UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES DE MADRID

MODELO DE PROGRAMACION PARA LA OPMIZACION DEL SECUENCIADO EN EL AMBITO DE LA INDUSTRIA AERONAUTICA

Jorge Guede Barcenilla

Tutores: Álvaro García Sánchez y Raúl Pulido Martínez

GRADO EN INGENIERÍA DE TECNOLOGÍAS INDUSTRIALES

FEBRERO 2017

RESUMEN DEL PROYECTO

Los problemas relacionados con la asignación de recursos humanos y no humanos a diferentes tareas en un problema de secuenciado con recursos limitados aparecen cada vez con mayor asiduidad en la industria. Dentro de este contexto, los encargados técnicos encargados de esta planificación deben encontrar soluciones que reduzcan costes, horas de trabajo u otros criterios.

En numerosas ocasiones, este trabajo se hace a mano, sin embargo, este método no resulta efectivo pues puede conducir a errores y, para el caso de problemas de gran tamaño, como los que se tratan en este trabajo, resulta impracticable. Estos errores son fatales para las empresas en general y más si cabe en el ámbito de la industria aeronáutica, que es el que nos ocupa en este trabajo, pues los costes en materias primas, mano de obra, maquinaria, entre otros, son muy elevados.

En este proyecto enmarcado dentro de otro de mayor magnitud llevado a cabo por una compañía aeronáutica se desarrolla una herramienta basada en la programación de un heurístico con el que se busca encontrar una solución lo suficientemente buena con un tiempo de ejecución reducido. Esto contrasta con otros métodos como la programación lineal entera, que si bien permite obtener soluciones en principio más cercanas al óptimo, tiene como contrapartida un tiempo de ejecución muy elevado.

El objetivo de este heurístico y este trabajo de fin de grado es la obtención de una asignación detallada de los recursos humanos y no humanos de manera que el tiempo entre el comienzo y la finalización del trabajo sea mínimo una vez fijados los recursos.

Este tipo de problema de optimización es conocido como RCSP, “Resource Constraint Scheduling Project”, en la literatura, es decir, problema de secuenciado con restricciones en los recursos. Existe otro enfoque para los problemas de secuenciado, el TCSP “Time Constraint Scheduling Project”, en el que fijado el tiempo en el que el trabajo debe estar listo, se asignan los recursos necesarios. Este segundo enfoque resulta ser el más habitual, sin embargo, no tiene tanto sentido aplicarlo a la industria aeronáutica.

El motivo es que en esta industria no es tan importante tener el pedido lo más pronto posible, pues el tiempo para tenerlo listo suele ser amplio, si no minimizar los recursos empleados, en vistas a ahorrar costes. El enfoque empleado para este problema, al no ser el habitual y no haber sido muy estudiado, añade valor al trabajo desarrollado.

Apéndices

Indice de imágenes y gráficos

Indice de tablas

1. Introducción

1.1. Presentación

La producción es la elaboración de un producto mediante el trabajo. En vistas del crecimiento demográfico y, por tanto, de las cantidades demandadas por el mercado, la producción juega un papel vital en nuestra sociedad. Además, es muy importante para las empresas mejorar y producir cada vez a precios más bajos, debido a la aparición constante de nuevos competidores en el mercado. Por estos motivos, resulta tan importante optimizar la producción, es decir, producir la mayor cantidad posible de producto a bajo coste, mejorando así, el rendimiento.

En este trabajo de fin de grado se ha abarcado el desarrollo y la programación de un modelo de optimización mediante un heurístico que permita obtener secuencias de fabricación de piezas aeronáuticas que se ajusten a las necesidades de la empresa, en un tiempo reducido, con su respectivo diagrama de Gantt de salida correspondiente a la programación de las tareas a realizar de una serie de piezas. Cabe destacar que este proyecto forma parte de un conjunto mayor, junto a otros desarrollados en diversas áreas de la empresa, que involucran otros temas como la simulación o la programación de problemas estocásticos.

1.2. Justificación

En este contexto, una empresa aeronáutica, debido al elevado coste de la materia prima, la mano de obra y la maquinaria para la manufactura necesita de herramientas muy potentes para gestionar la producción. Este heurístico, resulta muy útil para valorar la acometida de nuevos proyectos dentro de la empresa, pues ofrece soluciones muy buenas en un tiempo reducido, lo que permite comprobar si el desarrollo de un nuevo producto va a ser viable o no.

Una vez comprobada la viabilidad del producto, se podría desarrollar una herramienta más potente que este heurístico para tratar de obtener soluciones más cercanas al óptimo, si bien, y como veremos a lo largo de la exposición de este trabajo, el heurístico obtiene soluciones lo suficientemente buenas, pues, por ejemplo, en la realidad no es posible que los trabajadores estén trabajando el 100% del tiempo que

comprende su jornada laboral, siendo necesario establecer descansos. El tiempo real que un trabajador puede estar operando podría considerarse como el 90% de su jornada laboral y este heurístico obtiene soluciones de hasta un 92% de la ocupación (revisar).

Para dar más soporte a la importancia del desarrollo de este heurístico, vamos a resolver de manera manual y a modo de ejemplo, un problema de secuenciado de ocho tareas, representado esquemáticamente en la Ilustración 1, y cuyas especificaciones están expresadas en la Tabla 1.

