a la posibilidad futura de unir los dos programas, utilizando el programa MS Project para el seguimiento diario y el simulador para la nivelación de recursos.

Capítulo 3 Comprobación del simulador. Heurística "DH"

En el Capítulo 1 se ha realizado un breve resumen y explicación de los problemas de programación multiproyecto con restricciones de recursos así como de los métodos de resolución de los mismos. El simulador y sus funciones han sido presentadas en el Capítulo 2, mostrando todas las opciones que ofrece el programa. En este punto surge la necesidad de saber si este simulador funciona correctamente o si, por el contrario, genera una programación que no respeta alguna de las restricciones. El último capítulo está dedicado a este fin, la comprobación del correcto funcionamiento del simulador mediante tres métodos principalmente: la plataforma MPSPLib, los gráficos dentro del simulador y una funcionalidad del programa para comprobar los resultados.

Para cerciorarse del correcto funcionamiento del simulador es necesario utilizar una heurística que permita el ordenamiento de las actividades en función de algún valor. Con tal fin se podría haber utilizado alguna de las reglas de prioridad existentes y que se han comprobado útiles y eficaces a la hora de programar las actividades como es, por ejemplo, la que utiliza la mínima holgura como valor de decisión. Por el contrario, en este trabajo de plantea una heurística diferente, la heurística DH (*Drawers Heuristic*).

Por último, se plantea una comparación de la heurística presentada y los resultados de otras heurísticas dentro de la plataforma MPSPLib.

3.1 Métodos de comprobación.

Los programas de ordenador son siempre testeados y comprobados varias veces antes de su puesta a disposición del público y también durante su vida útil. Este simulador, como la mayoría de programas, también necesita algún método de comprobación que permita asegurar su correcto funcionamiento. No sería lógico que, en un futuro, este simulador se utilizase para ampliar los límites del conocimiento y que, llegados a cierto punto, se comprobase que el mismo contiene errores.

Es por esta razón por la que se han valorado no solo uno, sino tres métodos de comprobación independientes que aseguran su correcto funcionamiento. Una de las características necesarias de estos métodos de comprobación es que sean independientes entre ellas mismas para así asegurar que sus vías de comprobación no están relacionadas.

Los métodos de comprobación utilizados han sido:

- El propio repositorio de problemas MPSPLib.com, que cuenta con un apartado de "Rankings", al cual se puede subir (tras el registro de una cuenta personal, y la descripción del método usado para obtenerla) una solución a cualquiera de los problemas y es la propia aplicación web la que, si detecta durante el proceso de carga que la solución es errónea (incumple algunas de las restricciones planteadas), impide subir la solución y genera un mensaje de error.
- Comprobación visual mediante los gráficos de utilización de recursos generados por la simulación.
- Funcionalidad "Comprobar resultados" implementada en el simulador desarrollado.

3.1.1. **Método de comprobación plataforma MPSPLib**

El repositorio de problemas MPSPLib.com ha sido nombrado frecuentemente a lo largo de este trabajo. Consta de una colección de problemas multiproyecto con diferentes características y grados de complejidad, construida con el fin de estudiar y comparar la eficacia de los métodos de resolución que son propuestos por los investigadores.

Otra funcionalidad que ofrece es la posibilidad de poder subir los resultados obtenidos en la programación para evaluar su desempeño en comparación con otras técnicas de resolución. Este es el primer procedimiento que ha sido utilizado para la verificación de la factibilidad de los resultados obtenidos por el simulador presentado en este trabajo.

Cuando se sube una propuesta de solución para un problema dado, antes de ser aceptada la plataforma MPSPLib comprueba, en primer lugar, que esta cumple con la restricción de las relaciones de precedencia. En caso de error, indica qué actividad se encuentra mal programada (la actividad comienza antes de que finalice su predecesora). En el mensaje que proporciona la librería cuando se comete este error de programación es del tipo del que aparece en la Figura 3.1-1, en la cual se señala primero la actividad que no está respetando las relaciones de precedencia , en segundo lugar, la actividad que debería haber finalizado antes, y en tercer lugar, el nombre del proyecto al cual pertenecen estas actividades.

Figura 3.1-1: Mensaje de error por incumplimiento de relaciones de precedencia.

La segunda comprobación realizada por la plataforma consiste en verificar que la solución propuesta cumple con la restricción de recursos de carácter local por unidad de tiempo, arrojando un mensaje de error en el caso de que se utilicen más recursos locales de los disponibles. En caso de error, el mensaje devuelto indica el recurso local que se encuentra sobre asignado, la cantidad de recurso con la que se produce esa y el periodo de tiempo en el cual se genera la sobreasignación.

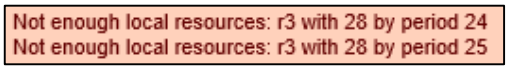

Figura 3.1-2: Mensaje de error por incumplimiento de la restricción de recursos locales.

Llegados a este punto cabe señalar una posibilidad de mejora no implementada para la herramienta de comprobación de la plataforma MPSPLib.com, ya que esta no es capaz de detectar las sobreasignaciones de los recursos globales en los problemas. Esto permite la presencia en la plataforma, dentro del listado de soluciones propuestas para un problema dado, soluciones que han sido aceptadas como válidas pero que realmente no cumplen con esas restricciones. Por ejemplo, en el problema 3 de la librería, la mejor solución contiene errores de sobreasignación de recursos globales que no han sido detectados. En la Figura 3.1-3 se puede observar el gráfico de asignación del recurso global 1 para esta solución y como esta asignación supera con creces el límite establecido (línea horizontal de puntos).

