Universidad de La Laguna

SECCIÓN DE MATEMÁTICAS | FACULTAD DE CIENCIAS

Anabel Estévez Carrillo

Aplicación de la programación de tareas en la planificación de proyectos de desarrollo tecnológico

Application of job scheduling for planning technological development projects

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2018

Dirigido por

Joaquín Sicilia Rodríguez
David Alcaide López de Pablo

Joaquín Sicilia Rodríguez
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38271 La Laguna, Tenerife

David Alcaide López de Pablo
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38271 La Laguna, Tenerife

Agradecimientos

A mis tutores,
Joaquín Sicilia Rodríguez y David Alcaide López de Pablo,
por guiarme, aconsejarme y dedicar su tiempo al proceso
que ha conllevado la elaboración de esta memoria.
A todos mis profesores del Grado,
por su excelente labor docente que me ha permitido
comenzar a adentrarme en el mundo de las matemáticas.
A mi familia y amigos,
por apoyarme tanto todo este tiempo.

Resumen · Abstract

Resumen

Los modelos de planificación de proyectos y programación de tareas buscan obtener una secuencia de actividades que permita alcanzar un objetivo, delimitando la ejecutabilidad de las acciones en el tiempo, asignándoles los recursos necesarios y teniendo en cuenta las restricciones generales del problema. Este trabajo utiliza modelos de planificación para mejorar la distribución de las actividades a realizar en una empresa de desarrollo tecnológico localizada en la isla de Tenerife. Como resultado, se proponen soluciones que podrían aplicarse al negocio considerado. Las ideas y desarrollos expuestos también son válidos en otras compañías que pudieran estar interesadas en la distribución de las tareas o actividades entre sus empleados.

Palabras clave: *Investigación Operativa – Planificación de proyectos – Programación de tareas – Máquinas paralelas*

Abstract

Project Planning and Tasks Scheduling models seek to obtain a sequence of activities that allow reaching an objective, delimiting the executability of the actions over time, assigning them the necessary resources and taking into account the general restrictions of the problem. This memoir uses planning models to improve the distribution of activities to be carried out in a technological development company located on Tenerife. As a result, solutions are proposed that could be applied to the considered business. The ideas and developments exposed could be extended and used in other companies that may be interested in the distribution of tasks or activities among their employees.

Keywords: *Operational Research – Project Planning – Task Scheduling – Parallel Machines*

Contenido

Introducción

En Investigación Operativa se han desarrollado diversas disciplinas orientadas a optimizar la asignación de recursos para lograr la máxima eficiencia. En esta memoria se estudian dos campos de aplicación que resultan valiosos para abordar problemas en ámbitos empresariales: la Planificación de Proyectos y la Programación de Tareas. Ambas materias abordan cuestiones de características similares. La diferencia entre ambas radica en que la planificación de proyectos analiza la ruta a seguir para completar un proyecto en el menor tiempo posible, y la programación de tareas considera también la asignación de las actividades a los recursos disponibles.

Así pues, la Programación y Secuenciación de Tareas ('*Task Scheduling and Sequencing*') es un campo de trabajo e investigación dentro de las matemáticas enfocado a determinar la forma óptima en la que podemos asignar recursos a tareas en periodos temporales determinados teniendo en cuenta el objetivo que se desea optimizar. En un problema de programación de tareas podemos diferenciar tres componentes: las tareas o actividades que se deben elaborar, los recursos disponibles para su realización y las finalidades u objetivos que se desean lograr, los cuales nos permiten identificar, entre varias planificaciones, aquellas que sean óptimas.

Muchas veces, en la práctica, se tiene que desarrollar un proyecto a medio o largo plazo, el cual requiere de un conjunto amplio de tareas, manteniendo ciertas relaciones de precedencia entre ellas. En ese caso, la Planificación de Proyectos (*'Project Planning'*) nos aporta una metodología y un conjunto de herramientas que nos ayudan a determinar la programación temporal de las tareas para finalizar lo más pronto posible el proyecto. También nos ayuda a identificar aquellas actividades fundamentales que debemos controlar para evitar cualquier retraso en la realización de las mismas, ya que ello llevaría a una demora en la finalización del proyecto.

Motivación y objetivos

Desde que empecé mis estudios del Grado en Matemáticas, las asignaturas de Matemática Aplicada relacionadas con la empresa han sido las que me han despertado un mayor interés, por lo que desde un primer momento tuve claro que mi Trabajo de Fin de Grado debía ser un trabajo práctico. En concreto, me interesaba abordar un problema de Investigación Operativa en el que poder utilizar modelos y algoritmos matemáticos con el objetivo de realizar un proceso de toma de decisiones. Durante la asignatura 'Modelos de Investigación Operativa' estudiamos una breve introducción a los problemas de planificación, que inmediatamente llamaron mi atención debido a la gran versatilidad que presentan y su amplia utilidad empresarial. Lamentablemente, debido al extenso contenido de la asignatura y el limitado tiempo disponible, no se pudo profundizar en esta materia todo lo que hubiera deseado. Fue entonces cuando me decanté por realizar un trabajo en este campo y contacté con los profesores D. Joaquín Sicilia Rodríguez y D. David Alcaide López de Pablo, quienes se mostraron interesados en ayudarme. Durante la realización de mis prácticas externas en una empresa de desarrollo tecnológico me propusieron utilizar los datos registrados en la compañía para elaborar un caso práctico donde poder aplicar las técnicas metodológicas recogidas en esta materia, lo que me resultaba muy interesante ya que podría utilizar los conocimientos adquiridos para resolver un problema real en dicha actividad empresarial, con la ventaja que supone conocer el funcionamiento de la empresa y el tipo de tareas que elaboran sus empleados a diario.

Como consecuencia, el objetivo del presente Trabajo de Fin de Grado consiste en aplicar la metodología que proporciona la Planificación de Proyectos y la Programación de Tareas, junto algunos de los conceptos matemáticos que hemos visto a lo largo de las diferentes asignaturas del grado, para analizar y desarrollar la forma más eficiente de planificar los trabajos que deben ejecutarse por los empleados de la empresa dentro de los proyectos previstos.

1

Fundamentos de la Planificación de Proyectos

1.1. Introducción

La Programación y Control de Proyectos, se desarrolló a finales de los años 50 con la aplicación del método PERT (*Project Evaluation and Review Techniques*) en el desarrollo del proyecto Polaris. Para llevar a cabo dicho proyecto fue necesario coordinar la ejecución de varias actividades, debido a los problemas en la programación con los que se enfrentaba la marina de los Estados Unidos en la fabricación de submarinos armados. Posteriormente, los equipos de científicos se percataron de que los nuevos métodos desarrollados tenían un carácter multidisplicinario y decidieron adoptarlos para abordar problemas de ámbitos muy variados. En la actualidad, las técnicas de programación de proyectos se emplean en campos tan diversos como la construcción de grandes edificios, puentes, puertos; la gestión de procesos industriales; o el lanzamiento de un producto de una empresa. Como es lógico, el amplio número de personas interesadas en los métodos de planificación de proyectos ha incrementado considerablemente su popularidad.

En este capítulo se realiza una introducción a los problemas de planificación referidos a la ejecución de un proyecto con un número determinado de actividades a realizar en un orden preestablecido. Por ello, antes de continuar con el desarrollo de esta materia, conviene profundizar en el significado del concepto de proyecto.

Un proyecto es un conjunto articulado y coherente de actividades orientadas a alcanzar un objetivo siguiendo una metodología definida, para lo cual precisa de un equipo de personas idóneas, así como de otros recursos cuantificados en forma de presupuesto, que prevé el logro de determinados resultados sin contravenir las normas y buenas prácticas establecidas, y cuya programación en el tiempo responde a un cronograma con una duración limitada.

Todo proyecto debe requerir la especificación de las actividades a realizar. Habitualmente, el análisis y desarrollo de un proyecto consta de tres grandes fases (ver Velasco y Campins [17]):

1. Planificación y programación. En la planificación, se definen las actividades, sus interrelaciones y se estima la duración y los recursos necesarios. Posteriormente, se aborda la programación de las actividades para determinar el calendario de ejecución del proyecto, detallando las fechas de comienzo y de finalización de cada tarea a realizar.
2. Seguimiento y control. Se compara el estado actual con la programación inicial, tomando medidas correctivas cuando surgen dificultades.
3. Análisis final y evaluación de los resultados. Se analizan las diferencias entre previsiones y duraciones reales de las actividades, entre presupuesto y coste real, etc.

De este modo, los métodos enfocados a la gestión de proyectos tendrán el objetivo de planificar, dirigir y controlar el desarrollo de un proyecto, con el menor coste posible y dentro del periodo de tiempo del proyecto.

El método PERT(*Program Evaluation and Review Technique*) constituye una de las dos técnicas pioneras en el campo de la programación y el control de proyectos. Este método trata de determinar la duración del proyecto junto con la ruta crítica de las actividades. La idea general es mostrar un proyecto en forma gráfica y relacionar sus componentes de manera que permita identificar las actividades que son críticas para la finalización del mismo.

Para lograr tal fin, los proyectos deben tener las siguientes características:

- Las actividades deben estar bien definidas y su completación debe marcar la finalización del proyecto.
- Las actividades deben ser independientes en el sentido de que deben comenzar, detenerse y conducirse.
- Las actividades deben estar ordenadas de tal forma que una siga a otra en una secuencia dada.

El uso de técnicas de programación y control de proyectos proporciona múltiples beneficios. Entre ellos, asegura que el producto o desarrollo que resulta del proyecto está claramente definido y acordado por todas las partes implicadas, con lo que la responsabilidad de cada parte está perfectamente clara y asignada. Además, proporciona los medios para un efectivo seguimiento del proyecto con la confianza que supone demostrar un control visible.

En la siguiente sección, se presentan los aspectos fundamentales de las técnicas de planificación de proyectos, desarrollando los pasos que comprende la elaboración de una programación mediante el método PERT.

1.2. El método PERT

El método PERT es una técnica de la administración y gestión de proyectos que resulta idónea para proyectos que se desarrollan a largo plazo, y en los que participan diferentes agentes, pues nos permite tener una visión global del proyecto al tiempo que detalla un calendario con la fecha de comienzo y finalización de cada actividad e identifica la ruta crítica, en la que se deberá extremar la vigilancia para no provocar un retraso en la completación del proyecto. PERT es un método probabilístico, pues considera que la variable del tiempo de ejecución de una actividad es una variable desconocida de la cual sólo se tienen datos estimativos. El tiempo esperado de finalización de un proyecto es la suma de todos los tiempos esperados de las actividades sobre la ruta crítica.

Para controlar un proyecto por medio del método PERT, se necesita seguir seis pasos: construcción del grafo, cálculo de los tiempos esperados de las tareas, obtención de los tiempos más pronto posible y más tarde permisible, cálculo de holgura, determinación del camino mínimo, y finalmente establecimiento de un calendario. En los siguientes apartados se profundizará en cada uno de estos pasos.

Construcción del grafo PERT

Para aplicar el PERT se requiere conocer previamente la lista de actividades que incluye un proyecto. Se considera que el proyecto está terminado cuando todas las actividades han sido completadas. Para cada actividad, puede existir un conjunto de actividades predecesoras que deben ser completadas antes de que comience la nueva. Se construye un grafo o red del proyecto para representar las relaciones de precedencia entre las actividades. En dicha representación gráfica, cada actividad se simboliza como un arco y cada nodo ilustra la culminación de una o más actividades.

Consideremos un proyecto que consta de dos actividades A y B. Supongamos que la actividad A es predecesora de la actividad B. La representación gráfica de este proyecto se muestra en la figura 1.1. Así, el nodo 2 representa la culminación de la actividad A y el comienzo de la actividad B.

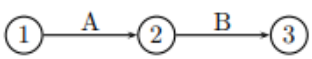


Figura 1.1. Proyecto de dos actividades A y B

Si suponemos ahora que las actividades A y B deben ser terminadas antes de que una actividad C pueda comenzar, el grafo del proyecto queda como se

muestra en la figura 1.2. En este caso, el nodo 3 representa que las actividades A y B se han terminado, además del inicio de la actividad C.

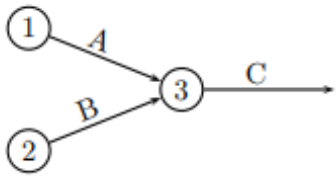


Figura 1.2. Proyecto de tres actividades A, B y C

Dado un conjunto de actividades y sus relaciones de precedencia, se puede construir una representación gráfica de acuerdo a las siguientes reglas:

1. El nodo 1 representa el inicio del proyecto. Por lo tanto, las actividades que parten del nodo 1 no pueden tener predecesoras.
2. El nodo terminal o final del proyecto debe representar el término de todas las actividades incluidas en el grafo.
3. Una actividad no puede ser representada por más de un arco en el grafo.
4. Dos nodos deben estar conectados por un único arco.

Para no violar las reglas 3 y 4, a veces es necesario introducir una actividad artificial o *dummy* que posee tiempo de duración nulo. Por ejemplo, supongamos que las actividades A y B son predecesoras de la actividad C y además comienzan al mismo tiempo. En este caso, una primera representación podría ser la indicada en la figura 1.3.