Tarea	Duración	MinOp	MaxOp	Estructural	Mecánicos	Area A	Area B
1	3	2	3	1	0	1	0
2	2	1	2	0	1	0	1
3	5	2	2	0	1	1	0
4	2	2	3	1	1	1	0
5	6	3	3	0	1	0	1
6	3	1	3	1	0	0	1
7	2	2	2	1	1	0	1
8	3	1	2	1	0	1	0

Tabla 1: se adjuntan la duración, operarios mínimos (MinOp), operarios máximos (MaxOp) necesarios para resolver cada tarea. Además, se añaden otros datos que no se emplearán para la resolución del ejemplo, significando "1" que se emplea ese recurso, y "0", que no. Son los tipos de operarios, "Estructural" y "Mecánicos", y las áreas de trabajo, "Area A" y "Area B".

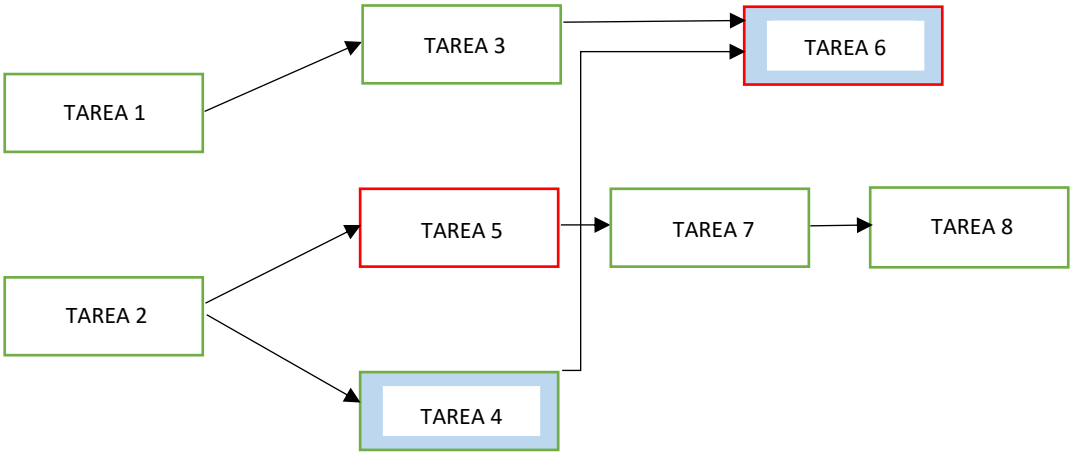

Ilustración 1: Esquema del "case 8.3". Los recuadros rellenos de azul corresponden a las tareas que han de ejecutarse con time-lag=0 (de manera consecutiva) y los recuadros con los bordes rojos corresponden a las tareas que no pueden ejecutarse a la vez.

El secuenciado de estas ocho tareas reviste una dificultad adicional, además de las restricciones habituales, existen otras dos adicionales, una que fuerza a una tarea a tener que ser programada inmediatamente después que otra, y otra que impide que dos tareas se ejecuten en paralelo.

El criterio para secuenciar una tarea antes que las demás, será establecido mediante una regla de prioridad, GRPW, es decir, "Greatest Rank Position Weight", que significa, mayor ranking según el peso. Se adjunta el esquema de la instancia ejemplo que, con los pesos correspondientes a cada tarea, en la Ilustración 2.

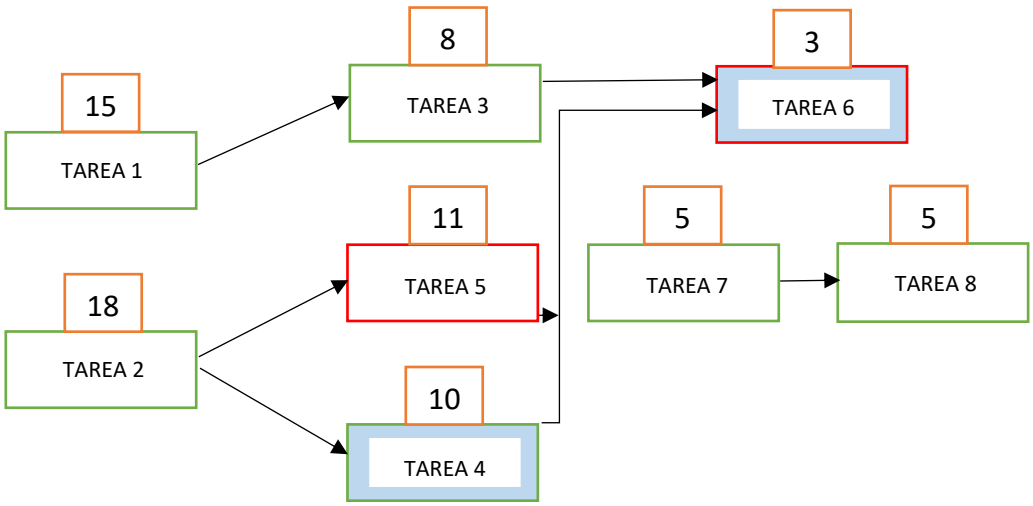

Ilustración 2: Esquema del "case 8.3". En esta imagen se pueden ver los valores de los pesos de cada tarea.

Esta regla consiste en asignar, a la tarea que esté en el conjunto de tareas que se pueden programar por haberse completado ya todos sus predecesores, un peso, siguiendo como criterio la suma de la duración de la tarea y la de todos sus sucesores. De esta manera, se asignaría la tarea con un mayor peso, es decir, con la mayor de la suma de su duración y la de sus predecesores.