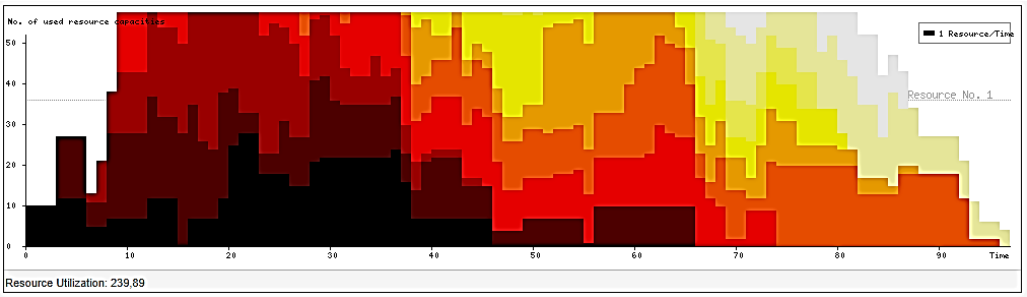

Figura 3.1-3: Gráfica del recurso 1 errónea.

3.1.2. **Método de comprobación gráfica de recursos del simulador**

Debido a la imposibilidad de la plataforma MPSPLib de detectar las sobreasignaciones en los recursos globales, se ha considerado oportuno introducir una forma alternativa de comprobar los resultados obtenidos.

Esta forma de comprobación de recursos globales se ha implementado mediante los propios gráficos suministrados por el simulador. Cada vez que el simulador avanza una unidad temporal en el proceso de programación, los datos de la asignación de todos los recursos se incorporan a una tabla que va almacenando toda esa información. La información de la tabla es utilizada para elaborar una serie de gráficos que muestran la asignación de cada recurso en cada unidad temporal, junto con una línea horizontal en el gráfico que indica cual es el límite de disponibilidad para ese recurso en el problema considerado. En el caso de que en la gráfica se observe que la asignación de recursos sobrepasa dicha línea, la programación no estaría cumpliendo la restricción de recursos y, por ende, no sería una programación viable.

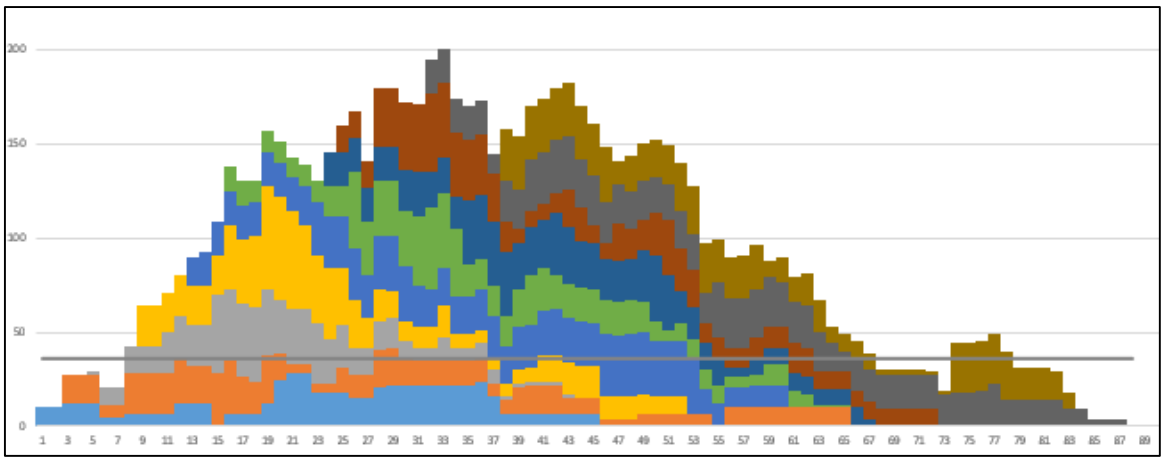

Figura 3.1-4: Gráfico de recursos del simulador con sobreasignación antes de simular.

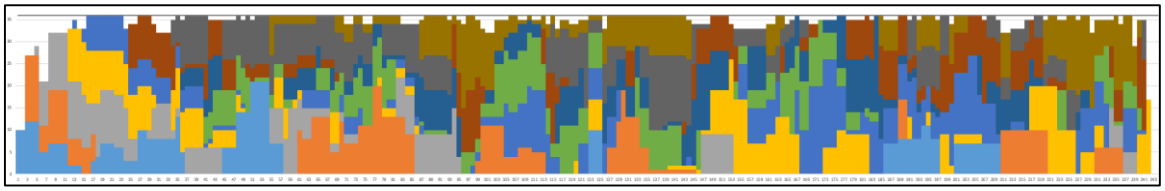

Figura 3.1-5: Gráfico de recursos del simulador sin sobreasignación después de simular.

Tras haber probado el simulador exhaustivamente con multitud de problemas se ha podido comprobar que las programaciones generadas con la herramienta no producen sobreasignaciones

en ninguno de los casos. Los problemas utilizados y el resultado obtenido en ellos se explican en el apartado 3.3.

3.1.3. **Método de comprobación función "Comprobar resultado"**

Pese a que, si bien con el método gráfico explicado anteriormente se podría comprobar visualmente que la programación obtenida no incumple ninguna de las restricciones de recursos y, aunque adicionalmente, al enviar una propuesta de solución a la plataforma MPSPLib.com esta vuelve a verificar tanto el cumplimento de las restricciones de recursos locales y precedencias entre actividades, se ha considerado necesario automatizar el procedimiento de verificación. Este sería el último método de comprobación implementado en el simulador, y que nos ofrece la posibilidad de asegurarnos con total fiabilidad de que no existe ninguna sobreasignación en la solución encontrada para un problema. Al seleccionar el botón correspondiente a esta funcionalidad, el programa recorre todas las unidades temporales recogiendo el uso de cada recurso en cada unidad temporal y arrojando un mensaje de error si se produce alguna clase de sobreasignación de recursos, ya sean globales o particulares. Para una explicación más en detalle se remite al lector a la sección 2.3.7 en la cual se explica este método más en detalle.