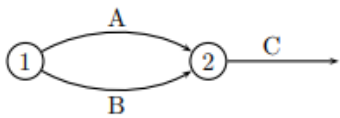


Figura 1.3. A y B predecesoras de C

Sin embargo, el grafo de la figura 1.3 viola la regla 4. Para corregir este problema, se introduce una actividad artificial indicada con un arco segmentado en la figura 1.4. La figura 1.4 refleja el hecho de que la actividad C tiene como predecesoras a A y B, pero sin quebrantar la regla 4. En otros casos, se deben agregar actividades artificiales para no infringir la regla 3.

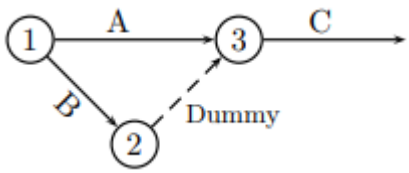


Figura 1.4. Incorporación de una actividad artificial

La numeración de los nodos se debe realizar de tal forma que una actividad siempre conecte un nodo de menor numeración con uno de mayor identificación en el sentido de avance del proyecto. Para ilustrar la representación de proyectos más complejos, existen dos procedimientos: la matriz de encadenamientos y el cuadro de prelaciones.

La matriz de encadenamiento es una matriz cuadrada cuya dimensión viene dada por el número de actividades en que se ha descompuesto el proyecto. Cuando un elemento de la matriz está marcado indica que para poder iniciar la actividad que corresponde a la fila que cruza ese elemento es necesario que se haya finalizado previamente la actividad que corresponde a la columna que cruza dicho elemento. De este modo, aquellas filas que no aparecen marcadas nos indican que las actividades no tienen ningún precedente.

Por otro lado, se puede emplear un cuadro de prelaciones formado por dos columnas. En la primera columna, están representadas todas las actividades en las que se ha descompuesto el proyecto y, en la segunda columna, figuran las actividades precedentes de su homóloga en la primera.

A partir de la matriz de encadenamientos o el cuadro de prelaciones se puede construir fácilmente el grafo PERT correspondiente.

Cálculo del tiempo esperado

La duración de una actividad puede no conocerse, en la mayoría de los casos, con exactitud. El método PERT aborda el problema del carácter aleatorio de las duraciones de las actividades de una manera muy peculiar, pues considera tres estimaciones de tiempo distintas: la estimación optimista (a), la estimación más probable (m) y la estimación pesimista (b) (ver Romero [15]).

La estimación optimista (a) representa el tiempo mínimo en que podría ejecutarse la actividad si todo marchara excepcionalmente bien.

La estimación más probable (m) llamada también estimación modal representa el tiempo que normalmente se empleará en ejecutar la actividad, es decir, cuando las circunstancias no sean excesivamente favorables ni excesivamente desfavorables.

La estimación pesimista (b) representa el tiempo máximo en que podría ejecutarse la actividad, si todas las circunstancias que influyen en su duración fueran totalmente desfavorables.

Una vez establecidas las tres estimaciones, se calcula el tiempo PERT denotado por D de ejecución de la actividad, ponderando las anteriores estimaciones por medio de la fórmula:

$$D = \frac{a + 4m + b}{6}$$

Determinación del tiempo más temprano y más tardío

Una vez construido el grafo que refleja las prelaciones entre las diferentes actividades en que se ha descompuesto el proyecto, y después de haber asignado los tiempos de ejecución a las actividades, podemos pasar a la siguiente fase, en la que se calcula los tiempos llamados *early* y *last* de cada suceso.

El tiempo *early* de un cierto suceso j trata de medir el tiempo mínimo necesario para llegar a él. El procedimiento de cálculo es iterativo, se efectúa de izquierda a derecha del grafo, comenzando por el suceso inicial del proyecto al que se le asigna un tiempo early de 0. Una vez calculado el tiempo *early* del inicio, se calculan los tiempos *early* de los sucesos en los que finalizan las actividades que nacen en el inicio del proyecto. De este modo, el tiempo *early* de un cierto suceso j, que representaremos por t_j, será igual a $t_j = max[t_i + t_{ij}], \forall i$ donde t_{ij} es la duración de la actividad que comienza en el suceso representado por el vértice i y finaliza en el suceso representado por el vértice j. Aplicando iterativamente esta fórmula podemos obtener los restantes tiempos early.

Una vez calculados los tiempos *early*, proseguimos calculando los llamados tiempos *last*. El tiempo *last* de un cierto suceso i trata de medir lo más tarde que podemos llegar al mismo de manera que la duración del proyecto (medida por el tiempo *early* del suceso final) no se retrase en ninguna unidad de tiempo. El procedimiento es iterativo, efectuándose de derecha a izquierda del grafo y comenzando por el suceso fin del proyecto, al que se le asigna un tiempo *last* igual al tiempo *early* previamente calculado. Una vez determinado el tiempo last del suceso final, se calculan los tiempos *last* de los sucesos en los que nacen actividades que concluyen en el suceso fin del proyecto. El tiempo *last* de un cierto suceso i, que representaremos por t_i^*, será igual a: $t_i^* = min[t_i^* - t_{ij}], \forall j$. Aplicando iterativamente esta fórmula obtenemos los restantes tiempos *last*.

Concepto de holgura y camino crítico

La importancia de los tiempos *early* y *last* es que constituyen la base para el cálculo de las holguras, que son la pieza fundamental en todo proceso de análisis del método PERT.

La *holgura* de un cierto suceso i, que representaremos por H_i, se define como la diferencia entre los tiempos last y early de dicho suceso, es decir:

$$H_i = t_i^* - ti$$

Nos indica el número de unidades de tiempo en que puede retrasarse la realización del mismo, de manera que la duración del proyecto no experimente ningún retraso. Por otro lado, la *holgura total* de una cierta actividad (i,j), que representaremos por H_{ij}^T se define como el tiempo que resulta de restar al tiempo *last* del suceso final el tiempo *early* del suceso inicial y la duración de la actividad, es decir:

$$H_{ij}^T = t_j^* - t_i - t_{ij}$$

La holgura total representa el número de unidades de tiempo en que puede retrasarse la realización de la actividad con respecto al tiempo PERT previsto, de manera que la duración del proyecto no se retrase. Es importante tener en cuenta el hecho de que, si una actividad consume una parte o el total de su holgura total, puede producir una disminución en la holgura total de la actividad siguiente.

Las actividades cuya holgura total sea cero se denominan *actividades críticas*. Uniendo todas las actividades críticas se forma un camino que va desde el vértice que representa el suceso inicio del proyecto al vértice que representa el suceso fin. Este camino recibe el nombre de camino crítico y resulta esencial para efectuar el control del proyecto. El responsable de la gestión del proyecto deberá extremar la vigilancia de estas actividades críticas, pues cualquier retraso en la realización de alguna de ellas provocará un demora en la finalización del proyecto. Por otra parte, no se deben desatender aquellas actividades no críticas, puesto que un retraso excesivo en su ejecución puede llegar a convertirlas en críticas, cambiando la estructura del camino crítico del grafo. Es importante tener en cuenta que el camino crítico es aquel de máxima longitud o tiempo que va desde el vértice que representa el suceso inicio del proyecto al vértice que representa el suceso fin. Por lo tanto, para calcular el camino crítico de un grafo se podrán aplican los algoritmos de teoría de grafos que permiten calcular el recorrido de máxima longitud o tiempo de un grafo. *

* Los algoritmos para determinar el camino crítico se pueden consultar, entre otros, en Meza y Ortega [12]

Establecimiento de un calendario de ejecución del proyecto

De los apartados anteriores podemos obtener información de gran utilidad para el responsable de gestionar un proyecto. De esta información es posible deducir un calendario de ejecución del proyecto, lo que constituye la pieza básica para efectuar el control del mismo. En este calendario, se consideran cuatro fechas para cada actividad: fecha de comienzo más temprana, fecha de comienzo más tardía, fecha de finalización más temprana y fecha de finalización más tardía.

La *fecha de comienzo más temprana* de una actividad (i,j) se representa como $\triangle_{ij}$ e indica lo más pronto posible que puede comenzar la actividad (i,j). Es sencillo deducir que esta fecha será igual al tiempo *early* del suceso inicio de la actividad, es decir:

$$\triangle^*_{ij} = t_i \tag{1.1}$$

La *fecha de comienzo más tardía* de una actividad (i,j) se representa como $\triangle^*_{ij}$ e indica lo más tarde que puede comenzar a ejecutarse la actividad (i,j) de manera que la duración prevista del proyecto no se retrase. Se calcula como la suma del tiempo *early* del suceso inicial y la holgura total de la actividad, es decir:

$$\triangle^*_{ij} = t_i + H^T_{ij} = t^*_j - t_{ij} \tag{1.2}$$

La *fecha de finalización más temprana* de una actividad (i,j) se representa como $\bigtriangledown_{ij}$ e indica lo antes que puede finalizarse la ejecución de la actividad (i,j). Se calcula como la suma del tiempo *early* del suceso inicial y el tiempo PERT previsto para esa actividad, es decir:

$$\bigtriangledown_{ij} = t_i + t_{ij} \tag{1.3}$$

Por último, la *fecha de finalización más tardía* de una cierta actividad (i,j) se representa como $\bigtriangledown^*_{ij}$ e indica la fecha tope en la que puede concluir la actividad (i,j), de manera que la duración prevista del proyecto no se retrase en ninguna unidad de tiempo. Obviamente, ésta será la fecha *last* del suceso final de la actividad, es decir:

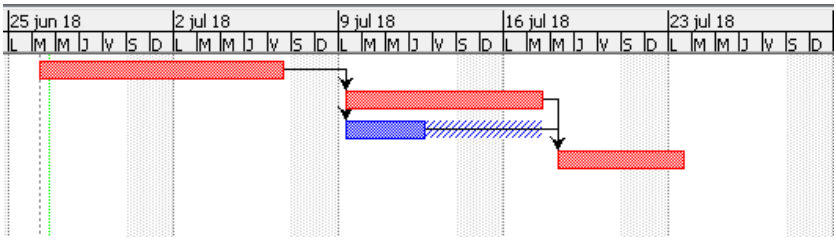
$$\bigtriangledown^*_{ij} = t^*_j \tag{1.4}$$

Cuando (i,j) es una actividad crítica, su holgura total es nula, por lo que las fórmulas (1.1), (1.2), (1.3) y (1.4) coinciden. Por otra parte, es sencillo comprobar que la diferencia entre las fechas de comienzo más tardía y más temprana es igual a la diferencia entre la fecha de finalización más tardía y más temprana, y además, coincide con el valor de la holgura total de la actividad (es decir, $\triangle^*_{ij} - \triangle_{ij} = \bigtriangledown^*_{ij} - \bigtriangledown_{ij} = H^T_{ij}$).

A partir de estas fórmulas se puede establecer un diagrama-calendario de ejecución del proyecto.

Existen numerosos paquetes informáticos dedicados a la programación de proyectos que se pueden emplear para obtener de manera sencilla una representación gráfica de este calendario. En el caso práctico abordado en este trabajo, se ha utilizado el paquete informático ProjectLibre. El programa ProjectLibre es un software de administración de proyectos con múltiples ventajas, entre las que destacan, por un lado, que es un software gratuito con código abierto y, por otro lado, es compatible con los sistemas operativos más populares.

En la siguiente figura se muestra un ejemplo de este diagrama en ProjectLibre:



2

Estructura general de la Planificación de Tareas

2.1. Planificación de Tareas: Preliminares

La planificación y secuenciación de actividades se desarrolló fundamentalmente en la Segunda Guerra Mundial. Debido a los esfuerzos bélicos, los gobiernos instaron a que sus científicos estudiaran y analizaran la asignación de los recursos militares limitados a las distintas maniobras y campañas militares. Tras la guerra, fueron muchos los equipos de científicos que vieron la utilidad de todo lo estudiado y desarrollado en las aplicaciones militares hacia distintos ámbitos de la vida civil. Así fue como, en los años 50 comenzaron a desarrollarse procedimientos y algoritmos orientados a la secuenciación correcta de las actividades como los de Johnson [9] y Jackson [8]. Con la aparición de los ordenadores, las posibilidades de desarrollo de nuevos métodos y técnicas crecieron notablemente.

Los problemas de planificación de tareas se pueden clasificar de acuerdo a ciertos criterios. Pinedo [14] distingue los modelos de secuenciación en deterministas o estocásticos. Los modelos deterministas son aquellos en los que todos los datos del problema de planificación son conocidos a priori. En cambio, los modelos estocásticos tienen en cuenta el componente aleatorio de los sucesos. Así, para una entrada de datos y unas condiciones iniciales, la salida del modelo será diferente cada vez que se resuelva el mismo. Es obvio que la naturaleza de los modelos de secuenciación tiene cierto componente estocástico, pero estos modelos son considerablemente más complicados. Incluso en el caso de los problemas deterministas, que son relativamente fáciles de plantear, encontramos grandes dificultades para modelizarlos y, consecuentemente, resolverlos.