Esta es la regla empleada para este trabajo, y el motivo de su elección será justificado en el capítulo cuatro.

Cada tarea, consume durante su duración operarios de un tipo, siendo posible en algunas tareas emplear operarios de los dos tipos. Cada uno de los tipos de operarios es un recurso que está limitado. Además, cada tarea sólo puede resolver en una de las dos áreas disponibles según sean las especificaciones de esa tarea. Cada área tiene una capacidad máxima. Esto añadiría una complejidad aún mayor si cabe para la resolución manual. A modo de simplificación y para este ejemplo, no se tendrán en cuenta esas restricciones de recursos.

Siguiendo esta regla de prioridad, y teniendo en cuenta las restricciones y los recursos necesarios para resolver cada tarea, y las simplificaciones asumidas, se procede a su resolución manual por etapas:

1. Se programa la tarea 2, pues es la de mayor peso, de entre las tareas 1 y 2, que son las que se pueden programar en esta etapa, entre los instantes 0 y 2.

2. Se programa la tarea 1, entre los instantes 0 y 3.

3. Se programa la tarea 5, entre los instantes 2 y 4.

4. Se programa la tarea 4, entre los instantes 2 y 4, e inmediatamente después, se debería programar la tarea 6 entre los instantes 4 y 7, pues viene exigido por la restricción. Sin embargo, esto no es posible ya que la tarea 3, predecesora de la 6, aún no ha sido programada. Esta es la primera dificultad que surge al resolverlo a mano. Por tanto, no se puede programar la tarea 4 todavía.

5. Se programa la tarea 3, entre los instantes 3 y 8. La tarea 6, no podrá comenzar hasta el instante 8, lo que añade otra complicación.

6. Se programa la tarea 4 de manera que esta tiene que finalizar en el instante 8, por tanto, se programa entre los instantes 6 y 8, pese a que habría sido factible programarla antes de no existir la restricción entre las tareas 4 y 6.

7. Se programa la tarea 6 entre los instantes 8 y 11

8. Se programa la tarea 7, esta no puede coincidir en su intervalo temporal en la tarea 6, pues como ya se comentó tienen una restricción que lo impide. La tarea 7 se secuencia entre los instantes 11 y 13.

9. Por último, la tarea 8 se secuencia entre los instantes 11 y 14 obteniéndose entonces un tiempo de duración del proyecto de 14 unidades de tiempo.

Como hemos visto, la resolución manual resulta engorrosa y complicada, pese a las simplificaciones realizadas. En la Tabla 2 se pueden ver los resultados de dicha resolución.

Tarea	Duración	Peso	Iteración	Inicio	Final
2	2	18	1	0	2
1	3	15	2	0	3
5	2	11	3	2	4
3	5	8	4	3	8
4	2	10	5	6	8
6	3	3	6	8	11
7	2	5	7	11	13
8	3	3	8	11	14

Tabla 2: resultados tras la resolución manual. Se incluyen, duración, peso, número de iteración, inicio y final de tarea

Además, esta resolución es impracticable para casos con más tareas y más restricciones. En este proyecto se resuelve el secuenciado de proyectos de hasta doscientas cuatro tareas mediante el heurístico, lo que sería imposible de manera manual.

Como añadido, esta resolución manual es determinista, lo que puede conducir a la mejor solución, o, lo que es mucho más probable, a una solución peor. Esto es debido a que la elección para secuenciar prioritariamente la tarea con mayor peso, si es posible, no tiene por qué proporcionar la mejor solución. A medida que crece el tamaño del proyecto este hecho adquiere más relevancia.

Una de las bondades del heurístico es que no es determinista, y permite explorar un amplio conjunto de soluciones, quedándose con la mejor dentro de ese conjunto. Como veremos más adelante, se empleará la aleatorización sesgada, de cara a obtener esa variabilidad en las soluciones.

Una vez expuestas las bondades de la herramienta a desarrollar, se exponen con detenimiento los objetivos de este proyecto

1.3. Objetivos

El objetivo general de este trabajo es la implementación de una herramienta de apoyo basada en un modelo de optimización heurístico que permita la obtención de la planificación de la producción de una línea de fabricación de componentes aeronáuticos de modo que permita atender la demanda establecida. Esto se logra a través de la asignación de operarios a las distintas tareas correspondientes a la planificación de la producción requerida por el cliente, buscando la minimización del tiempo de duración del proyecto fijados unos recursos. Para lograr dicho objetivo se definieron los siguientes objetivos específicos que permitieron, a través de ellos, la consecución del proyecto completo:

1. Desarrollar un heurístico que obtenga las mejores soluciones posibles en un período corto de ejecución.

2. Desarrollo de la salida de manera que se obtiene un Gantt que contiene el plan de producción.

3. Evaluar el impacto del uso del modelo, analizando soluciones para distinto número de trabajadores, empleando diferentes modos de manera aleatoria, de manera determinista o siguiendo otros criterios.

4. Analizar la capacidad del modelo, esto es, incrementando el número de iteraciones o variando el criterio de parada, observar a partir de qué punto la herramienta computa en tiempos inaceptables.

1.4. Metodología

El proyecto se ha desarrollado en diez etapas claramente diferenciadas. Esta división está basada en la cantidad de información manejada y las posibilidades de hacer comprobaciones de manera conjunta con la empresa aeronáutica.