3.2 Heurística DH

En el algoritmo de programación utilizado, durante el proceso de asignación de los recursos disponibles a las actividades candidatas, es necesario utilizar alguna regla de prioridad que decida qué actividades se van a programar primero del conjunto de actividades elegibles en ese momento. Si bien esta regla de prioridad podría haber sido cualquiera de las ya conocidas y comprobadas en la literatura ("Total Slack" u Holgura Total, "Global Slack" u "Holgura Global", etc. (Browning y Yassine, 2010)) en este punto se ha decidido proponer una diferente, y que da lugar a una heurística menos convencional, la cual se ha denominado como heurística de cajones o "*Drawers Heuristic*". Esta heurística se basa en realizar una priorización de actividades diferente a las habituales. La mayoría de reglas de prioridad utilizan una característica de las actividades para su ordenación dentro de la lista de actividades elegibles mientras que la heurística propuesta utiliza "cajones" ordenados en los cuales las actividades incluidas tienen las mismas características, ofreciendo así la posibilidad de ordenar las actividades según más de un factor y ver cómo afecta esta ordenación al resultado final.

La manera de ejecutar la heurística es la siguiente:

0. Inicialización de los parámetros. Se definen las características a evaluar en el grupo de actividades elegibles en cada paso del algoritmo y, según las cuales, serán asignadas a cada cajón. El número de cajones al que asignaremos actividades elegibles con las mismas características puede modificarse según el criterio de la persona que utilice el simulador. En el caso de los problemas estudiados en este documento se han definido estos cuatro cajones:
   0.1. Primer cajón: a este cajón se asignan las actividades elegibles que tienen una holgura total igual a 0 y que pertenecen al proyecto que más tarde termina. Estas actividades son las que forman parte del camino crítico de la cartera de proyectos.
   0.2. Segundo cajón: se asignan a este cajón actividades que tienen holgura cero pero no forman parte del proyecto que más tarde termina. Estas actividades son actividades pertenecientes al camino crítico de un proyecto, pero no tienen por qué serlo de la cartera.
   0.3. Tercer cajón: se introducen en este cajón las actividades que no tienen holgura cero, pero que sí pertenecen al proyecto que más tarde termina.

0.4. Cuarto cajón: por último, se colocan aquí las actividades que no tienen holgura cero y que, a parte, no pertenecen al camino crítico.

1. Inicialización del contador temporal t=0. Comenzamos intentando programar las actividades que podrían comenzar en la fecha de inicio más temprano posible de la cartera.
2. En cada paso t del algoritmo, se intentarán programar todas las actividades elegibles que sea posible, es decir, actividades aún no programadas que podrían empezar en el instante de tiempo t. Para ello,
   2.1. Primero se clasifican una a una las actividades de la lista de elegibles en uno u otro cajón sí, según lo definido anteriormente, cumplen las condiciones de dicho cajón. Después, las actividades que pertenecen a cada cajón se reordenan de forma aleatoria. Para ello, al introducir cada actividad dentro de un "cajón" se asigna a esta, según unas condiciones predefinidas, un valor aleatorio entre un límite superior y un límite inferior, cuyo rango se define de forma diferente para cada cajón. Por ejemplo, a una actividad en el primer cajón (actividades que pertenecen al camino crítico de la cartera) se le puede asignar un valor aleatorio en un rango entre 1,5 y 1,99; para el segundo cajón, se les puede dar a las actividades un valor aleatorio en un rango entre 1 y 1,49[19]. Al final obtenemos una lista de actividades candidatas a ser programadas que puede ser ordenada, según ese valor asignado a cada una de ellas, de mayor a menor valor. En esta ordenación se respeta implícitamente el orden de asignación de las actividades a los cajones (gracias a los distintos rangos), pero estando estas aleatorizadas dentro de ellos. Esta parte de la Heurística se detalla con un ejemplo en la Figura 3.2-1.
   2.2. Siguiendo el orden de la lista de actividades candidatas obtenida, se recorre está seleccionando las actividades de mayor a menor valor asignado. La actividad seleccionada en cada momento se programará sólo en el caso de que estén disponibles las cantidades de todos y cada uno de los recursos que requiere para su ejecución, en cada periodo temporal durante toda su duración. Si no es así, esta actividad no se puede programar en este momento t y se retrasará su fecha más temprana de inicio posible en una unidad temporal (es decir, volverá a ser elegible en el próximo paso del algoritmo). A continuación, se selecciona la siguiente actividad con mayor valor y se repite el proceso hasta terminar la lista clasificada de actividades elegibles.
3. Se hace t= t+1 y se repite el paso 2 del algoritmo, es decir, se intentarán programar todas las actividades elegibles que sea posible, es decir, actividades aún no programadas que podrían empezar en el instante de tiempo t+1.
4. El proceso finaliza correctamente cuando todas las actividades estén programadas y se haya obtenido una programación factible desde el punto de vista de las restricciones temporales y de disponibilidad recursos.

[19] Nótese que una actividad del segundo cajón nunca podría llegar a obtener un valor superior que cualquiera de las actividades del primer cajón.