La mayor parte de la investigación sobre problemas de secuenciación y planificación se ha centrado en resolver cuestiones referentes a la planificación determinística, desarrollando modelos y técnicas para solucionarlos. Sin embargo, no se trata de un campo ya cerrado, pues continuamente aparecen nuevos

métodos y procedimientos que mejoran las técnicas existentes. Estos modelos se basan en diversas suposiciones:

La primera suposición alude a los recursos y actividades. Una máquina es un recurso que puede ejecutar una única actividad en cada instante de tiempo. Y del mismo modo, en general, cada actividad o trabajo puede ser procesado en una única máquina en cada instante. Se asume que el tiempo de procesamiento de cada trabajo en una máquina es independiente de la secuenciación y se permite que las máquinas estén ociosas.

La segunda consideración hace referencia a la naturaleza determinística de los problemas. Toda la información que define la entrada del problema se conoce con certeza, y puesto que el número de posibles planificaciones es finito, la Planificación Determinística es una parte de la Optimización Combinatoria.

2.2. Formulación de los problemas de Planificación de Tareas

De manera general, para analizar los problemas de planificación se denota al conjunto $M = \{M_i | i = 1, ..., m\}$ como el conjunto de m máquinas que han de realizar n trabajos o tareas pertenecientes al conjunto $J = \{J_j | j = 1, ..., n\}$. En adelante, siempre que no haya lugar a confusión, podemos simplificar la notación y citaremos 'trabajo j' en lugar de J_j o bien 'máquina i' en lugar de M_i.

Una secuenciación consiste en la asignación de un trabajo j en uno o varios intervalos de unidades temporales a una o varias máquinas, i. Para representar secuenciaciones de manera gráfica se puede emplear un diagrama de Gantt, que puede ser orientado a máquinas o a trabajos, como podemos ver en la figura 2.1.

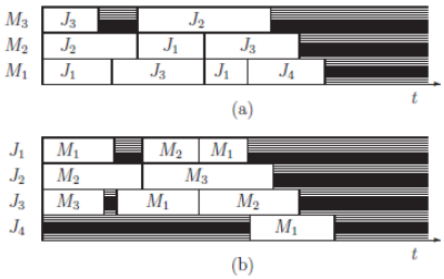


Figura 2.1. Diagrama de Gantt orientado a máquinas (a) o a trabajos (b)

A su vez, para cada uno de los trabajos se debe disponer de la siguiente información:

- Un *número* m_j *de operaciones* $O_{1j}, O_{2j}, ..., O_{m_j j}$ en las que puede dividirse el trabajo J_j. Una vez fijada la planificación, cada una de las operaciones O_{ij} puede ser procesada en una única máquina. Se puede definir $\mu_{ij} = k$, si la operación O_{ij} del trabajo j será procesada en la máquina k. Como caso particular, cuando el trabajo j requiere de una única operación, se tiene que $m_j = 1$ y en este caso el hecho de que dicha operación debe asignarse a la máquina M_k se denota por $\mu_j = k$.
- *Tiempos de procesamiento* de cada trabajo J_j. De manera general, denominamos p_{ij} al tiempo que requiere la máquina i para procesar el trabajo j. Cuando dicho tiempo es independiente de la máquina que lo procese, es decir, si $p_{ij} = p_j \ \forall i = 1, ..., m$, podemos simplificar dicha notación y se considera p_j.
- Una *fecha de disponibilidad* r_j a partir de la cual puede comenzar a procesarse el trabajo J_j. Cuando $r_j = r$, $\forall j = 1, ..., n$ todos los trabajos pasan a estar disponibles al mismo tiempo y nos encontramos ante un problema de planificación estático. En caso contrario, se tratará de un problema de planificación dinámico.
- Una *fecha de comienzo* S_j que depende de la planificación e indica el instante de tiempo en el que comienza a procesarse el trabajo J_j. Por definición, se cumple que la fecha de comienzo de un trabajo nunca será menor que su fecha de disponibilidad $r_j \leq S_j$, $\forall j$. El tiempo que transcurre desde que un trabajo J_j está disponible hasta que comienza a procesarse se representa por la diferencia $a_j = S_j - r_j$.
- Una *fecha límite o de vencimiento* d_j, que indica el instante en el que la última operación de un trabajo debe ser terminada sin incurrir en penalización o coste.
- Una *fecha de finalización* C_j que depende de la planificación e indica el instante de tiempo en el que termina de procesarse el trabajo J_j. Podemos estudiar la demora de un determinado trabajo estudiando la diferencia $L_j = C_j - d_j$. Nótese que cuando un trabajo finaliza antes de la fecha límite, dicho valor es negativo. Se define entonces la tardanza como $T_j = max\{L_j, 0\}$ y la precocidad como $E_j = max\{-L_j, 0\}$.
- Un *peso ponderado* w_j que indica la prioridad de un trabajo j frente a otros. Este valor podría afectar a la disposición de operaciones en una máquina, pero no será una prioridad absoluta, dependerá de la magnitud que se le otorgue.
- Una *función de costo real* f_{kj} por cada criterio k con $1 \leq k \leq K$, donde $f_{kj}(t)$ es el coste asociado, según el criterio k, al trabajo J_j cuando éste se completa en el instante de tiempo t.

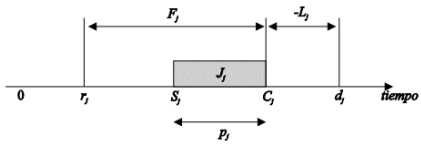


Figura 2.2. Datos y variables asociadas a un trabajo

Con estas definiciones se puede determinar otros parámetros de interés, como C_{max}, denominado *makespan*, que representa la fecha de finalización de todos los trabajos, es decir, el mínimo instante en el que todos los trabajos se han completado. Así, por ejemplo, se puede determinar el tiempo en el que una máquina M_j está ocioso como $I_j = C_{max} - \sum_{i=1}^{n} p_{ij}$. Nótese que esta definición tiene sentido, puesto que C_{max} es el instante en el que todos los procesos cesan y $\sum_{i=1}^{n} p_{ij}$ es el tiempo total de procesamiento en la máquina M_j.

2.3. Clasificación de los problemas de Planificación.

Será conveniente disponer de una notación simple que permita representar los diversos tipos de problemas de planificación. Una de las más extendidas actualmente es la nomenclatura propuesta por Graham et al.[6] que tiene en cuenta tres componentes. Se trata de una clasificación triparamétrica en la que se puede catalogar los diferentes problemas de planificación atendiendo a tres campos $\alpha|\beta|\gamma$. En el parámetro α se recogen las características que describen las m máquinas o procesadores $M_i(i = 1, ..., m)$; en el parámetro β las propiedades de los n trabajos o tareas a procesar $J_j(j = 1, ..., n)$ y finalmente, el último parámetro γ hace referencia al criterio de optimización considerado (ver Alcaide [1]). A su vez, cada campo es susceptible de dividirse en subcampos que permiten describir diferentes características de las máquinas, los trabajos o los criterios de optimización.

2.3.1. Características de las máquinas.

Podemos distinguir los problemas de planificación y secuenciación en los que existe una única máquina de aquellos en los que existen varias máquinas. Desde el momento en el que existan varias máquinas, existe la posibilidad de que todas puedan ejecutar las mismas funciones (máquinas no especializadas) o bien haya máquinas que se especializan en ciertas tareas y no pueden procesar otras. El valor del parámetro α será 1, P, Q, R, F, O ó J dependiendo de las diferentes clases de máquinas.

Máquinas no especializadas

En este caso se considera que cada trabajo consta de una única operación ($m_j = 1$) que puede ser procesada en cualquier máquina. Sin embargo, pueden existir diferencias entre el tiempo de procesamiento de una máquina a otra.

$\alpha = 1$. Se dispone de una única máquina, $m = 1$, y por tanto el tiempo de procesamiento $p_{1j} = p_j, \forall j = 1, 2, ...n$ indicaría el tiempo de procesamiento en dicha máquina.

$\alpha = P$. Se dispone de m máquinas no especializadas e idénticas, es decir, todas ellas cuentan con la misma velocidad de procesamiento. Entonces, el tiempo de procesamiento del trabajo $J_j, \forall j$, no depende de la máquina que lo procese y así, se tiene que $p_{ij} = p_j, \forall i = 1, ..., m$.

$\alpha = Q$. Se dispone de m máquinas no especializadas en paralelo y uniformes, esto es, disponen de diferentes velocidades de proceso; q_i indica la velocidad constante de la máquina M_i, que no depende de J_j. El tiempo de procesamiento del trabajo J_j es $p_{ij} = p_j/q_i$ asumiendo que sólo se procesa en la máquina i. Nótese que si todas las máquinas cuentan con la misma velocidad el problema se reduce al caso previo.

$\alpha = R$. Se dispone de m máquinas no especializadas en paralelo y no relacionadas entre sí. En este caso, la velocidad de proceso depende no sólo de la máquina, sino también del trabajo. Nótese que, si la velocidad de las máquinas es independiente de los trabajos, el problema se reduce al caso previo.

Máquinas especializadas

En este caso, no todas las máquinas pueden procesar todas las tareas, sino que están especializadas en alguna de ellas. Por tanto, el número de operaciones de cada trabajo no tiene por qué ser necesariamente 1, sino que es posible dividir cada trabajo J_j en m_j operaciones $O_{1j}, O_{2j}, ..., O_{mj}$ que pueden ser realizadas en diferentes máquinas. La clasificación en este caso dependerá de la relevancia del orden en que se realicen las operaciones.

$\alpha = F$ (Flow-shop). Se dispone de m máquinas en serie, donde la operación O_{ij} se procesa en la máquina M_i en un tiempo p_{ij}. En este caso, dicho orden es relevante y siempre el mismo, es decir, todos los trabajos deben seguir la misma trayectoria de máquinas, el mismo patrón de flujo. Puede ocurrir que un trabajo J_j no precise pasar por todas las máquinas. Para indicar que el trabajo J_j no requiere ser procesado por la máquina M_i se considera $p_{ij} = 0$.

$\alpha = O$ (Open-shop). Se dispone de m máquinas, donde cada trabajo J_j consiste en una serie de $m_j = m$ operaciones $O_{1j}, O_{2j}, ..., O_{mj}$. Cada operación O_{ij} se procesa en la máquina M_i en un tiempo p_{ij}, es decir, cada trabajo debe ser procesado en cada una de las m máquinas. A diferencia de los problemas flow-shop, no hay restricción en relación a la ruta de cada trabajo a través

de las máquinas, el orden en que se realicen las operaciones es irrelevante. El plan determina la ruta que debe seguir cada trabajo y diferentes trabajos pueden tener distintas rutas.

$\alpha = J$ (Job-shop). Se dispone de m máquinas, donde cada trabajo J_j consiste en una serie de m_j operaciones $O_{1j}, O_{2j}, ..., O_{m_j j}$. A diferencia del caso anterior, no tiene por qué ser $m_j = m$. Cada operación O_{ij} se procesa en la máquina M_i en un tiempo p_{ij}, y la trayectoria de las máquinas de cada trabajo está dada, pero no tiene por qué ser la misma para todos los trabajos. Una máquina puede procesar dos o más operaciones de un mismo trabajo. Es lógico pensar que no se ejecutarían dos operaciones seguidas pues, en este caso, dichas operaciones podrían considerarse como una sola. Es decir, $\mu_{i-1,j} \neq \mu_{i,j}\, \forall i = 2, ..., m_j$ y $\forall j = 1, ..., n$.

2.3.2. Características de los trabajos.

Las restricciones impuestas por los trabajos se especifican en el campo β y pueden incluir múltiples entradas $\beta = \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6$. Estas posibles entradas son:

Interrupciones

El símbolo $\sqcap$ indicará que el parámetro se deja en blanco sin especificar nada. Los valores de β_1 significarán entonces:

$\beta_1 = \sqcap$**.** Cuando no se indica nada, los trabajos no pueden ser interrumpidos en el proceso.

$\beta_1 = pmtn$**.** Los trabajos pueden ser interrumpidos y continuarse más tarde en el estado en que se dejó.

Relaciones de precedencia

Estas relaciones se establecen cuando uno o más trabajos deben ser completados antes de que a otro trabajo se le permita iniciar su proceso.

$\beta_2 = \sqcap$**.** No hay relaciones de precedencia entre los trabajos.

$\beta_2 = prec$**.** Existe una relación de precedencia entre los trabajos que puede ser representada por un grafo acíclico dirigido $G = (V, A)$, donde el conjunto de vértices $V = \{1, ..., n\}$ corresponde a los trabajos $J_j (j = 1, ..., n)$ y el arco (j, k) indica que el trabajo J_j debe ser completado antes de comenzar a procesar J_k. De forma análoga, cuando un trabajo consta de más de una operación se puede plantear un grafo G dirigido acíclico cuyos vértices representan las operaciones $O_{ij} (j = 1, ..., n; i = 1, ..., m_j)$ y los arcos la relación descrita anteriormente.

$\beta_2 = tree$**.** El grafo de precedencias G es un árbol. El indicativo *intree* indica que de cada vértice sale a lo sumo un arco, salvo del vértice raíz. Mientras que el indicativo *outtree* significa que a cada vértice llega a lo sumo un arco, salvo el vértice raíz. Dicho árbol será un árbol de ensamble ó un árbol de ramificación, respectivamente.

Fechas de disponibilidad

Sabemos que el trabajo J_j no puede iniciar su procesamiento antes de su fecha de disponibilidad r_j, esta información se recoge en el parámetro β_3.