En el desarrollo del proyecto se pudieron diferenciar:

1. Lectura de literatura relacionada con los métodos heurísticos.

2. Repaso del lenguaje de programación C y aprendizaje de nuevas herramientas del programa necesarias para el desarrollo del proyecto.

3. Recogida de información acerca de los requerimientos, modo de trabajo, procesos y características del trabajo realizado en la planta.

4. Realización de un primer heurístico para trabajos de pocas tareas a modo de práctica previa al desarrollo del programa de este proyecto.

5. Interpretación y elaboración de los requerimientos y criterios a partir de la información obtenida para el desarrollo del heurístico definitivo.

6. Investigación en profundidad de la literatura disponible para obtener la mejor forma de realizar el heurístico

7. Elaboración del modelo con el software C.

8. Verificación del modelo en funcionamiento, asegurando la correcta representación de la realidad y la factibilidad de las soluciones obtenidas.

9. Elaboración de la salida de datos en C para su uso en R de manera que se obtiene el plan de producción en un Gantt lo que lo hace muy visual y comprensible para los trabajadores de la fábrica.

10. Comparación de las soluciones obtenidas con el heurístico con las soluciones manuales empleadas por la empresa aeronáutica.

En todas las etapas del proyecto, se realizaron consultas con uno de los encargados del proyecto mayor de la compañía aeronáutica para conocer si el proyecto se estaba elaborando de la forma deseada y para aplicar posibles modificaciones o añadir nuevos aspectos

1.5. Software empleado

1.5.1. Lenguaje de programación C

Se emplea para este apartado información extraída de la página web de la Universidad de Granada (www.decsai.ugr.es, 2016).

Se ha decidido emplear, para la realización del programa, el lenguaje de programación C. Es un lenguaje de medio nivel. Los lenguajes de alto nivel se asemejan a nuestra forma de razonar, aislando al programador de los detalles técnicos (referentes a la máquina física). Esto hace que este tipo de lenguajes sean poco eficientes. Por el contrario, los lenguajes de bajo nivel controlan directamente la circuitería del ordenador, pudiendo obtenerse con ellos la eficiencia máxima (sin embargo, resultan incómodos y poco portables). Así las ventajas de los lenguajes de alto nivel respecto a los de bajo nivel son:

1. Sencillez.

2. Uniformidad.

3. Portabilidad.

En un nivel intermedio se sitúa el lenguaje C, permitiendo beneficiarse de las ventajas de ambos tipos de lenguajes, y reduciendo sus inconvenientes.

Es un lenguaje de propósito general. Se ha utilizado para el desarrollo de muy diversas aplicaciones: sistemas operativos, hojas de cálculo, gestores de bases de datos...

Es un lenguaje portable, es decir, es independiente del hardware. Los programas escritos en C son fácilmente trasportables a otros sistemas.

Ventajas:

1. Es un lenguaje potente y eficiente, permitiendo obtener programas rápidos y compactos.

2. Proporciona un completo control de cuanto sucede en el interior del ordenador.

3. Permite una amplia libertad de organización del trabajo.

Inconvenientes:

1. Es más complicado de aprender que otros lenguajes de programación como por ejemplo el Pascal.

2. Requiere una cierta experiencia para poder sacarle el máximo rendimiento.

3. Sin disciplina es difícil mantener el control del programa.

Su largo recorrido, amplía comunidad de usuarios, potencia, posibilidades y fiabilidad, son los motivos por los que se ha decidido emplear C, frente a otras opciones como Python, teniendo en cuenta también los problemas de disciplina. Por ello se tendrá mucho cuidado en la programación, empleando indentado adecuado y usando los comentarios de manera muy frecuente.

1.5.2. Lenguaje de programación R

Se emplea para este apartado información extraída de dos sitios web (http://rstadistica.blogspot.com.es/2015/10/VentajasDesventajasR.html, 2015) y (https://www.genbetadev.com/formacion/r-un-lenguaje-y-entorno-de-programacion-para-analisis-estadistico, 2014)

R es un lenguaje y entorno de programación para análisis estadístico y gráfico

Ventajas:

1. R es un software libre. Muchos de los softwares comerciales estadísticos cuestan cientos de dólares. Como SigmPlot cuesta cerca de 900 dólares, Minitab más de 1500 dólares, MatLab 2150 dólares, entre otros.

2. Es multiplataforma, R funciona en Mac, Windows, y en números sistemas UNIX. Esto significa que cualquier persona puede trabajar con tus datos, figuras, análisis y más importante aún usar tus instrucciones (también conocido como scripts o código) para generar las figuras y el análisis. Así que cualquier persona, y en cualquier lugar del mundo, con acceso a cualquier SO, es decir, Sistema Operativo, puede usar R sin ninguna licencia.

3. Es de código abierto, existe una gran comunidad de voluntarios trabajando para mejorarlo, lo cual permite ser moldeado y dirigido a cuestiones específicas. Se crean así programas y paquetes que funcionen en el entorno R. Programas tales como R-studio, Java GUI for R, R- commander, RKWard, entre otros, y con más de 6000 paquetes indexado en CRAN, Biocoductor, GitHub y R-Forge.

4. Actualizaciones continuas, la gran comunidad de usuarios hace que se actualice constantemente.

5. R es una plataforma estadística, lo cual ofrece todas las técnicas de análisis de datos. Además de programar nuevos métodos y rutinas estadísticas de una manera fácil y robusta.