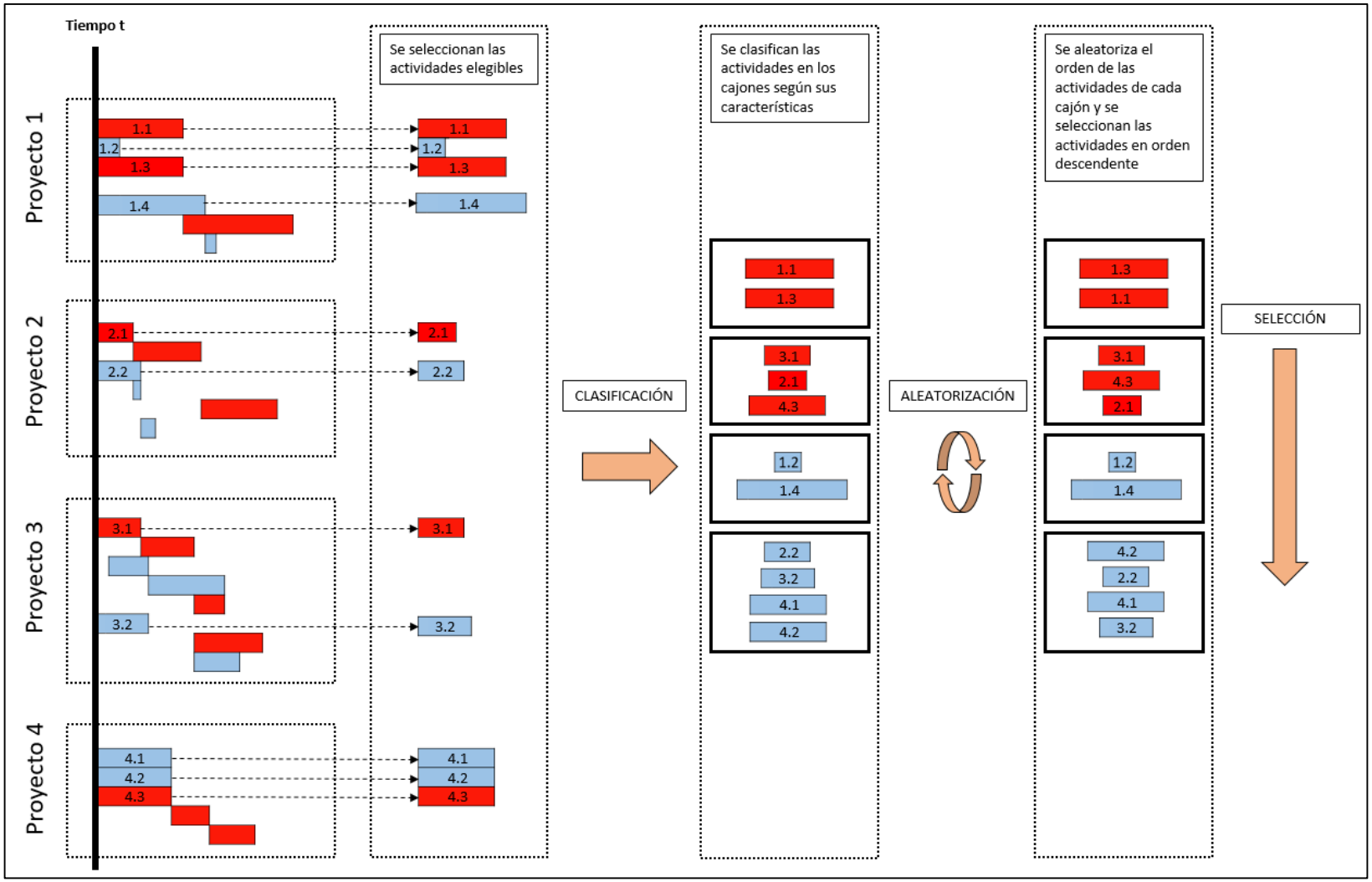

Figura 3.2-1: Procedimiento de la heurística de cajones en paso t del algoritmo.

Cabe destacar que esta heurística utiliza características *booleanas*, es decir, una actividad puede tener o no tener la característica, no pudiendo obtener un valor determinado de la misma.

Para la comprobación del funcionamiento del simulador, según el primer método explicado (mediante la librería MPSPLib), se ha utilizado esta heurística con los primeros sesenta problemas de la librería. Para ello se ha procedido a realizar mediante el simulador 100 ejecuciones de cada problema, seleccionar el mejor resultado para cada problema según el criterio de menor Duración Total de la cartera de proyectos ("Total Makespan" o "TMS"), los cuales se ha subido como soluciones propuestas al problema correspondiente en la plataforma MPSPLib.com.

Estas soluciones han sido aceptadas por la plataforma en su totalidad lo que, junto con los otros dos métodos de comprobación utilizados, lleva a concluir que el simulador propuesto en este trabajo es válido y generar soluciones factibles que no incumplen ninguna de las restricciones del problema RCMPSP.

3.3 Resultados y comparativa con otras heurísticas

En la rama de la Investigación Operativa, donde los investigadores proponen distintas metodologías para resolver un mismo problema, generalmente muy complejo, resulta esencial disponer de una librería de instancias de ese problema que, generadas en la medida de lo posible utilizando la metodología del Diseño de Experimentos, recoja una representación significativa de todas las posibles variaciones del mismo, desde las más sencillas hasta las más complejas.

Este es el motivo por el cual existen plataformas como la de MPSPLib.com, donde se ponen a disposición de la comunidad científica esas colecciones de instancias. Se tiene así la posibilidad no sólo de disponer de problemas con los que desarrollar y proponer nuevas metodologías de resolución, sino también de poder comprobar cuál es el desempeño de un método dado en comparación con los demás al poder comparar los resultados obtenidos con cada instancia.