$\beta_3 = \sqcap$**.** El procesamiento de cualquier trabajo puede comenzar en cualquier momento, indicando una planificación estática. Este caso engloba tanto cuando todos los trabajos están disponibles desde el instante inicial $r_j = 0$, $\forall j$ o bien desde un determinado momento $r_j = r\, \forall j$.

$\beta_3 = r_j$. Las fechas de disponibilidad no tienen por qué ser iguales para todos los trabajos, indicando una planificación dinámica.

Cotas al número de operaciones

En el ambiente job-shop el número m_j de operaciones de un trabajo J_j puede ser superior al número m de máquinas, por lo que un trabajo puede visitar una máquina en más de una ocasión. En el desarrollo de algunos algoritmos o demostraciones de que problemas job-shop son NP-duros se utiliza el hecho de que dicho número m_j está acotado superiormente por una constante m_{UB}.

$\beta_4 = \sqcap$. No se conoce ninguna cota para m_j.
$\beta_4 = m_j \leq m_{UB}$. Se especifica una cota superior para m_j.

Tiempos de proceso

Este parámetro resulta necesario ya que muchos problemas no son resolubles con tiempos de proceso generales p_{ij}, pero se pueden resolver cuando consideramos tiempo de proceso unitarios.

$\beta_5 = \sqcap$. Los p_{ij} son enteros arbitrarios.
$\beta_5 = p_{ij} = 1$. El tiempo de proceso de cada operación es unitario.

Recursos adicionales

La incorporación de los recursos adicionales es adecuada en sistemas de computadores, así como en aquellas situaciones en las que la realización de las tareas en las máquinas precise de otros recursos limitados como mano de obra, espacio de almacenamiento, etc. La forma de incorporar estos recursos adicionales al modelo es mediante el parámetro β_6.

$\beta_6 = \sqcap$. No existen recursos adicionales.
$\beta_6 = res\lambda\sigma\rho$. Existen recursos adicionales. En este caso, λ indica el número de recursos adicionales, σ las unidades disponibles y ρ las unidades de cada recuso adicional que se requieren. Puede darse el caso de que $\lambda, \sigma, \rho = \bullet$, lo que significa que son número arbitrarios.

2.3.3. Criterios de optimalidad.

El tercer campo hace referencia al criterio de óptimalidad elegido y sus características. Denominamos γ el criterio considerado, que será en general una función creciente de los tiempos de completación de los trabajos. Usualmente se trabaja con funciones de la forma $\gamma = f_{max}$ ó $\gamma = \sum f_j$. Recordemos que, dada una secuenciación, podemos determinar para cada trabajo:

- El tiempo de completación C_j, que indica el instante en el que termina de procesarse el trabajo J_j.
- La demora $L_j = C_j - d_j$, que indicará la diferencia del tiempo en el que se completa el trabajo con respecto a la fecha de vencimiento. Si es positivo, ha habido un retraso respecto a cuando debería haber finalizado el trabajo, mientras que, si es negativo, se ha producido un adelanto.
- La tardanza que viene dada por $T_j = max\{0, L_j\}$ e indica el retraso habido en el procesamiento del trabajo J_j.
- El indicador de trabajo tardío, que valdrá 1 si el trabajo no se concluye antes de su fecha de vencimiento, es decir:

$$U_j = \begin{cases} 1, & \text{si } C_j > d_j, \\ 0, & \text{en otro caso.} \end{cases}$$

Los criterios de óptimalidad más comúnmente empleados involucran la minimización de $f_{max} = \{C_{max}, L_{max}\}$ donde $f_{max} = max\{f_j(C_j)\}$, siendo $f_j(C_j) = C_j$ ó L_j, es decir, se trata de minimizar el máximo tiempo de completación o de demora, respectivamente. Si se hace referencia al coste total se deberá minimizar

$$\sum f_i \in \{\sum C_j, \sum T_j, \sum U_j, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$$

donde $\sum f_j = \sum_{i=1}^{n} f_j C_j$ con $f_j(C_j) = C_j, T_j, U_j, w_j C_j, w_j T_j, w_j U_j$.

Los criterios $\sum w_j C_j$ y $\sum w_j L_j$ difieren por una constante $\sum w_j d_j$, por lo que son equivalentes. Además, cualquier secuenciación que minimice L_{max} minimizará T_{max} y U_{max}, pero no viceversa.

La relación de complejidad de los problemas de planificación para los posibles valores de los parámetros α, β y γ comentados anteriormente, los podemos esquematizar en los siguientes grafos:

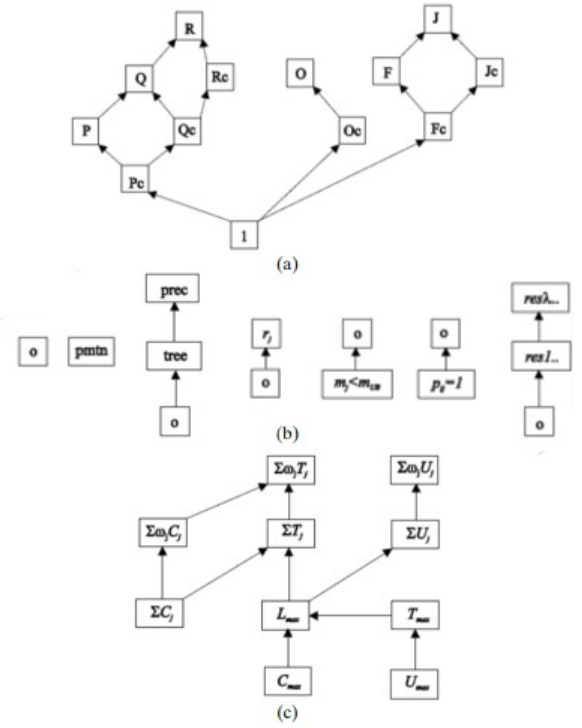


Figura 2.3. Jerarquía de complejidad de los problemas de planificación determinista: (a) máquinas, (b) trabajos, (c) funciones objetivo.

Para finalizar, citamos a continuación un par de ejemplos que describen la configuración del problema de planificación que se desea analizar.

Ejemplo 2.1. $F_m|p_{ij} = p_j|w_jC_j$ denota un entorno flow-shop con m máquinas en serie; los tiempos de procesamiento del trabajo J_j son idénticos e iguales a p_j en las m máquinas. El objetivo es encontrar el orden en que los n trabajos deben pasar por el sistema para que la suma de los tiempos de finalización ponderados se reduzca al mínimo.

Ejemplo 2.2. $J_m||C_{max}$ denota un problema job-shop con m máquinas. No hay recirculación, por lo que un trabajo visita cada máquina como máximo una vez. El objetivo es minimizar el makespan.

2.4. Complejidad Computacional

En esta sección se estudiarán las clases en las que se pueden clasificar los problemas de optimización combinatoria desde el punto de vista de la complejidad. Esta clasificación es sumamente importante en la práctica, ya que proporciona una idea a priori de la dificultad del problema. Debido a que este ámbito es muy extenso, se abordarán los conceptos fundamentales a título informativo, con el objetivo de realizar un encuadre general de los problemas de planificación de tareas.

En general, para poder resolver problemas no triviales se debe desarrollar una metodología que pueda computar la solución a dicho problema. Para ello, se implementa un método o procedimiento de solución en una computadora, lo que se conoce como algoritmo.

Un algoritmo consiste en un conjunto ordenado de operaciones sistemáticas, que permite hacer un cálculo y hallar la solución de un problema. Dados un estado inicial y unos parámetros de entrada, siguiendo los pasos sucesivos se llega a un estado final, obteniéndose así dicha solución.

Para que un algoritmo pueda resolver un ejemplo o entrada de un problema de optimización, debe tener las siguientes propiedades:

- Validez: el algoritmo es válido si converge a una solución que puede ser óptima o aproximada a la óptima.
- Finitud: el algoritmo debe terminar su ejecución en un número finito de pasos.

Supongamos que tenemos un problema y hay dos algoritmos que lo resuelven, en este caso, la mejor elección será aquel que resulte más eficiente. La complejidad computacional es precisamente una medida de los recursos que requiere su ejecución en función del tamaño de la entrada del problema.

Los recursos comúnmente estudiados en complejidad computacional son: el tiempo, mediante una aproximación al número de pasos de ejecución que un algoritmo emplea para resolver un problema; y el espacio, mediante una aproximación a la cantidad de memoria utilizada para determinar la solución.

De este modo, como hemos dicho, la complejidad computacional de un problema se mide en función de los recursos consumidos por el algoritmo para resolverlo, es decir, será medida como una función tal que, para cada entrada de tamaño n, nos da la cantidad de recursos requeridos por el algoritmo correspondiente a esas entradas. La eficiencia del algoritmo se mide por un límite superior $T(n)$ en el número de pasos que toma el algoritmo para cualquier entrada de tamaño n.

Es un resultado conocido en la teoría de la Complejidad Computacional que todo problema de optimización tiene asociado un problema de decisión, con dos respuestas posibles 'Sí' o 'No'; y un problema de reconocimiento del lenguaje. La idea para estudiar la complejidad de un problema de optimización es sencilla: resolver un problema se reduce a encontrar un algoritmo adecuado que resuelva los problemas asociados.

Los problemas de reconocimiento del lenguaje se resuelven con las denominadas máquinas de Turing, que pueden ser determinísticas y no determinísticas. Así, los problemas de clase P son aquellos cuyos problemas de reconocimiento del lenguaje asociados pueden resolverse con una máquina de Turing determinística en tiempo polinomial. De manera intuitiva, consideramos que los problemas contenidos en P son aquellos que pueden ser resueltos en forma razonablemente rápida y suelen ser ejecutados en la práctica.

Por otro lado, los problemas de la clase NP son aquellos cuyo problema de reconocimiento del lenguaje asociado se resuelve polinomialmente con una máquina de Turing no determinísta. Para resolver un problema de clase NP, se puede ejecutar un método de resolución no determinista, que consiste en aplicar heurísticos para obtener soluciones hipotéticas que se van desestimando o aceptando a un ritmo polinomial. Muchos problemas de planificación pertenecen a la clase NP.

Existe una amplia gama de problemas de tipo NP de los cuales destacan algunos de extrema complejidad, denominados NP-completos. Gráficamente, se puede decir que algunos de estos problemas se encuentran en la 'frontera extrema' de la clase NP y son catalogados los peores problemas posibles de clase NP. Por otro lado, la clase de complejidad NP-duro o *NP-hard* puede ser descrita como aquella que contiene a los problemas que son como mínimo tan difíciles como un problema de NP. Sin embargo, demostrar que un modelo sea de un tipo u otro no es una tarea trivial. Si el problema es NP-duro, como ocurre en la mayoría de los problemas prácticos, se pueden adoptar dos aproximaciones diferentes a la solución. La primera se basa en elegir algún método exacto para resolver el problema, pero tiene la dificultad que si el problema es de grandes dimensiones su resolución necesitaría mucho tiempo de cálculo y capacidad de memoria. La segunda opción requerirá encontrar un método heurístico rápido para encontrar una solución aproximada y realizar un análisis comparativo de contraste de la calidad de la solución obtenida. Esta opción está justificada en muchos de los problemas de planificación debido a la elevada complejidad que presentan la mayoría de ellos.

Intuitivamente, si para un determinado problema resulta sencillo encontrar una solución, entonces se podrá comprobar si un determinado resultado verifica las condiciones del problema, por lo que P es un subconjunto de la clase NP, $P \subset NP$. La gran cuestión es si ocurre lo mismo a la inversa, es decir, si se puede calcular una solución fácilmente para un problema en el que determinar si un resultado es una solución es sencillo. Si esto fuese así, entonces se tendría la igualdad $P = NP$. Sin embargo, la cuestión de la inclusión estricta entre las clases de complejidad P y NP es uno de los problemas abiertos más importantes de las matemáticas.

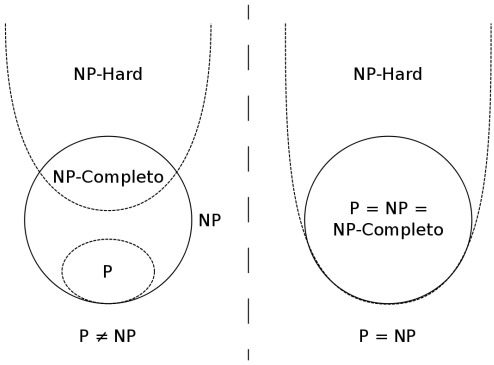


Figura 2.4. Diagrama de Euler de las familias de problemas P, NP, NP-completo, y NP-duro.

3

Problemas de planificación sobre máquinas en paralelo

3.1. Introducción al problema de máquinas paralelas

En este capítulo se estudiarán problemas de planificación de tareas en los que existen varias máquinas en paralelo disponibles para procesar los trabajos. Este tipo de problemas son de gran importancia desde un punto de vista teórico y práctico. Desde una perspectiva teórica, pueden considerarse como una generalización de los problemas con una única máquina. Por otro lado, desde un enfoque práctico resultan relevantes debido a que los problemas de planificación con recursos paralelos son frecuentes en el mundo real.