6. Los gráficos disponibles en R son de gran calidad y de una versatilidad impresionante.

7. R remplaza la combinación de varios programas para el proceso de análisis de datos, por ejemplo Excel, Minitab, SAS, SigmaPlot, entre otros. Esto no solo resulta en el alto costo de las licencias de múltiples programas, si no también, en la gran cantidad de archivos con diferentes formatos que no podrían leer otros programas estadísticos. En cambio, con solo utilizar R, se puede realizar todo el análisis de datos e inclusive leer archivos de diferentes formatos.

8. R se está convirtiendo en un estándar en la sociedad científica, por hacer figuras de calidad de publicación, además de poder exportarse a diferentes formatos incluidos PDFs.

Desventajas:

1. R tiene una vasta documentación de ayuda, descripción de paquetes y de funciones, y es difícil encontrar información específica en un momento dado.

2. Los mensajes de error que R muestra, no especifican los fallos que se realizan y solo un usuario con cierta experiencia en el uso de R puede saberlo.

3. R es un lenguaje de programación en línea de comando, lo cual no involucra el uso de menús como otros programas estadísticos, esto hace que muchas personas que no están familiarizadas en la programación, les resulte muy difícil migrar a R. Pero esto más que una desventaja es una ventaja, porque al programar se entenderá mejor la base de la estadística y el análisis de datos, comparado con otras personas que no utilizan R.

Por estos motivos se ha elegido el lenguaje de programación R para representar la salida de los datos obtenida en C en .txt. Se realizarán diversas gráficas en R, como se verá en el capítulo de resultados.

1.6. Estructura del documento

Aún por completar (primero escribir todo y luego hacer este apartado)

Tras esta introducción, en el que se han expuesto los objetivos generales que aborda el proyecto, los motivos de su realización adjuntándose un ejemplo en el que se muestran las dificultades de realizar un programa de secuenciado a mano, la metodología a emplear, y el software utilizado, se ha dado una visión general de la temática que tratará el proyecto, y las herramientas que se emplearán para realizarlo, además de una primera justificación de su viabilidad. En el siguiente capítulo se describirán las características del sistema y de las instancias.

2. Descripción del sistema y caracterización de las instancias.

En este capítulo, se detallan las características más importantes del sistema y del problema abordado en este proyecto. Sin duda, corresponde a la etapa de recogida de información y sirvió como base para el inicio del desarrollo del modelo. Esta información fue enviada por la compañía proveniente del proyecto de mayor tamaño que trata también este problema de secuenciado de tareas. Resultó fundamental revisar si estos datos eran correctos debido a qué si se llega con errores a la fase de desarrollo del modelo, la solución obtenida no tendrá validez ninguna y reflejará un escenario distinto a la realidad de la factoría.

2.1. Generalidades

En la planta de la compañía aeronáutica se producen varios componentes para otras plantas. El proceso de producción que nos ocupa, comprende el desmontaje de aviones militares para después montarlos y así obtener un avión para uso civil (falta información)

2.2. La planta

Se van a exponer las características de la planta. Para este caso, disponemos de diferentes áreas donde se pueden realizar las operaciones necesarias para completar las tareas. En general, habrá dos áreas y dos tipos de trabajadores activos. El área A será la empleada por los trabajadores Mecánicos, y el área B, por los trabajadores Estructurales. También existe posibilidad de mezcla de tipo de áreas y tipos de operarios.

Tanto la cantidad de trabajadores, como la capacidad de área de cada tipo, están restringidas, pues son un recurso limitado.

2.3. Caracterización de las instancias.

El proceso está comprendido por tareas, las cuales tienen diferentes especificaciones, enumeradas según su orden de aparición:

1. **Numero de tareas:** número de tareas de las instancias, este número de tareas es empleado por el código para decidir cómo escanear los datos. De esta forma, no será necesario modificar el código para trabajar con instancias de distinto número de tareas.

2. **Número de áreas:** número de áreas en las que se realizarán las tareas

3. **Número de trabajadores:** número de tipos de trabajadores que realizarán las tareas.

4. **Capacidad de las áreas:** recursos de área de cada tipo disponibles. Se escanean según el número de áreas que tenga la instancia.

5. **Capacidad de trabajadores:** recursos de trabajadores de cada tipo disponibles. Se especifican en el programa o se introducen por teclado durante su ejecución como se verá en el capítulo de resultados.

6. **Mínimo y máximo de trabajadores:** para ejecutar una tarea determinada, el número de trabajadores empleados debe estar entre el mínimo y el máximo de trabajadores permitidos para esa tarea.

7. **Precedencias:** restricción de precedencia. Se especifican predecesor, número de sucesores y los sucesores. Determina el orden en el que las tareas deberán programarse, de tal manera, que una tarea no puede secuenciarse hasta que todos sus predecesores han terminado.

8. **Restricción "No Paralela":** impide la ejecución de dos tareas en el mismo intervalo temporal. Se especifican predecesor, número de sucesores y los sucesores.

9. **Restricción "Consecutiva":** fuerza la programación de una tarea a continuación de otra. Se especifican predecesor, número de sucesores y los sucesores.

10. **Número de modos:** es habitual que sólo haya un posible modo de resolución, pero puede haber varios.

11. **Modos:** algunas tareas cuentan con diferentes modos de resolución, es decir, diferentes posibilidades en cuanto a número y tipo de trabajadores, tipo de área y duración. La duración decrece de manera proporcional al número de trabajadores, esto es, a mayor número de trabajadores ejecutando la tarea, menor duración de ésta.