Esto resulta especialmente importante en especial en problemas, como es el caso del que nos ocupa, el RCMPSP, donde por la naturaleza del problema, explicada en el apartado 1.2.1 (problema NP-Hard) no se puede estar seguro de haber obtenido el resultado óptimo, y por lo tanto, se desconoce, por así decirlo, "cómo de buena" es la solución obtenida. Por este motivo se compararán los resultados obtenidos por la heurística DH con los resultados disponibles en MPSPLIB.com obtenidos por otras heurísticas.

La función objetivo a comparar será la del mínimo tiempo total ("Total Makespan" o "TMS") de toda la cartera de proyectos de cada problema. Para ello se ha considerado que, si bien funciones como el retraso medio de los proyectos ("Average Project Delay" o "APD") pueden llegar a ser relevantes, en la vida real, la variable de decisión última para elegir una u otra programación recae casi siempre en el menor tiempo total de ejecución de todos los proyectos.

El resumen de los resultados obtenidos al aplicar la heurística a los 60 primeros problemas de la MPSPLib (ID's del 1 al 60) utilizando el criterio de minimización del tiempo total de ejecución de la cartera (TMS) es el siguiente:

- En 11 de los 60 problemas (18,33% de los problemas) se ha obtenido el mejor resultado, superando así en ellos a otros métodos heurísticos y metaheuristicos propuestos. Si no se consideran las soluciones erróneas actualmente aceptadas de forma incorrecta en la librería, este número ascendería a 12.

- En 26 de los 60 problemas (43,33% de los problemas) se ha obtenido un resultado igual o mejor que el mejor resultado presente en esos problemas.
- En 43 de los 60 problemas (71,66% de los problemas) se ha obtenido un resultado que difiere menos del 5% de la mejor solución.
- En 54 de los 60 problemas (90% de los problemas) se ha obtenido un resultado que difiere menos del 10% de la mejor solución.

Hay que destajar que la principal ventaja de la heurística propuesta es que, frente a lo complejidad de algunas propuestas recogidas en la librería MPSPLib, esta heurística puede ser considerada como de fácil aplicación, que no conlleva un gran esfuerzo computacional y que es sencilla de aplicar a todos los problemas que se desee sin tener que diseñar un método de resolución "*adhoc*" para cada problema.

Por ello, y en virtud de los resultados obtenidos, se considera que la aplicación de esta heurística puede resultar de gran interés de cara a la programación de forma conjunta de las actividades de múltiples proyectos pertenecientes a una cartera y en la que el objetivo sea minimizar la duración total de la ejecución de la cartera.

Los resultados detallados obtenidos por la heurística "DH" se pueden observar en la Tabla 1, tabla que recoge a lo largo de varias páginas, para cada una de las primeras 60 instancias de la librería MPSPLib estudiadas, la comparativa del mejor resultado obtenido para el criterio TMS con el mejor resultado existente en el momento de consulta de la librería. Las filas sombreadas en verde corresponden a los casos en que la obtención del mejor resultado por parte de la heurística propuesta ha superado a los existentes en la librería.

ID	Nombre del problema	Número de actividades	Número de proyectos	Autor	TMS	Método
1	mp_j30_a10_nr1	30	10	TLA_kuehn	187	GA_WS
				Álvarez-Campana Rodríguez	196	DH/MPR
2	mp_j30_a10_nr2	30	10	A. Ahmeti / N. Musliu	108	MATHEUR
				Álvarez-Campana Rodríguez	114	DH/MPR
3	mp_j30_a10_nr3	30	10	Gómez/Fernández *	97	ACO+SMT
				Álvarez-Campana Rodríguez	242	DH/MPR
4	mp_j30_a10_nr4	30	10	Lopez-Paredes/Pajares/Villafañez	143	PSGSMINSLK
				Álvarez-Campana Rodríguez	143	DH/MPR
5	mp_j30_a10_nr5	30	10	TLA_kuehn	184	GA_WS
				Álvarez-Campana Rodríguez	185	DH/MPR
6	mp_j30_a2_nr1	30	2	Gómez/Fernández	61	ACO+SMT
				Álvarez-Campana Rodríguez	72	DH/MPR
7	mp_j30_a2_nr2	30	2	pérez/posada	58	GA-RK
				Álvarez-Campana Rodríguez	61	DH/MPR
8	mp_j30_a2_nr3	30	2	Trautmann/Homberger	65	MAS/PS
				Álvarez-Campana Rodríguez	65	DH/MPR
9	mp_j30_a2_nr4	30	2	Dietz/Homberger	54	CMAS/ES-STV
				Álvarez-Campana Rodríguez	56	DH/MPR
10	mp_j30_a2_nr5	30	2	Ameisenbär(Daniela)	58	CMAS/ES
				Álvarez-Campana Rodríguez	65	DH/MPR
11	mp_j30_a20_nr1	30	20	Birkner/Homberger	417	CMAS/ES-BORDA
				Álvarez-Campana Rodríguez	431	DH/MPR
12	mp_j30_a20_nr2	30	20	Álvarez-Campana Rodríguez	275	DH/MPR

* Resultados erróneos que no respetan las restricciones de recursos.