Los problemas de planificación sobre máquinas paralelas se pueden descomponer en dos procesos: en el primero, se aborda un problema de asignación para determinar qué trabajos debe realizar cada máquina; y en el segundo, se estudia un problema de secuenciación en el que se precisa la secuencia de trabajos asignados a cada máquina.

A lo largo de este capítulo consideraremos sistemas en los que todos los procesadores son idénticos, por lo que cualquier máquina es capaz de realizar cualquier tarea y disponen de la misma velocidad de proceso. El objetivo considerado es minimizar el '*makespan*' es decir, el tiempo mínimo en el que todos los trabajos se han completado. En la práctica, a menudo se tiene que hacer frente al problema de equilibrar la carga de las máquinas en paralelo. Por ello, éste es un objetivo de considerable interés, ya que minimizando el makespan de un conjunto de máquinas en paralelo se asegura un buen equilibrio.

La clasificación computacional de los problemas sobre máquinas no especializadas varía dependiendo de si los trabajos son interrumpibles o no. Los problemas de planificación en los que los trabajos no se pueden suspender suelen ser difíciles, mientras que, permitir interrupciones simplifica generalmente el análisis del problema. Por ello, se estudiarán por un lado los problemas sin interrupciones y, por otro, los problemas con interrupciones.

3.2. Planificación sin interrupciones

En primer lugar, consideraremos el problema $P_m||C_{max}$, que trata de asignar n tareas a m procesadores iguales de modo que cada trabajo se asigna a exactamente un procesador y, en un instante de tiempo dado, cada procesador sólo puede ejecutar un único trabajo. Este problema es NP-duro en sentido fuerte. Por lo tanto, para determinar una planificación óptima se requieren métodos generales de optimización como la ramificación y acotación o la programación dinámica. En el caso de la ramificación y acotación, no es sencillo obtener cotas inferiores ajustadas; en el caso de la programación dinámica, el número de estados tiende a ser extremadamente grande cuando se trabaja con más de dos máquinas. Como estos algoritmos exactos no tienen mucho éxito, excepto en problemas relativamente pequeños, debido a requerir grandes recursos computacionales y temporales, se pueden emplear algunos procedimientos heurísticos que funcionan bastante bien.

Una de estas heurísticas es la regla LPT (*Longest Processing Time*). Esta regla asigna en el instante inicial los m trabajos más largos a las m máquinas disponibles. Después de esto, cada vez que una máquina queda libre se le asigna el trabajo más largo de entre aquellos que aún no han sido procesados. Esta heurística intenta colocar los trabajos más cortos al final de la planificación, donde pueden ser empleados para equilibrar las cargas. El siguiente resultado establece que a lo sumo la regla LPT puede superar el caso óptimo en un 33′33 %.

Lema 1 *Sea $C_{max}(LPT)$ el makespan obtenido empleando la heurística LPT para un problema $P_m||C_{max}$. Consideremos C^*_{max} el makespan óptimo. Entonces, se tiene que:*

$$\frac{C_{max}(LPT)}{C^*_{max}} \leq \frac{4}{3} - \frac{1}{3m}$$

Al añadir relaciones de precedencia al problema de minimizar el makespan en máquinas paralelas, éste no se simplifica. El problema $P_m|prec|C_{max}$ es NP-duro en el sentido fuerte. Incluso el caso con tiempos de procesamiento unitarios, $P_m|prec, p_j = 1|C_{max}$, no resulta sencillo. No es una sorpresa, entonces, que sólo se tengan resultados para casos especiales. Así, cuando se asume que el grafo de precedencias toma la forma de un árbol, es decir, $P_m|tree|Cmax$, el problema es fácilmente solucionable.

Introduciremos el caso especial en el que las relaciones de precedencia son en forma de árbol *intree* con lo que ningún trabajo tiene más de un sucesor directo. En dicho árbol, el trabajo final, para el que no existen sucesores, se denomina trabajo terminal. Además, consideraremos tiempos de procesamiento unitarios para todos los trabajos. Estamos entonces frente al problema $P|tree = intree, p_j = 1|C_{max}$, que puede resolverse en tiempo $O(n)$ por el algoritmo de Hu [7]. Este algoritmo se desarrolla en dos etapas: etapa de etiquetado y etapa de planificación.

Algoritmo de Hu

Etapa de etiquetado

Paso 1. Asignar la etiqueta cero al nodo correspondiente al trabajo terminal.
Paso 2. Suponiendo que las etiquetas 1,2,..., j-1 han sido asignadas. Asigna la etiqueta j a los trabajos que no tiene sucesores sin etiquetar.
Paso 3. Repetir el paso anterior hasta que todos los trabajos hayan sido asignados a etiquetas.

Etapa de planificación

Paso 1. Crear una lista de los trabajos en orden de etiqueta decreciente, en la medida en la que las restricciones de precedencia lo permitan.
Paso 2. Asignar el primer trabajo de la lista a la primera máquina que quede libre hasta que la lista quede vacía.

De este modo, en la fase de etiquetado se asigna a cada trabajo una etiqueta igual al tiempo requerido para procesar las tareas que siguen al trabajo j en la única ruta que conecta el trabajo j y el trabajo terminal (ver Baker y Trietsch [2]). Nótese que, cuando la fase de planificación coloca los trabajos con las etiquetas más grandes en el cronograma, esencialmente da prioridad a los trabajos que inician las rutas más largas en el árbol restante.

En la siguiente figura se muestra un ejemplo del algoritmo del algoritmo de Hu:

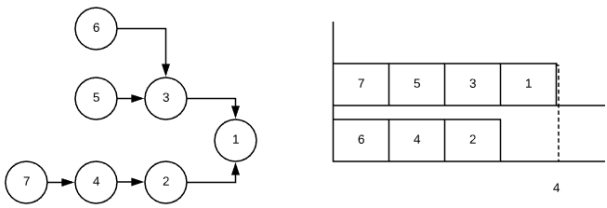


Figura 3.1. Un ejemplo de implementación del Algoritmo de Hu.

El algoritmo de Hu proporciona una planificación óptima cuando el problema cuenta con tareas que tienen tiempos de procesamiento unitarios y una estructura de árbol. Aunque un árbol tiene un único trabajo terminal, podemos aplicar el algoritmo a la programación de varios árboles creando un trabajo terminal ficticio para que sirva como sucesor de los trabajos terminales de cada uno de los árboles. Si asignamos la etiqueta cero al trabajo ficticio, cada etiqueta representa el tiempo restante en la ruta directa desde el nodo hasta la finalización.

3.3. Planificación con interrupciones

Consideremos el mismo problema con el que iniciamos la sección anterior, permitiendo ahora que se produzcan interrupciones, es decir $P_m|pmtn|C_{max}$. Generalmente, permitir que se produzcan interrupciones simplifica el análisis del problema. En este caso, veremos que algunos de los resultados estudiados para problemas en los que no se permiten interrupciones pueden aplicarse también cuando los trabajos pueden ser interrumpidos y continuarse más tarde. Comenzaremos presentando el modelo de programación lineal del problema:

$$minimizar\ C_{max}$$

sujeta a:

$$\sum_{i=1}^{m} x_{ij} = p_j \qquad \forall j = 1, ..., m \tag{3.1}$$

$$C_{max} - \sum_{i=1}^{m} x_{ij} \geq 0 \qquad \forall j = 1, ..., n \tag{3.2}$$

$$C_{max} - \sum_{j=1}^{n} x_{ij} \geq 0 \qquad \forall i = 1, ..., m \tag{3.3}$$

$$x_{ij} \geq 0 \tag{3.4}$$

La variable x_{ij} representa el tiempo total que el trabajo j pasa en la máquina i. los El primer conjunto de restricciones (3.1) asegura que cada trabajo reciba la cantidad requerida de procesamiento. El segundo conjunto de restricciones (3.2) asegura que la cantidad total de procesamiento recibido por cada trabajo es menor o igual que el makespan. El tercero (3.3) asegura que la cantidad total de procesamiento en cada máquina es menor que el makespan.

Este problema de programación lineal puede ser resuelto en un tiempo polinomial. La solución no proporciona una planificación, tan sólo especifica la cantidad de tiempo que debe pasar el trabajo j en la máquina i. Sin embargo, con esta información, es sencillo determinar una planificación. Para ello, la regla de McNaughton [11] tiene en cuenta que una cota inferior a la solución óptima es:

$$max\{max_j\{p_j\}, \frac{1}{m}\sum_j p_j\}$$

Tener un límite inferior permite la construcción de un algoritmo muy simple que determina una planificación que alcance esta cota, es decir, una planificación óptima. El procedimiento consiste en ir llenando las máquinas sucesivamente, planificando en ellas los trabajos en cualquier orden e interrumpiendo

aquel trabajo que rebase la cota anterior. De esta manera, el número máximo de interrupciones que puede tener la planificación es $m-1$.

Si añadimos relaciones de precedencia el problema no resulta sencillo e incluso el caso especial con tiempos de procesamiento unitarios, es decir, $P|pmtn, prec, p_j = 1|C_{max}$, es NP-duro.

Sin embargo, cuando las relaciones de precedencia siguen una estructura de árbol, y consideramos $P|pmtn, intree|C_{max}$ el problema puede resolverse en tiempo polinomial por el algoritmo de Muntz y Coffman[13]. Este algoritmo se ayuda del concepto de *procesor sharing* que permite un uso compartido de los procesadores, con lo que a un trabajo dado se le puede asignar 'parte de una máquina'. Si se le asigna al trabajo J_j la capacidad del recurso θ $(0 \leq \theta \leq 1)$, el tiempo de procesamiento del trabajo en dicha máquina sería p_j/θ. Dicho de otro modo, θ indica el ratio en el que un trabajo es procesado por una máquina en particular. Si $\theta = 1$ la máquina se dedica solamente a ejecutar el trabajo. De este modo, el algoritmo de Muntz y Coffman para $P|pmtn, intree|C_{max}$ se desarrolla en dos etapas: en la primera etapa tiene lugar el etiquetado de los trabajos, como una generalización del algoritmo de Hu; posteriormente, en la segunda etapa, los trabajos se ordenan con orden decreciente de etiquetas. Considerando este orden, y actualizando convenientemente los ratios de procesamiento, se logra determinar una planificación óptima.

Algoritmo de Muntz y Coffman

Etapa de etiquetado

Paso 1. Asignar a cada trabajo terminal i la etiqueta p_i.

Paso 2. Etiquetamos el resto de trabajos con el tiempo requerido para procesar las tareas que lo siguen. Esto es, a cada trabajo J_j se le asigna la etiqueta.

$$\alpha_j = max\{\alpha_i / j \, predecesor \, directo \, de \, i\} + p_j$$

Al comenzar el algoritmo, p_j se interpretará como el tiempo de procesamiento del trabajo. Sin embargo, si J_j ha comenzado a procesarse y ha sido previamente interrumpido, p_j pasará a indicar el tiempo de proceso necesario para completarlo.

Etapa de planificación

Paso 3. Asignar una máquina a cada uno de los trabajos J_j con etiquetas maximales α_j. En caso de que se produzca un empate entre las etiquetas, donde k trabajos compiten por r máquinas $(r < k)$, asignar un ratio de capacidad de procesamiento r/k a cada uno de estos trabajos.

Paso 3. Esta asignación se mantiene hasta que ocurre alguna de las siguientes situaciones:

i) Se completa un trabajo.

ii) Existen dos trabajos, i y j, cuyas etiquetas verifican $\alpha_i \geq \alpha_j$ pero cuyos ratios cumplen que $\theta_i < \theta_j$ en la asignación considerada. Intuitivamente, esta situación indicaría que el trabajo J_j tiene mayor prioridad que el trabajo J_i, a pesar de que J_i que inicia una ruta más larga.

En cualquier caso, se deberá volver a la etapa de etiquetado para crear una nueva asignación, hasta completar todo el procesamiento.

De este modo, el algoritmo de Muntz y Coffman tiene complejidad $O(n^2)$ y produce planificaciones con, a lo sumo, $O(nm)$ interrupciones. El siguiente resultado analiza la efectividad del algoritmo:

Lema 2 *Sea $C_{max}(MC)$ el makespan calculado para una planificación obtenida empleando el algoritmo de Muntz y Coffman en un problema $P|pmtn, intree|C_{max}$. Denotaremos C^*_{max} el makespan óptimo. Entonces, se verifica que:*

$$\frac{C_{max}(MC)}{C^*_{max}} \leq 2 - \frac{2}{m}$$

Con los resultados expuestos, podemos afirmar que existen algoritmos que nos permiten resolver en tiempo polinomial el problema de minimizar el makespan en un sistema de planificación con máquinas paralelas idénticas cuando se permiten interrupciones. Sin embargo, en muchos casos, la planificación óptima se obtiene considerando un alto número de interrupciones, por lo que deja de ser interesante desde un punto de vista práctico. Esta cuestión es de gran interés en los problemas de planificación que afrontan las empresas actualmente, puesto que mientras que en los problemas de planificación tradicionales con múltiples procesadores el costo de las interrupciones era relativamente pequeño, en el entorno informático moderno, las interrupciones generalmente implican comunicación y, a veces, requieren la migración del trabajo realizado a través de una red o *network*. Esto puede aumentar significativamente el costo de la solución obtenida. Por lo tanto, es natural buscar itinerarios que minimicen el tiempo total de finalización, al tiempo que cometen una pequeña cantidad de interrupciones.