    La forma de emplear los diferentes modos será analizada en el apartado de resultados. Se especifica primero la tarea, después el número del modo y la duración de la tarea. Restan cuatro números, los dos primeros corresponden a las áreas, siendo cada número la cantidad de recursos de tipo área consumidos, en el área uno que corresponde al primer número o en el área dos, que corresponde al segundo. Los dos siguientes números representan lo mismo para el caso de los tipos de trabajador.

En la *Ilustración 3* se encuentran los datos de entrada. Cada número de la enumeración previa, es situado en la imagen. De esta manera, el cuadro con el número uno señala el número de tareas, que en la enumeración anterior era el primer dato. Este esquema es el que se sigue para localizar todos los datos en la *Ilustración 3*.

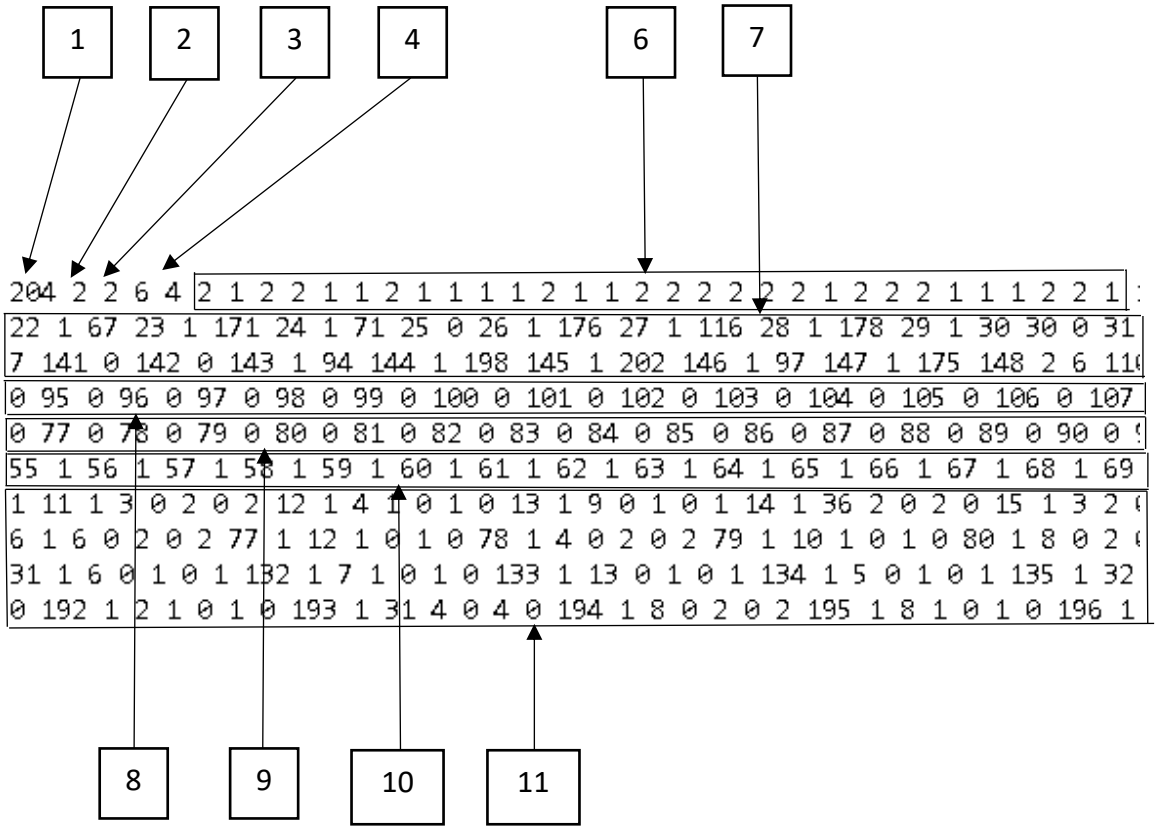

Ilustración 3: datos de entrada recogidos en un .txt,. Los recuadros están numerados de manera que se corresponden con los datos enumerados previamente.

El método para resolver el problema será secuenciar todas las tareas, respetando las restricciones y las capacidades de recursos disponibles. La función objetivo, será el tiempo de realización del proyecto, y se buscará minimizarlo.

En este capítulo, se han expuesto las características del problema y de las instancias a resolver y cómo se recogen sus datos en un .txt de tal manera que el código puede leerlas, sean cuales sean sus características, evitando generar programas diferentes para diferentes instancias. A continuación, se expondrán datos de entrada extra introducidos para obtener los diferentes resultados buscados en el proyecto,

además de los requerimientos del cliente para efectuar el proyecto de manera correcta. Por último, se expondrá el estado del arte.

3. Datos de entrada adicionales y requerimientos

En el capítulo anterior, se realizó una descripción de la planta, de las características del proceso y una caracterización detallada de las instancias. Este capítulo enlaza la información otorgada por la empresa en forma de características con la interpretación de éstas en forma de requerimientos y la de sus pretensiones con objetivos del modelo.

A lo largo de esta tarea fue esencial una comunicación fluida con la empresa, puesto que, en numerosas ocasiones, las características del modelo deseadas no se implementaban de la forma requerida, por lo que ha sido preciso hacer muchas correcciones.