ID	Nombre del problema	Número de actividades	Número de proyectos	Autor	TMS	Método
13	mp_j30_a20_nr3	30	20	Álvarez-Campana Rodríguez	303	DH/MPR
14	mp_j30_a20_nr4	30	20	Álvarez-Campana Rodríguez	182	DH/MPR
15	mp_j30_a20_nr5	30	20	Gómez/Fernández *	159	ACO+SMT
				Álvarez-Campana Rodríguez	407	DH/MPR
16	mp_j30_a5_nr1	30	5	pérez/posada	82	GA-RK
				Álvarez-Campana Rodríguez	85	DH/MPR
17	mp_j30_a5_nr2	30	5	RashidiPour/Shakhsi-Niaei	78	
				Álvarez-Campana Rodríguez	81	DH/MPR
18	mp_j30_a5_nr3	30	5	Tony Wauters	103	HYPER
				Álvarez-Campana Rodríguez	113	DH/MPR
19	mp_j30_a5_nr4	30	5	Dietz/Homberger	76	RES
				Álvarez-Campana Rodríguez	76	DH/MPR
20	mp_j30_a5_nr5	30	5	Gómez/Fernández	86	AC0+HC+SMT
				Álvarez-Campana Rodríguez	90	DH/MPR
21	mp_j90_a10_nr1	90	10	Álvarez-Campana Rodríguez	154	DH/MPR
22	mp_j90_a10_nr2	90	10	Dietz/Homberger	128	RES
				Álvarez-Campana Rodríguez	128	DH/MPR
23	mp_j90_a10_nr3	90	10	Álvarez-Campana Rodríguez	209	DH/MPR
24	mp_j90_a10_nr4	90	10	Dietz/Homberger	150	RES
				Álvarez-Campana Rodríguez	150	DH/MPR
25	mp_j90_a10_nr5	90	10	TLA_kuehn	227	
				Álvarez-Campana Rodríguez	241	DH/MPR
26	mp_j90_a2_nr1	90	2	Dietz/Homberger	88	RES
				Álvarez-Campana Rodríguez	88	DH/MPR
27	mp_j90_a2_nr2	90	2	Túlio Toffolo	117	MATHEUR
				Álvarez-Campana Rodríguez	127	DH/MPR
28	mp_j90_a2_nr3	90	2	Dietz/Homberger	114	RES
				Álvarez-Campana Rodríguez	114	DH/MPR
29	mp_j90_a2_nr4	90	2	Dietz/Homberger	92	RES
				Álvarez-Campana Rodríguez	92	DH/MPR
30	mp_j90_a2_nr5	90	2	Dietz/Homberger	121	CMAS/SA
				Álvarez-Campana Rodríguez	121	DH/MPR
31	mp_j90_a20_nr1	90	20	Tony Wauters	97	GT-MAS
				Álvarez-Campana Rodríguez	99	DH/MPR
32	mp_j90_a20_nr2	90	20	RashidiPour/Shakhsi-Niaei	163	GenConst
				Álvarez-Campana Rodríguez	169	DH/MPR
33	mp_j90_a20_nr3	90	20	Dietz/Homberger	122	RES
				Álvarez-Campana Rodríguez	122	DH/MPR
34	mp_j90_a20_nr4	90	20	Álvarez-Campana Rodríguez	170	DH/MPR
35	mp_j90_a20_nr5	90	20	Álvarez-Campana Rodríguez	224	DH/MPR
36	mp_j90_a5_nr1	90	5	Dietz/Homberger	79	CMAS/ES
				Álvarez-Campana Rodríguez	79	DH/MPR
37	mp_j90_a5_nr2	90	5	Dietz/Homberger	114	RES
				Álvarez-Campana Rodríguez	118	DH/MPR
38	mp_j90_a5_nr3	90	5	Dietz/Homberger	138	RES
				Álvarez-Campana Rodríguez	140	DH/MPR
39	mp_j90_a5_nr4	90	5	Dietz/Homberger	123	RES
				Álvarez-Campana Rodríguez	125	DH/MPR
40	mp_j90_a5_nr5	90	5	A. Ahmeti / N. Musliu	151	MATHEUR
				Álvarez-Campana Rodríguez	162	DH/MPR
41	mp_j120_a10_nr1	120	10	Dietz/Homberger	130	CMAS/ES
				Álvarez-Campana Rodríguez	130	DH/MPR
42	mp_j120_a10_nr2	120	10	TLA_kuehn	233	GA_WS
				Álvarez-Campana Rodríguez	277	DH/MPR
43	mp_j120_a10_nr3	120	10	RashidiPour/Shakhsi-Niaei	138	GenConst
				Álvarez-Campana Rodríguez	141	DH/MPR
44	mp_j120_a10_nr4	120	10	Álvarez-Campana Rodríguez	367	DH/MPR

ID	Nombre del problema	Número de actividades	Número de proyectos	Autor	TMS	Método
45	mp_j120_a10_nr5	120	10	A. Ahmeti / N. Musliu	476	MATHEUR
				Álvarez-Campana Rodríguez	479	DH/MPR
46	mp_j120_a2_nr1	120	2	Tony Wauters	155	HYPER
				Álvarez-Campana Rodríguez	161	DH/MPR
47	mp_j120_a2_nr2	120	2	Túlio Toffolo	133	MATHEUR
				Álvarez-Campana Rodríguez	143	DH/MPR
48	mp_j120_a2_nr3	120	2	A. Ahmeti / N. Musliu	272	MATHEUR
				Álvarez-Campana Rodríguez	276	DH/MPR
49	mp_j120_a2_nr4	120	2	TLA_kuehn	146	GA_WS
				Álvarez-Campana Rodríguez	146	DH/MPR
50	mp_j120_a2_nr5	120	2	Túlio Toffolo	108	MATHEUR
				Álvarez-Campana Rodríguez	118	DH/MPR
51	mp_j120_a20_nr1	120	20	Túlio Toffolo	76	MATHEUR
				Álvarez-Campana Rodríguez	87	DH/MPR
52	mp_j120_a20_nr2	120	20	RashidiPour/Shakhsi-Niaei	202	GenConst
				Álvarez-Campana Rodríguez	212	DH/MPR
53	mp_j120_a20_nr3	120	20	TLA_kuehn	230	GA_WS
				Álvarez-Campana Rodríguez	252	DH/MPR
54	mp_j120_a20_nr4	120	20	TLA_kuehn	201	GA_WS
				Álvarez-Campana Rodríguez	212	DH/MPR
55	mp_j120_a20_nr5	120	20	Túlio Toffolo	178	MATHEUR
				Álvarez-Campana Rodríguez	191	DH/MPR
56	mp_j120_a5_nr1	120	5	Túlio Toffolo	75	MATHEUR
				Álvarez-Campana Rodríguez	83	DH/MPR
57	mp_j120_a5_nr2	120	5	Túlio Toffolo	164	MATHEUR
				Álvarez-Campana Rodríguez	182	DH/MPR
58	mp_j120_a5_nr3	120	5	Álvarez-Campana Rodríguez	198	DH/MPR
59	mp_j120_a5_nr4	120	5	Álvarez-Campana Rodríguez	184	DH/MPR
60	mp_j120_a5_nr5	120	5	Álvarez-Campana Rodríguez	256	DH/MPR