El problema que trata de determinar cuántas interrupciones (en total o por trabajo) bastan para garantizar una solución óptima en un tiempo polinomial sigue sin ser resuelto.

González y Johnson [4] desarrollaron un algoritmo capaz de obtener planificaciones óptimas para el problema $P|pmtn, tree|C_{max}$ con a lo sumo n-2 interrupciones. En comparación con el algoritmo de Muntz y Coffman, este proceso comienza por las raíces en lugar de por las hojas del árbol y determina las prioridades entre los trabajos considerando el tiempo total de procesamiento en subárboles en lugar de atender a los caminos críticos.

En el artículo publicado por González y Sahni [5] queda probado que todo sistema con m procesadores uniformes y n trabajos puede ser planificado

de manera óptima utilizando a lo sumo $2(m-1)$ interrupciones. Para ello, la prueba realizada es constructiva. En primer lugar, se desarrolla un algoritmo que genera una planificación óptima para el sistema considerado. Posteriormente, se demuestra que los resultados obtenidos tienen a lo sumo $2(m-1)$ interrupciones. La complejidad de tiempo del algoritmo resultante es $O(n + mlog(m))$ cuando se incluye el tiempo para ordenar las tareas y los procesadores por duración de la tarea y velocidad del procesador, respectivamente. Para el caso de los procesadores idénticos, el algoritmo se reduce al de McNaughton y, por lo tanto, el número máximo de interrupciones que puede tener la planificación es $m-1$.

Sin embargo, aún quedan varias preguntas abiertas para trabajos futuros. Por un lado, para los trabajos interrumpibles se trata de optimizar la cota para el número de interrupciones. La cuestión es ahora evaluar si existe un valor d tal que $2 < d < 2(m-1)$ de manera que el problema es solucionable en un tiempo polinomial y la cota máxima de interrupciones a_j para cada trabajo J_j sea menor o igual que d, es decir, $a_j \leq d, \quad \forall j$. Además, si bien el problema con máquinas indénticas y cotas al número de interrupciones arbitrarias a_j se ha logrado resolver, en el caso de tener máquinas uniformes el problema sigue abierto, incluso cuando para todos los trabajos J_j, la cota al número de interrupciones a_j es una constante. Por otro lado, en los procedimientos ya diseñados se pueden optimizar algunos pasos para lograr un tiempo de ejecución más eficiente (ver Shachnai, Tamir y Woeginger [16]).

4

Planificación de proyectos en una empresa de desarrollo tecnológico

En este capítulo aplicaremos los métodos expuestos anteriormente para planificar y programar las actividades de una empresa de desarrollo de soluciones tecnológicas. En la primera sección se realiza una descripción de la empresa, exponiendo sus objetivos, los recursos humanos disponibles y las características de los proyectos a desarrollar. Posteriormente, podremos construir una solución en la planificación que caracterice y represente adecuadamente las actividades de la empresa y su funcionamiento, con el fin sugerir posibles alternativas de actuación que incrementen su rendimiento.

4.1. Descripción de los objetivos y recursos disponibles de la empresa

La organización que ha colaborado en este estudio, facilitándonos los datos de las tareas a realizar y el personal disponible, se dedica a desarrollar, comercializar e implantar soluciones informáticas para gestionar de forma correcta una empresa, tanto en el ámbito financiero, contable, comercial, etc. Actualmente esta consultora radicada en Santa Cruz de Tenerife ofrece un amplio abanico de soluciones tecnológicas, entre los más demandados destacan los proyectos de ERP (*Enterprise Resource Planning*), CRM (*Costumer Relationship Management*) y Apps Móviles (*Mobile Application*).

El avance imparable de la digitalización en todas las fases y cadenas de valor, ha hecho que el uso de soluciones tecnológicas ya no sea un terreno reservado a las grandes corporaciones, sino que está al alcance de cualquier empresa de tamaño medio. Con ello, en los últimos años, la empresa ha incrementado notablemente el número de organizaciones interesadas en adquirir sus servicios y en muchas ocasiones se precisa de una cuidada planificación para gestionar de forma eficiente los proyectos demandados. El objetivo de la empresa es reducir

el tiempo necesario para completar los proyectos. Este es un objetivo de especial interés puesto que, por un lado, una gestión rápida podría mejorar la satisfacción del cliente; y por otro, aumentaría la capacidad de la empresa, permitiendo desarrollar un mayor número de soluciones.

Recursos humanos y materiales disponibles

La empresa está estructurada de manera funcional por lo que los empleados se agrupan en departamentos atendiendo al tipo de tareas que realiza cada uno de ellos. Habitualmente, un proyecto requiere la participación de varias áreas funcionales. La empresa dispone de 11 empleados que habitualmente trabajan en la realización de proyectos. Cada empleado deberá trabajar una media de 40 horas a la semana, de las cuales 35 se dedican al avance de los proyectos solicitados por los clientes y 5 a los periodos de descanso y desarrollo de actividades internas (formación, redacción de artículos, creación de contenido de redes sociales, etcétera).

Las funciones de los empleados se especifican en la siguiente tabla:

Funciones de los empleados	
Empleados	Funciones
M1	Director de proyecto y analista funcional
M2	Director de proyecto y analista funcional
M3	Director de proyecto y analista funcional
M4	Encargado de sistemas
M5	Diseñador
M6	Desarrollador SAP, desarrollador APP
M7	Desarrollador SAP, desarrollador APP
M8	Desarrollador SAP
M9	Desarrollador SAP
M10	Desarrollador CRM
M11	Desarrollador CRM

Tabla 4.1. Listado de empleados y funciones.

La labor de cada uno de ellos es la siguiente:

- Director de proyecto y analista funcional. Funcionan como el nexo entre las necesidades del cliente y el equipo informático encargado de elaborar la solución solicitada, analizando los procesos involucrados en el negocio para descubrir posibles necesidades.
- Encargado de sistemas. Se ocupa de arreglar los problemas informáticos que puedan existir, así como de elaborar tareas de mantenimiento (copias de seguridad, software actualizado, ordenadores libres de malware, etcétera).

- Diseñador. Las tareas del diseñador son muy variadas e incluyen la creación de la imagen de marca e identidades corporativas, impresión de logotipos y la creación de material audiovisual como vídeos y animaciones empresariales que se emplean como material publicitario y contenido en redes sociales.
- Desarrolladores. Realizan las actividades relacionadas con la implementación de software. Generalmente, el software de la solución propuesta cuenta con piezas o módulos diferenciados y en muchos casos la base requiere de las mismas piezas. La labor de los desarrolladores es preparar estas piezas para que puedan integrarse y desarrollar un software adaptado a las características y necesidades del cliente. Como podemos ver en la tabla anterior, los desarrolladores se especializan en la implementación según el tipo de proyecto. De este modo, la empresa cuenta con 2 desarrolladores de CRM y 4 desarrolladores de ERP, de los cuales 2 son también desarrolladores de Apps.

En cuanto a los recursos materiales, la empresa dispone de un amplio número de ordenadores de altas prestaciones junto con paquetes informáticos y software que les permiten a los empleados desarrollar perfectamente su labor.

Los objetivos consisten en llevar a cabo los proyectos de desarrollo tecnológico que se aplicarán en las empresas que lo soliciten. Por todo ello, se necesita planificar adecuadamente la tarea a realizar por cada uno de los empleados de forma que se minimice los tiempos requeridos para finalizar los proyectos considerados.

En este trabajo vamos a analizar la planificación del proyecto titulado: Desarrollo de una solución CRM. No obstante, los proyectos de desarrollo de las diferentes soluciones tienen características similares y los resultados obtenidos son fácilmente trasladables a cualquiera de ellos.

Proyecto de desarrollo de una solución CRM

Un CRM es un paquete informático de gestión de las relaciones con clientes, orientada normalmente a gestionar tres áreas básicas: la gestión comercial, el marketing y el servicio postventa o de atención al cliente. Permite centralizar en una única base de datos todas las interacciones entre una empresa y sus clientes. Por ello, un software CRM cuenta con tres grandes módulos interconectados:

- Base de datos de clientes. Una solución CRM permite compartir y maximizar el conocimiento de un cliente dado y de esta forma entender sus necesidades y anticiparse a ellas.
- Base de datos de oportunidades. Las empresas que utilizan soluciones CRM generan más oportunidades de venta, agilizando la gestión con presupuestos actualizados en tiempo real y procesos de ventas optimizados.
- Base de datos de campañas. Un CRM permite dirigir y gestionar de forma más sencilla las campañas de captación de clientes y de fidelización, contro-

lando el conjunto de acciones realizadas sobre los clientes y gestionando las acciones comerciales a partir de un cuadro de mando detallado.

Estas bases de datos no son independientes, sino que están relacionadas. De este modo, cuando una campaña de captación consigue que un cliente potencial se muestre interesado, se abre una nueva oportunidad de venta. Si finalmente, esta oportunidad se cierra con éxito y se efectúa la compra se registrará la información en la base de datos de clientes. La información dentro de cada base de datos se debe adaptar a la empresa que utilice la solución CRM.

El proyecto de desarrollo de una solución CRM consiste en varias fases:

Fase 1: Preparación inicial. Esta fase supone una primera toma de contacto con el cliente. Se identifica, analiza y documenta los requisitos del producto CRM a desarrollar.

Fase 2: Análisis. En esta fase se estudia de manera detallada el desarrollo de la solución CRM. Para ello, es necesario acordar con el cliente plazos y presupuestos de acuerdo a los requisitos señalados en la fase anterior. Además, en esta fase se analiza la empresa en detalle, estudiando la información de la que dispone para registrar en el CRM y se asigna un equipo de trabajo interno al proyecto. Como ya hemos dicho, la información que registrará la solución, así como el análisis de la misma, depende de la empresa que lo solicite; por ejemplo, la estructura de un CRM de una administración pública no será igual que la de un distribuidor de fruta. Cuando se comienza un proyecto CRM para un tipo de empresa con la que no se ha trabajado anteriormente puede ocurrir que el equipo de trabajo necesite formarse para determinar la mejor manera de implementar el software. La fase se completa cuando todas las partes involucradas en el proyecto determinan y acuerdan el alcance del mismo, así como las responsabilidades y tareas que asume cada uno.

Fase 3: Implementación. Como ya hemos comentado, en la implementación de una solución CRM se disponen de módulos o piezas que deben integrarse. La labor a completar en esta fase será desarrollar posibles módulos que no estén diseñados anteriormente para probar, individual e integradamente, las componentes de la solución diseñada. Esto permitirá encontrar y arreglar errores, comprobando que se satisfacen los requisitos especificados.

Fase 4: Pruebas. En esta fase se instala el CRM en su ambiente operacional, que comenzará a usarse a modo de prueba. Esto permite que el cliente se familiarice con el software y detecte posibles deficiencias que deben ser solucionadas antes de su puesta en marcha. Además, todos los empleados de la empresa para la que se diseña el CRM, que estén en contacto con los clientes, deben superar un periodo de formación, en el que se indicará todo lo necesario para hacer un buen uso de la solución proporcionada.

Fase 5: Arranque. Finalmente, el cliente comienza a utilizar la solución en modo operativo. Se proporcionarán horas de asistencia remota que serán de utili-

dad en el caso de que surjan dudas o se produzca algún error en la ejecución del programa.

De este modo, las actividades en el desarrollo de una solución CRM se pueden descomponer como se muestra en la tabla 4.2.

Fase	Designación	Actividad
1	A	Preparación inicial
2	B	Acuerdo de plazo y presupuesto
2	C	Análisis de la empresa
2	D	Designación de equipo de trabajo interno
2	E	Formación de equipo de trabajo interno
2	F	Acuerdo del alcance del proyecto
3	G	Implementación de la solución
4	H	Evaluación y aprobación de los procedimientos
4	I	Creación de un manual
4	J	Formación de los trabajadores
5	K	Arranque

Tabla 4.2. Actividades en el desarrollo de un proyecto CRM.

Generalmente, no hay conflicto entre los recursos que se precisan para elaborar las actividades citadas. Sin embargo, en la actividad G de implementación de la solución, el gran número de trabajos que se pueden desarrollar paralelamente y los limitados recursos humanos (2 desarrolladores) hacen que la asignación se deba realizar con especial cuidado para minimizar el tiempo que requiere completar el proyecto. Por ello, en la siguiente sección comenzamos abordando esta cuestión como un problema de planificación de tareas.

4.2. Resolución del problema de planificación de tareas para implementar una solución CRM

El problema a resolver en este caso será la asignación de las tareas de implementación de una solución CRM a los dos empleados (desarrolladores CRM) de los que dispone la empresa. Consideraremos que las diferencias en los tiempos de procesamiento entre los empleados no son significativas, por lo que abordamos un problema de máquinas paralelas idénticas. Las tareas de desarrollo no requieren de comunicación con el cliente, sino que se elaboran escribiendo un código en el lenguaje de programación correspondiente. Este código puede ser guardado y continuado sin penalizaciones, por lo que consideramos que las tareas son interrumpibles.