3.1. Datos de entrada

Además de los datos proporcionados por la empresa, ya expuestos en el capítulo dos, será necesario introducir otros datos de cara a la experimentación. Datos extra añadidos son:

1. Porcentaje de ocupación global frontera

2. Tiempo de ciclo (que determina el “Lower Bound”)

3. Variación en el número de trabajadores

4. Método de selección de modos (aleatorio, máximo-mínimo, entre otros)

Estos aspectos serán tratados en profundidad en el apartado de resultados. Pueden ser introducidos por teclado durante la ejecución del programa o estar directamente insertados en el código.

3.2. Requerimientos

Tras la interpretación de los datos proporcionados, en este apartado se muestran los requerimientos dados por la compañía, de obligado cumplimiento para el modelo. Se exponen a continuación los requerimientos.

Requerimiento 1

Implementar un programa que sirva para resolver el problema que nos ocupa en diferentes casos, como, por ejemplo, instancias con distinto número de tareas, distinto número de áreas, de capacidad de áreas, entre otros.

Requerimiento 2

Implementación del heurístico de manera que el criterio para la selección de una tarea sea una regla de prioridad.

Requerimiento 3

Aplicación de aleatorización sesgada, esencial para que el heurístico funcione de la manera deseada, pudiendo así obtener un gran número de soluciones.

Requerimiento 4

Claridad y simplicidad en el código desarrollado, uso de comentarios explicativos e indexación.

Requerimiento 5

Capacidad para resolver instancias con todas las combinaciones posibles de tipo de área y tipo de trabajador.

Requerimiento 6

Efectuar un “Local Search”. Una vez alcanzada una solución bastante buena, realizar este “Local Search” para un número determinado de tareas, para comprobar si se pueden obtener soluciones aún mejores**.**

Requerimiento 7

Salida del Gantt con la programación de las tareas mediante la herramienta de programación R.

Requerimiento 8

Resolución del problema mediante “Lower Bound” y “Porcentaje de Ocupación Global Frontera”. Se expondrán en el apartado de resultados.

Requerimiento 9

Equilibrado de la carga de trabajo, es decir, que cada trabajador tenga una carga de trabajo similar a sus compañeros. Esto evitará problemas para el departamento de Recursos Humanos.

Requerimiento 10

Obtención en la salida de datos de la evolución (mejora) del tiempo de solución a medida que aumentan las iteraciones y la iteración concreta en la que se produce cada mejora.

Requerimiento 11

Obtención en la salida de datos del tiempo de ejecución del programa. De este manera, se consigue analizar la eficiencia del programa frente a diferentes configuraciones de los datos de entrada y de la salida deseada.

Requerimiento 12

Obtención de los porcentajes de ocupación de cada trabajador y porcentaje de ocupación total media.

Requerimiento 13

Obtención de una gráfica en el que se expongan los distintos tiempos de finalización del proceso según la cantidad de trabajadores empleados

Requerimiento 14

Establecimiento de un límite de iteraciones para el programa y de diferentes criterios de parada.

Tras la exposición de los datos de entrada adicionales empleados para obtener las diferentes salidas que se desean obtener en el proyecto, y los requerimientos especificados a cumplir para que el programa ofrezca un desempeño y unos resultados adecuados, se expone el soporte teórico que se encuentra detrás de todo el desarrollo del proyecto, el estado del arte, cuya revisión ha sido imprescindible para el éxito de este trabajo de fin de grado. Además, se justifican los métodos empleados en el proyecto con el apoyo de otros trabajos similares a este encontrados en la literatura.

4. Estado del arte y método de resolución elegido

Una vez se han especificado los datos de entrada adicionales que se van a emplear, así como los requerimientos, se va a exponer el estado del arte y el método de resolución elegido. Se lleva a cabo una breve exposición de lo que comprende la programación lineal, la ciencia heurística, los algoritmos heurísticos y por último sus ventajas e inconvenientes. Se expondrá también brevemente el método lineal.

Por último, se justificarán los métodos de resolución elegidos, con el respaldo de otros trabajos similares encontrados en la literatura.

Para ello, se ha consultado bibliografía como pueden ser: *An Experimental Investigation of Resource Allocation in Multiactivity Projects* de E. M. Davies , en adelante, (Davies, 1973) y *Serial and parallel resource-constrained projet scheduling methods revisited: Theory and computation* de Rainer Kolisch, en adelante, (Kolisch, 1996).

4.1. Problema de optimización de secuenciado

La optimización se define como el proceso que conduce a la solución óptima de un problema. Con solución óptima queremos decir mejor (en algún sentido) que cualquier otra solución.

En programación matemática, en general, y en programación lineal, en particular, se denomina optimización al proceso sistemático de resolución seguido para alcanzar la solución óptima (máximo o mínimo) de la función objetivo y verificar las restricciones de todo tipo que limitan la consecución de ese objetivo.

Sin embargo, la gran mayoría de problemas reales no pueden ser resueltos con algoritmos deterministas, como los que emplea la programación lineal, bien porque sus características no han permitido el desarrollo de ninguna técnica "exacta" que asegure la localización de la solución óptima, o porque aun pudiendo ser las técnicas exactas utilizadas, el tiempo necesario para obtener la solución del problema resulte prohibitivo. La alternativa para estos casos, la constituyen los métodos heurísticos, que mediante

diferentes mecanismos buscan una solución "buena" (aunque no necesariamente óptima) en un tiempo razonable.