Tabla 1: Resultados y comparación de los mejores resultados obtenidos por la heurística propuesta "DH" en relación a los resultados disponibles en MPSPlib.com (http://www.mpsplib.com/ranking.php?method=&criterion=tms&j=&p=&g=) a fecha de 2021/06/15.

CONCLUSIONES

Este trabajo documenta la labor realizada para desarrollar un simulador que permite la resolución de problemas de programación multiproyecto con restricciones de recursos (Problemas RCMPSP). El simulador ofrece a su vez la posibilidad de implementar y probar la eficiencia de diferentes mecanismos a la hora de obtener programaciones factibles para ese tipo de problemas. Suple así algunas carencias que presenta el software comercial actualmente disponible cuando trata de realizar una nivelación simultanea tanto en el uso de los recursos globales, compartidos por un conjunto de proyectos que forman parte de una cartera, como de los recursos locales asignados en exclusiva a cada uno de estos proyectos.

El software comercial de uso tan extendido como MS Project es capaz de realizar una nivelación en la utilización de los recursos modificando la programación de las actividades del proyecto o de una cartera de proyectos, pero lo hace sin atender al objetivo principal de la programación el cual, habitualmente, es la minimización del tiempo total de ejecución de la cartera o "TMS". El resultado obtenido generalmente es una reprogramación de actividades que, aunque respeta las limitaciones de disponibilidad de recursos, lo hace a costa de crear cierta infrautilización de esos recursos que a su vez aumenta más de lo necesario el tiempo de ejecución del conjunto de los proyectos.

El simulador desarrollado, en cambio, ofrece la posibilidad de realizar esas reprogramaciones en el conjunto de los proyectos utilizando diversas heurísticas basadas en reglas de prioridad. El usuario o director de proyectos puede probar con diferentes reglas de prioridad, para así ver cual, además de obtener un resultado factible desde el punto de vista de las limitaciones en los recursos, se adapta mejor al cumplimiento de la función objetivo.

Al tratarse de una herramienta en cuyo desarrollo ha primado la funcionalidad, lógicamente este carece de la apariencia, funcionalidades y opciones que poseen otros programas comerciales que hacen que estos sean más vistosos y atractivos. No obstante, cabe señalar que, al estar programado en lenguaje *Visual Basic for Applications* (VBA), esto puede facilitar la posibilidad de conectarlo con otros softwares de gestión de proyectos, y en especial, el propio Microsoft Project, suplementando así las carencias de estos. Además, el código del simulador ha sido desarrollado siguiendo una estructura de desarrollo modular, lo que permitirá que sea posible añadir en el futuro, con cierta facilidad, nuevas heurísticas a las que actualmente tiene disponible el usuario, y probar así su rendimiento.

Para verificar el desarrollo del simulador, la correcta implementación de las heurísticas, así como la calidad de los resultados obtenidos se ha recurrido a su estudio empleando los problemas propuestos en la librería MPSPLib. Las programaciones solución proporcionadas por el simulador han sido comprobadas a través de tres técnicas independientes, mostrándose plenamente eficaz a la hora de obtener programaciones factibles que respetan las restricciones de recursos, así como las relaciones de precedencia.

Por otro lado, y como otra de las aportaciones más importantes, se ha presentado una nueva heurística denominada de cajones o heurística DH (*Drawers Heuristic*). Implementada en el simulador y haciendo uso de los problemas propuestos en la librería MPSPLib ha demostrado que, además de fácil de implementar, resulta ser un método efectivo de reprogramación de actividades. Destacar el hecho de que, adicionalmente, ha obtenido en un buen número de casos mejores resultados (y en otros muchos igualado) que muchas de las mejores soluciones obtenidas con los otros métodos heurísticos y metaheurísticos allí recogidos.
Si bien no se puede decir de modo general que la heurística DH sea mejor que otras, la facilidad de implementación, la sencillez de su ejecución y su adecuación a un amplio rango de problemas hace que sea, cuanto menos, una buena opción a la hora de programar las actividades en entornos multiproyecto con restricciones de recursos.

Como conclusiones finales, destacar que el simulador podría ser utilizado para futuras investigaciones debido a su robustez, su eficacia y su manejo intuitivo y visual. Por otra parte, la heurística de cajones puede abrir nuevas posibilidades a la hora de programar actividades bajo diferentes criterios combinados; y finalmente, ambos elementos juntos podrían llegar a tener resultados muy interesantes en un futuro aumentando el límite del conocimiento sobre los problemas RCMPSP.

Futuros pasos

El simulador presentado en este trabajo ha demostrado ser eficaz a la hora de programar actividades de un conjunto de proyectos con unas características determinadas (características de los problemas MPSPLib). Si bien estos problemas son útiles para el estudio de métodos de resolución, se trata de problemas teóricos que a la hora de la verdad no son capaces de reflejar en su totalidad la complejidad de un proyecto en la vida real.