Las tareas a considerar son las siguientes:

Implementación de un CRM	
Tarea	Descripción
G1	Desarrollo del módulo de clientes
G2	Desarrollo del módulo de campañas
G3	Desarrollo del módulo de oportunidades
G4	Pruebas de integración de los módulos
G5	Migración de datos
G6	Verificación interna de la solución
G7	Validación

Tabla 4.3. Tabla de tareas de desarrollo de un CRM.

Se tienen relaciones de precedencias entre las tareas y estas siguen forman un grafo tipo árbol (*tree*). La siguiente imagen muestra las relaciones de precendencia, donde los arcos representan las tareas, y los vértices indican el principio o fin de cada una de ellas.

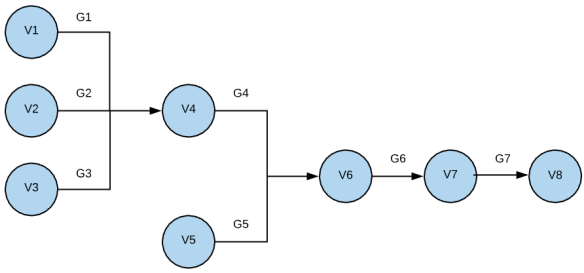


Figura 4.1. Relaciones de precedencia en las tareas de desarrollo de un CRM

Por tanto, el problema a resolver es un problema de tipo $P_2|pmtn,\ intree|C_{max}$, que puede resolverse empleando el algoritmo de Muntz y Coffman. En este caso, el tenemos un total de 7 tareas y 2 trabajadores, por lo que n=7 y m=2. En la siguiente tabla, el parámetro p_j denota el tiempo más probable de proceso del trabajo Gj (o simplificando j) en cualquiera de las máquinas y S_j el conjunto de sucesores directos del trabajo j.

Tarea j	1	2	3	4	5	6	7
Tiempo p_j	10	8	8	5	2	2	1
Sucesor S_j	4	4	4	6	6	7	-

Tabla 4.4. Tiempos de proceso (en horas) y relación de sucesores para los trabajos de desarrollo de CRM

Para aplicar el algoritmo de Muntz y Coffman al problema descrito comenzamos en la fase de etiquetado. En primer lugar, etiquetamos el trabajo o tarea terminal, j = 7, por tanto $\alpha_7 = p_7 = 1$.

Seguidamente hacemos la asignación de etiquetas de los demás trabajos:

$$\alpha_6 = max_{i \in S_6}\{\alpha_i\} + p_6 = 1 + 2 = 3\,; \quad \alpha_5 = max_{i \in S_5}\{\alpha_i\} + p_5 = 3 + 2 = 5$$
$$\alpha_4 = max_{i \in S_4}\{\alpha_i\} + p_4 = 3 + 5 = 8\,; \quad \alpha_3 = max_{i \in S_3}\{\alpha_i\} + p_3 = 8 + 8 = 16$$
$$\alpha_2 = max_{i \in S_2}\{\alpha_i\} + p_2 = 8 + 8 = 16\,; \; \alpha_1 = max_{i \in S_1}\{\alpha_i\} + p_1 = 8 + 10 = 18$$

Comenzamos con los trabajos de mayor etiqueta y que no tienen un predecesor que dependa de él, en este caso, la etiqueta máxima será la del trabajo 1, pues $\alpha_1 = 18$. Además, vemos que se produce un empate entre las etiquetas de los trabajos 2 y 3, pues $\alpha_2 = \alpha_3 = 16$. En la siguiente tabla se muestran los trabajos planificables. Nótese que α_j representará la etiqueta de trabajo j, p_j el tiempo de procesamiento restante que requiere el trabajo j y finalmente θ_j indicará el ratio de procesamiento asignado.

j planificable	1	2	3
α_j	18	16	16
p_j restante	10	8	8
θ_j	1	$\frac{1}{2}$	$\frac{1}{2}$

Como hay dos trabajos con igual etiqueta se comparte un procesador con ratio de proceso $\frac{1}{2}$. Al trabajo J_1 de etiqueta máxima se le asigna una única máquina con ratio de proceso 1. Se mantiene esta asignación hasta que ocurre alguno de los supuestos del paso 4. Veamos en qué instante ocurre alguno de los dos supuestos:

a) Se acaba algún trabajo.
Sabemos que el tiempo de procesamiento que necesitan los trabajos para ser completado $p_1^{'}$, $p_2^{'}$ y $p_3^{'}$ son funciones que varían con el tiempo. Estas funciones son $p_1^{'}(t) = 10 - t$ y $p_2^{'}(t) = 8 - \frac{1}{2}t$. Diremos que un trabajo j se ha completado cuando $p_j^{'} = 0$. Por lo tanto, en este caso, se tiene que:

$$p_1^{'}(t) = 10 - t = 0 \Rightarrow t = 10$$
$$p_2^{'}(t) = p_3^{'}(t) = 8 - \tfrac{1}{2}t = 0 \Rightarrow t = 16$$

b) Aparece un trabajo con mayor etiqueta y menor ratio de proceso.
Las etiquetas α_1, α_2 y α_3 son funciones que dependen del tiempo. Estas funciones son $\alpha_1 = 18 - t$, y $\alpha_2 = \alpha_3 = 16 - \frac{1}{2}t$. Podemos comprobar que b) se satisface cuando $\alpha_1 \leq \alpha_2 = \alpha_3$:

$$\alpha_1 \leq \alpha_2 \Rightarrow 18 - t \leq 16 - \tfrac{1}{2}t \Rightarrow t \geq 4$$

Como el menor de estos instantes es t = 4, la decisión tomada en el instante t = 0 se mantiene hasta el instante t = 4. En ese momento hay que tomar una nueva decisión.

Por tanto, en este instante t=4, debemos actualizar las etiquetas y los tiempos de procesamiento remanentes. Como en el caso anterior, los resultados se muestran en la siguiente tabla:

j planificable	1	2	3
α_j	14	14	14
p_j restante	6	6	6
θ_j	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$

Como hay tres trabajos con igual etiqueta se comparten los dos procesadores con ratio de proceso $\frac{2}{3}$. Se mantiene esta asignación hasta que ocurre alguno de los supuestos del paso 4. Vamos a estudiar ahora el menor instante en el que ocurre alguno de los sucesos:

a) Se acaba algún trabajo. En este caso, se tiene que $P_1^{'}(t) = P_2^{'}(t) = P_3^{'}(t) = 6 - \frac{2}{3}t$. Por tanto, los trabajos se completarán cuando:

$$p_1^{'}(t) = p_2^{'}(t) = p_3^{'}(t) = 6 - \tfrac{2}{3}t = 0 \Rightarrow t = 9$$

b) Aparece un trabajo con mayor etiqueta y menor ratio de proceso.
Esto ocurrirá en el instante en el que el trabajo j=5 tenga una etiqueta mayor que la de los trabajos j=1, j=2 y j=3. Dado que el trabajo j=5 no está siendo procesado, se tiene que $\alpha_5 = 5$. Y, por tanto, debemos resolver:

$$\alpha_1 = \alpha_2 = \alpha 3 = 14 - \tfrac{2}{3}t \leq 5 \Rightarrow t \geq 13'5$$

Como el menor de estos instantes es t = 9, la decisión tomada en el instante t = 4 se mantiene hasta el instante t = 13, en el que los trabajos 1, 2 y 3 finalizan. En ese momento hay que tomar una nueva decisión.

Así, en este instante t=13, los trabajos 1, 2 y 3 se han completado. Ahora, los trabajos planificables serán j=4 y j=5 y tendremos la siguiente tabla:

j planificable	4	5
α_j	8	5
p_j restante	5	2
θj	1	1

En este caso, se asigna un único trabajo a cada máquina. Se mantiene esta asignación hasta que ocurre alguno de los siguientes supuestos:

a) Se acaba algún trabajo. En este caso, se tiene que $p_4^{'}(t) = 5-t$ y $p_5^{'}(t) = 2-t$. Por tanto, los trabajos se completarán cuando:

$$p_4^{'}(t) = 5 - t = 0 \Rightarrow t = 5$$
$$p_5^{'}(t) = 2 - t = 0 \Rightarrow t = 2$$

b) Aparece un trabajo con mayor etiqueta y menor ratio de proceso. Este supuesto no ocurre ya que los siguientes trabajos a realizar necesitan de la finalización de sus predecesores. Por tanto, hasta que no termine el trabajo 4 y 5 no puede comenzar el trabajo 6.

Como el menor de estos instantes es t = 2, la decisión tomada en el instante t = 13 se mantiene hasta el instante t = 15, en el que el trabajo 5 finaliza. En ese momento hay que tomar una nueva decisión.

De acuerdo con las relaciones de precedencia, sabemos que, en adelante, existirá un único trabajo planificable en cada instante. En t=15 el trabajo 4 requiere de 3 horas para ser completado, y el trabajo 6 no podrá comenzar hasta ese momento. Así, desde el instante t=15 hasta t=18 se asignará una máquina al trabajo 4 con ratio de procesamiento $\theta = 1$.

En t=18, el único trabajo procesable será el trabajo 6, pues el trabajo 7 no puede comenzar hasta que éste finalice. Por tanto, desde el instante t=18 hasta el instante t=20, se asignará una única máquina al trabajo 6 con ratio de procesamiento $\theta = 1$. Y finalmente, en el instante t=20 el trabajo 6 habrá finalizado y se comenzará a procesar el 7, que se completará en t=21.

De este modo, la planificación óptima dada por el algoritmo se muestra en la figura 4.2.

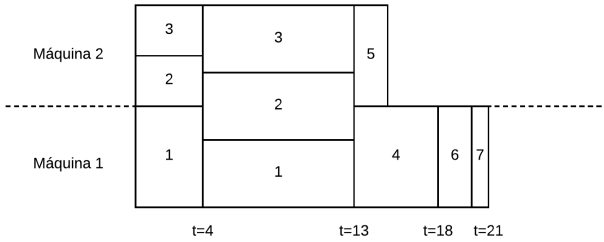


Figura 4.2. Aplicación del algoritmo de Muntz y Coffman usando procesadores compartidos.

Esta solución se puede transformar, de manera que no se utilice procesador compartido, obteniendo así la figura 4.3:

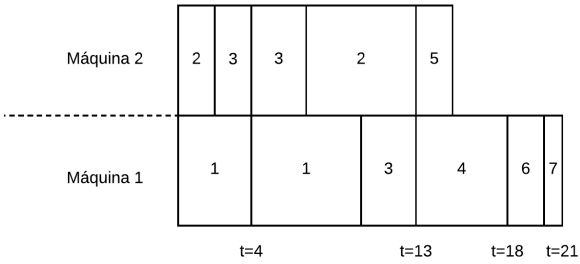


Figura 4.3. Aplicación del algoritmo de Muntz y Coffman sin usar procesador compartido.

La solución propuesta por el algoritmo de Muntz y Coffman tiene a lo sumo $O(nm) = O(14)$ interrupciones. En este caso, se producen solamente 2 interrupciones, la primera se produce en el instante t=2 y afecta al trabajo 2, y la segunda interrumpe el trabajo 3 en el instante t=7.

4.3. Resolución del problema de planificación del proyecto de desarrollo de una solución CRM

Una vez que hemos resuelto el problema de planificación de tareas para la actividad de implementación de un software CRM podemos estudiar, de manera globalizada, la programación de este proyecto. Las actividades en las que se descompone la realización de un proyecto de desarrollo de CRM se recogen en la tabla 4.2. Sus correspondientes tiempos de ejecución figuran reflejados en el siguiente cuadro.

Duración (en horas)	A	B	C	D	E	F	G	H	I	J	K
Tiempo optimista	2	1	2	1	1	2	19	1	3	3	4
Tiempo más probable	3	2	4	1	2	4	21	2	6	4	4
Tiempo pesimista	4	3	6	1	9	5	29	3	9	11	10
Tiempo PERT	3	2	4	1	3	4	22	2	6	5	5

Tabla 4.5. Actividades en el desarrollo de un proyecto CRM y tiempos de ejecución (en horas).

Las relaciones de precedencia entre las diferentes actividades que constituyen el proyecto son:

Actividad	A	B	C	D	E	F	G	H	I	J	K
Precedentes	-	A	A	B	D	C, E	F	G	G	I	H, J

Tabla 4.6. Relaciones de precedencia de las actividades en el desarrollo de un proyecto CRM.

A partir del cuadro de prelaciones podemos construir el correspondiente grafo PERT, que estará representado en la figura 4.4. Nótese que cada actividad lleva asignada su tiempo PERT.

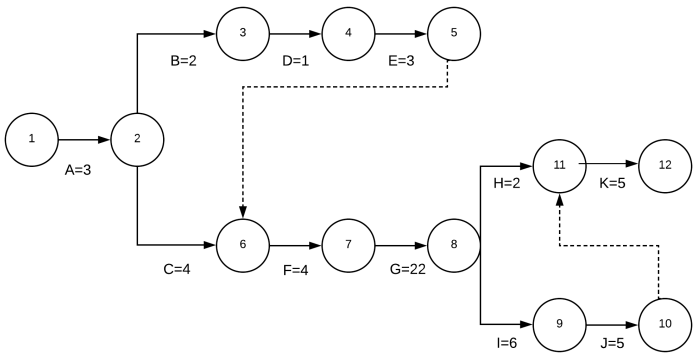


Figura 4.4. Diagrama PERT de un proyecto CRM

Ahora, podemos calcular los tiempos *early* y *last* de los diferentes sucesos. Recordemos que comenzábamos asignando al suceso de inicio del proyecto un tiempo early de 0. De este modo, el tiempo *early* de un suceso j, que se repre-

sentaba como t_j, viene dado por $t_j = max[t_i + t_{ij}]$, $\forall i$, donde t_{ij} es la duración de la actividad que comienza en el vértice i y finaliza en el vértice j.

Por otro lado, el proceso para deteminar los tiempos *last* se iniciaba asignando al suceso fin un tiempo *last* que fuera igual al tiempo *early* previamente calculado. Entonces, el tiempo *last* de un cierto suceso i, que se representa como t_i^* será igual a $t_i^* = min[t_j^* - t_{ij}]$, $\forall j$.

De este modo, los resultados obtenidos se muestran en la siguiente tabla.

Actividad	Designación	Duración (horas)	t_i	t_j	t_i^*	t_j^*
1-2	A	3	0	3	0	3
2-3	B	2	3	5	3	5
2-6	C	4	3	9	3	9
3-4	D	1	5	6	5	6
4-5	E	3	6	9	6	9
6-7	F	4	9	13	9	13
7-8	G	22	13	35	13	35
8-11	H	2	35	46	35	46
8-9	I	6	35	41	35	41
9-10	J	5	41	46	41	46
11-12	K	5	46	51	46	51

Tabla 4.7. Tiempos early y last asociados a los sucesos inicio y final de las actividades en el desarrollo de un proyecto CRM.

Conocidos los tiempos *early* y *last* de cada suceso, calculamos, por aplicación de las correspondientes fórmulas, las holguras. Recordemos que la holgura de un cierto suceso i, que representamos por H_i se define como la diferencia entre los tiempos *early* y *last* de dicho suceso, es decir, $H_i = t_i^* - t_i$. Por otro lado, la holgura total de una actividad (i,j), que representamos como H_{ij}^T se define como el tiempo que resulta de restar al tiempo *last* del suceso final el tiempo *early* del suceso inicial y la duración de la actividad, es decir, $H_{ij}^T = t_j^* - ti - t_{ij}$. De esta manera, una vez calculadas dichas holguras, las mismas se recogen en la tabla 4.8.

De la información contenida en la tabla 4.8 se calcula el diagrama-calendario que está representado en la figura 4.5. Este diagrama-calendario representa la fecha de comienzo y finalización más temprana de las actividades. Está sacado con el software libre ProjectLibre. Las barras representadas en rojo corresponden al camino crítico y las barras azules son las actividades no críticas. A su vez, las barras que aparecen rayadas será la holgura de las actividades no críticas. El calendario está diseñando cuando cada día se realizan jornadas de 7 horas. Las filas representarán las tareas a desarrollar ordenadas alfabéticamente y las columnas representarán los días laborales dedicados al proyecto.

Actividad	Designación	Duración	t_i	t_j	t_i^*	t_j^*	H_i	H_j	H_{ij}^T	Crítica
1-2	A	3	0	3	0	0	0	0	0	CC
2-3	B	2	3	5	3	5	0	0	0	CC
2-6	C	4	3	9	3	9	0	0	2	-
3-4	D	1	5	6	5	6	0	0	0	CC
4-5	E	3	6	9	6	9	0	0	0	CC
6-7	F	4	9	13	9	13	0	0	0	CC
7-8	G	22	13	35	13	35	0	0	0	CC
8-11	H	2	35	46	35	46	0	0	9	-
8-9	I	6	35	41	35	41	0	0	0	CC
9-10	J	5	41	46	41	46	0	0	0	CC
11-12	K	5	46	51	46	51	0	0	0	CC

Tabla 4.8. Holguras relacionadas con las actividades en el desarrollo de un proyecto CRM.

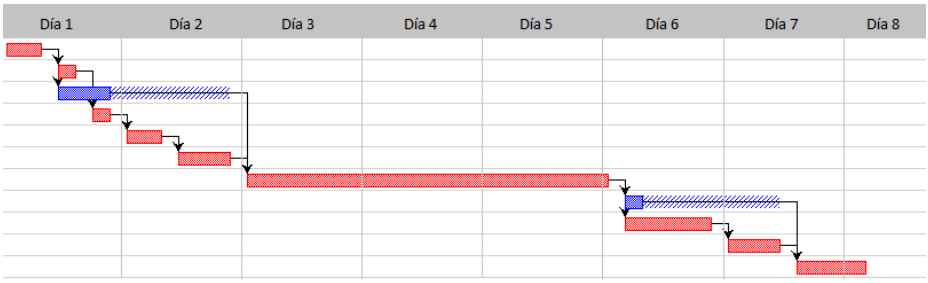


Figura 4.5. Calendario de un proyecto CRM

Concluimos entonces que un proyecto de CRM puede completarse en 51 horas laborales. Esto requiere de siete jornadas y dos horas laborales. La planificación presentada permite a la empresa ahorrar tiempo en el desarrollo de un proyecto CRM, sobre el que antes no se tenía ningún control. Hemos detectado como podemos asignar los recursos humanos de manera más eficiente a las actividades críticas para finalizar el desarrollo de un software CRM en el plazo de tiempo mínimo posible.

4.4. Conclusiones y trabajos futuros

En este Trabajo de Fin de Grado se han comentado los conceptos fundamentales de la metodología usada en la programación y planificación de proyectos, así como la secuenciación y asignación de tareas.

Posteriormente, hemos descrito el conjunto de actividades a ejecutar en el desarrollo de un proyecto tecnológico que se lleva a cabo en una empresa localizada en la isla de Tenerife. Una vez recogidos los recursos humanos disponibles para la planificación de proyectos y la secuenciación de tareas para obtener una solución al problema de programación de actividades en el desarrollo del proyecto tecnológico.

Más concretamente, hemos aplicado el algoritmo de Muntz y Coffman para obtener una buena planificación que permita asignar las diferentes tareas que se deben realizar por los dos desarrolladores CRM dentro de la actividad de implementación de la solución, con el objetivo de finalizar lo más pronto posible la ejecución de esas tareas.

Una vez determinado el tiempo requerido para la implementación de la solución tecnológica, se ha utilizado la metodología PERT que nos ofrece la planificación de proyectos para calcular la duración esperada de todas las actividades del proyecto, así como identificar las actividades críticas que deben controlarse para evitar la demora en la ejecución del mismo.

Como trabajos futuros, se podría plantear la incorporación en la programación de actividades de los distintos tipos de soluciones (ERP, Apps, etcétera.) La planificación de cada uno de ellos de manera individual puede considerarse una variante del problema resuelto en este trabajo. Sin embargo, cuando consideremos estos proyectos de manera conjunta tendremos que analizar las actividades críticas, la compatibilidad de los recursos, la duración del proyecto y el coste mínimo.

Bibliografía

[1] Alcaide, D. (1995) *Problemas de planificación y secuenciación determinística: modelización y técnicas de resolución. Serie Tesis Doctorales. Universidad de La Laguna.*

[2] Baker, K.R. y Trietsch, D. (2009) *Principles of Sequencing and Scheduling. John Wiley and Sons, Inc.*

[3] Brucker, P. (2007) *Scheduling Algorithms. Springer-Verlag.*

[4] González, T. y Johnson, D.S. (1980) *A New Algorithm for Preemptive Scheduling of Trees. J.Assoc.Comput. 27, 287-312.*

[5] González, T. y Sahni, S. (1978) *Preemptive Scheduling of Uniform Processor Systems. J.Assoc.Comput. 25, 92-101.*

[6] Graham, R.L; Lawler, E.L; Lenstra, J.K ; Rinnoy Kan, A.H.G (1979) *Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of discrete mathematic, vol 5, pp. 287-326.*

[7] Hu, T.C. (1961) *Parallel sequencing and assembly line problems. Oper. Res. 9, 841-848.*

[8] Jackson, J.R. (1955) *Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, UCLA.*

[9] Johnson, S.M. (1954) *Optimal two and three stage production schedules with setup includes. Naval Research Logistic Quartely, 1, 61-68.*

[10] Lawler, E.L. (1976) *Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston.*

[11] McNaughton, R. (1959) *Scheduling with deadlines and loss functions. Management Sci. 6, 1-12.*

[12] Meza, O. y Ortega, M. (2007) *Grafos y algoritmos. Editorial Equinoccio.*

[13] Muntz, R.R. y Coffman, E.G. (1970) *Preemtive scheduling of real time task on multiprocessor systems. J. Assoc. Comput. 17, 324-338.*

[14] Pinedo, M.L (2009) *Scheduling: Theory, Algorithms, and Systems. Springer-Verlag.*

[15] Romero, C. (2002) *Técnicas de programación y control de proyectos. Ediciones Pirámide.*

[16] Shachnai, H., Tamir, T. y Woeginger, G. (2005) *Minimizing Makespan and Preemption Costs on a System of Uniform Machines. Algorithmica. 42, 309-334.*

[17] Velasco, J y Campins, J.A. (2013) *Gestión de proyectos en la empresa: Planificación, programación y control. Ediciones Pirámide.*

Lista de Figuras

Application of job scheduling for planning technological development projects

Anabel Estévez Carrillo

Facultad de Ciencias · Sección de Matemáticas

Universidad de La Laguna

alu0100826975@ull.edu.es

Abstract

In this report, we provide methods to schedule the activities of a technological development company with the aim of finishing as soon as possible the execution of the activities that workers must carry out. To solve this problem, it is necessary to study the fundamental concepts of project management and tasks scheduling. As a result, solutions are proposed that could be applied to the business in question. The ideas and developments made in this report could also be extended and applied to other companies that might be interested in the distribution of tasks among their workers.

1. Introduction

In Operational Research, various disciplines have been developed aimed at optimizing the allocation of resources to achieve maximum efficiency. The Task Scheduling and Sequencing is a field of investigation within mathematics focused on determining the optimal way in which we can allocate resources to tasks or activities in specific time periods taking into account the objective to be optimized. Many times, in practice, we have to develop a project in the medium or long term which requires the conjunction of a large number of tasks, maintaining certain precedence relationships between them. In this case, Project Scheduling provides us with a methodology and a set of tools that help us to determine the time schedule of tasks to finalize the project as soon as possible. It also helps us to identify the fundamental activities that we must control to avoid any delay of the same, since this would lead to a delay in the completion of the project.

2. First chapter. Fundamentals

This chapter includes an introduction to the scheduling problems related to the execution of a project with a number of activities to be processed in a preset order. Specifically, we describe the methodology followed by PERT. This project management technique allows us to have a global vision of the project, while detailing the entire process and the resources involved. It is considered a probabilistic method because it considers that the execution time of an activity is an unknown variable for which only estimates are available. The following steps are involved in PERT technique: the activities involved in the project are drawn up in a graph that shows precedence relationships, the time required for completing each activity of the project is estimated, the critical activities of the project are determined, the duration of the project is calculated and an execution schedule is established.

3. Second chapter. General Structure

In the second chapter, we present the formulation and representation of task scheduling problems. The most widespread classification is the one proposed by Graham, in which the different scheduling problems can be cataloged according to three fields: α includes the characteristics that describe the m machines or processors; in the parameter β those of the n jobs or tasks to be processed; and finally, the parameter γ refers to the considered optimization criterion. Finally, we discuss some concepts of computational complexity that allow us to evaluate the efficiency of the methods of solving task planning problems.

4. Third chapter. Parallel Machines

In this chapter we study different methods to solve a task planning problem in which there are several machines in parallel to process jobs. The objective considered is to minimize the makespan, that is, the minimum time in which all the works have been completed. This ensures a good balance between the loads of the machines in parallel. Scheduling problems in which jobs can not be interrupted are usually difficult, while allowing interferences simplifies the analysis of the problem. For that reason, on the one hand, we study the problems without interruptions (LPT heuristic and HU's algorithm) and, on the other, the problems with interruptions (McNauthon and Muntz and Coffman algorithm).

5. Fourth chapter. Case Study

In this chapter, based on the theory presented throughout this memory, we study the project scheduling of a technological development company located in Santa Cruz de Tenerife. We have described the set of activities to be processed in the development of a CRM technology project. Subsequently, we have applied the project planning and tasks sequencing to obtain a solution to the problem of programming activities in the development of the technological project. More specifically, we have applied the Muntz and Coffman algorithm to assign the different tasks in the implementation of the solution. Once the time required for the implementation of the solution has been determined, PERT has been used to calculate the most probable duration of the project.

References

[1] ALCAIDE, D. (1995) *Problemas de planificación y secuenciación determinística: modelización y técnicas de resolución. Serie Tesis Doctorales. Universidad de La Laguna.*

[2] BAKER, K.R. Y TRIETSCH, D. (2009) *Principles of Sequencing and Scheduling. John Wiley and Sons, Inc.*

[3] BRUCKER, P. (2007) *Scheduling Algorithms. Springer-Verlag.*

[4] PINEDO, M.L (2009) *Scheduling: Theory, Algorithms, and Systems. Springer-Verlag.*

[5] ROMERO, C. (2002) *Técnicas de programación y control de proyectos. Ediciones Pirámide.*