Por este motivo, una de las líneas futuras de actuación relacionada con la mejora del simulador, es su adaptación para que sea capaz de reprogramar proyectos más complejos: con mayor número de recursos, actividades y actividades predecesoras. También se le podría proporcionar valor añadido mediante la mejora de la *interface* de usuario, así como aumentando sus funcionalidades.

Por otro lado, se podría intentar desarrollar una pasarela que permita conectar el software de gestión de proyectos Microsoft Project y el propio simulador, produciéndose una sinergia que conllevaría una mejor gestión de los recursos y de los proyectos. El simulador está programado con leguaje Microsoft (Visual Basic), es por ello por lo que esta línea de progreso del simulador tiene un gran potencial. Otra posible opción es la creación de un "*add-in"* o complemento que permitiese ejecutar el simulador directamente desde la interface de Microsoft Project.

En cuanto a la heurística propuesta, la heurística de cajones, se puede decir que, bajo la configuración utilizada (los cajones y sus características), ha obtenido buenos resultados en comparación con otras heurísticas. La realización del resto de problemas presentes en la librería MPSPLib podría ser útil para asegurar sus buenos resultados.

En el futuro se podría llegar a realizar un estudio de sensibilidad para evaluar el desempeño de la heurística, con parámetros cómo el número de cajones, sus características y su ordenación, estudiando cómo afectan estos a la efectividad y la calidad de las programaciones solución obtenidas en función de las características y complejidad de los problemas a resolver (mayor o menor número de proyectos, número de actividades en cada proyecto, cantidad de recursos limitados, etc.). Esto permitiría determinar la idoneidad de la heurística según las características del proyecto.

BIBLIOGRAFÍA

Ángel-Martínez, E. Del *et al.* (2019) 'Un nuevo algoritmo de ramificación y acotamiento para el problema de la bisección de vértices', *Research in Computing Science*, 148(8), pp. 331–343. doi: 10.13053/rcs-148-8-25.

Araúzo, J. A., Pajares, J. y Lopez-Paredes, A. (2010) 'Simulating the dynamic scheduling of project portfolios', *Simulation Modelling Practice and Theory*, 18(10), pp. 1428–1441. doi: 10.1016/j.simpat.2010.04.008.

Bartusch, M., Möhring, R. H. y Radermacher, F. J. (1988) 'Quantitative Models , Data Structuring and Information Processing', *Annals of Operations Research*, 16, pp. 201–240.

Blazewicz, J., Lenstra, J. K. y Kan, A. H. G. R. (1983) 'Scheduling subject to resource constraints: classification and complexity', *Discrete Applied Mathematics*, 5(1), pp. 11–24. doi: 10.1016/0166-218X(83)90012-4.

Browning, T. R. y Yassine, A. A. (2010) 'Resource-constrained multi-project scheduling: Priority rule performance revisited', *International Journal of Production Economics*, 126(2), pp. 212–228. doi: 10.1016/j.ijpe.2010.03.009.

Homberger, J. (2007) 'A multi-agent system for the decentralized resource-constrained multi-project scheduling problem', *International Transactions in Operational Research*, 14(6), pp. 565–589. doi: 10.1111/j.1475-3995.2007.00614.x.

Kelley, J. E. (1959) 'Critical-Path Planning and Scheduling', pp. 160–173.

Kolisch, R. (2012) *Production and Logistics Integration, Journal of Business Logistique*.

Morillo, D., Moreno, L. y Díaz, J. (2014) 'Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión Parte 1', *Ingeniería y Ciencia*, 10(19), pp. 247–271. doi: 10.17230/ingciencia.10.19.12.

Morillo Torres, D., Moreno, L. y Díaz, J. (2014) 'Metodologías Analíticas y Heurísticas para la Solución del Problema de Programaciónde Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 2', *Ingeniería y Ciencia*, 10(20), pp. 203–227. doi: 10.17230/ingciencia.10.20.12.

Nievergelt, J. (2000) 'Exhaustive Search , Combinatorial Optimization and Enumeration ':, *Power*, pp. 18–35.

Patterson, J. H. (1984) 'Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem.', *Management Science*, 30(7), pp. 854–867. doi: 10.1287/mnsc.30.7.854.

Prado, D. (2015) *PERT/CPM*.

Valls, V., Ballestín, F. y Quintanilla, S. (2005) 'Justification and RCPSP: A technique that pays', *European Journal of Operational Research*, 165(2), pp. 375–386. doi: 10.1016/j.ejor.2004.04.008.

Valls, V., Martí, R. y Lino, P. (1996) 'A heuristic algorithm for project scheduling with splitting allowed', *Journal of Heuristics*, 2(1), pp. 87–104. doi: 10.1007/BF00226294.

Villafañez, F. *et al.* (2014) 'From the RCPSP to the DRCMPSP: Methodological foundations', in *Proceedings of the 2014 International Conference on Artificial Intelligence, ICAI 2014 - WORLDCOMP 2014*, pp. 594–598.

Villafañez, F. (2014) *Programación multi-proyecto con restricciones de financiación (F-DRCMPSP)*.

Villafáñez, F. *et al.* (2019) 'A generic heuristic for multi-project scheduling problems with global and local resource constraints (RCMPSP)', *Soft Computing*, 23(10), pp. 3465–3479. doi: 10.1007/s00500-017-3003-y.

Villafáñez, F. A. *et al.* (2018) 'A unified nomenclature for project scheduling problems (RCPSP and RCMPSP)', *Direccion y Organizacion*, 64, pp. 56–60.

INDICE DE FIGURAS

INDICE DE TABLAS