Como una primera aproximación, un problema de secuenciado consta de un “Makespam”, o duración del proyecto, de la asignación de recursos para un conjunto de actividades de duración conocida y de las necesidades de recursos, que deben realizarse garantizando algunas relaciones de precedencia.

Sólo se pueden asignar recursos a una tarea a la vez. Por lo tanto, la limitación de recursos, podrá imponer relaciones de precedencia adicionales entre actividades que consumen el mismo recurso, aumentando posiblemente la duración del proyecto. Al mismo tiempo, realizar actividades simultáneamente para ahorrar tiempo usualmente resultará en mayores costos asociados a la mayor cantidad de recursos consumidos.

Estas consideraciones conducen a los siguientes problemas de optimización:

RCSP - Resource Constrained Scheduling Problem: Su objetivo es, con una limitada cantidad de recursos, acortar en lo posible la duración del proyecto. Este es el que se busca resolver en el proyecto. Fue definido por Brucker (Brucker, 1999).

TCSP - Time Constrained Scheduling Problem: Dado un límite de tiempo para la duración del proyecto el objetivo es encontrar el secuenciado que nos proporciona un menor consumo de recursos, si los recursos se suponen disponibles en cantidades ilimitadas a un costo fijo.

4.2. Breve descripción del modelo en términos matemáticos

Los problemas de programación son un tipo de problemas de optimización combinatoria. Estos se definen por un espacio de solución X, que es discreta o que puede ser reducido a un conjunto discreto Y por un subconjunto de soluciones factibles $Y \subseteq X$

Cada solución está asociada con una función objetivo. El objetivo del problema es encontrar una posible solución $y \in Y$ de tal manera que $f(y)$ es minimizada o maximizada.

Utilizando la definición de RCSP propuesta por Koné (Koné, 2009), los dos problemas de programación son un problema de optimización combinatoria definido por una 6-tupla (W , *p* , A, K, B, *p*), donde:

1. W es un conjunto de actividades

2. *p* es un vector de tiempos de procesamiento por actividad

3. A es el conjunto de restricciones temporales

4. R es el conjunto de recursos

5. *b* representa la matriz de demanda (consumo de recursos por actividad)

6. B es el vector de capacidad de recursos - para el RCSP

7. LT es el vector de "Lead Time" de capacidad - para el TCSP

El objetivo es identificar un horario factible, que asigna un comienzo / terminación tiempo (C_i/T_i) para cada actividad, así como una asignación de recursos, teniendo en cuenta las limitaciones temporales y reducir al mínimo el plazo de ejecución total del proyecto (RCSP) o el consumo de recursos (TCSP). En el caso que nos ocupa, será el plazo mínimo de ejecución total del proyecto.

En cuanto a las limitaciones temporales, la notación más común es la activity-on-the-node (AoN), es decir la red de actividades y arcos, donde los nodos representan las actividades y los arcos restricciones de precedencia. También se puede incluir información sobre los tiempos de procesamiento por actividad en la representación gráfica.

Se presentan dos restricciones adicionales:

1. **Restricción "No Paralela":** impide la ejecución de dos tareas en el mismo intervalo temporal. En el caso más estudiado en el proyecto, esta restricción no tiene relevancia.

2. **Restricción "Consecutiva":** fuerza la programación de una tarea a continuación de otra. Complica en cierta medida la programación pues debe de existir disponibilidad de recursos para programar las dos tareas consecutivas, y, además, deben tener todos sus predecesores finalizados.

4.3. Métodos para resolver el RCSP

Como otros problemas de combinatoria, el RCSP puede ser resuelto utilizando varias técnicas.

Se trata de un problema NP-Hard, "nondeterministic polynomial time", es decir, un problema polinómico no determinista que no se puede reducir a un problema más sencillo originando un problema de complicada ("Hard") resolución. Los casos con más de sesenta actividades son difíciles de resolver utilizando métodos exactos. Por lo tanto, una amplia gama de métodos heurísticos y metaheurísticos se han propuesto. En esta sección, ofrecemos un resumen de algunas de las propuestas.

4.3.1. Métodos exactos

La información contenida en este apartado ha sido extrída de (https://upcommons.upc.edu/bitstream/handle/2099.1/3638/31132-1.pdf, 2003).

Los métodos exactos garantizan encontrar la solución óptima, pero si el problema es complejo, el tiempo empleado en hallar y garantizar la solución deseada puede resultar no viable. El problema tratado en este proyecto fue resuelto por Airbus empleando la programación lineal.

Los procedimientos exactos de uso más común son los siguientes:

1. Exploración dirigida o B&B “branch-and-bound” Organiza las soluciones en paquetes, progresivamente más pequeños y determina para cada paquete un indicador de la calidad de las soluciones que contiene. Este indicador permite considerar qué paquetes son los más interesantes para explorar. La exploración consiste en sustituir el paquete por dos o más subpaquetes que en conjunto tienen todas las soluciones de aquél.

2. Programación lineal entera (PLE), binaria (PLB) o mixta (PLM) Casi la totalidad de los problemas combinatorios admiten una formulación como programas lineales enteros o binarios, y especialmente como mixtos. El inconveniente de este método reside en la gran cantidad de variables que surgen en la mayoría de los problemas.

3. Programación dinámica (PD) En lugar de enfocar el problema como la optimización de una función global, se resuelve por etapas en cada una de las cuales se puede tomar una decisión independiente de las decisiones consideradas con anterioridad.

4. Programación dinámica acotada o “bounded dynamic programming” (BDP) La estructura es la misma que en el procedimiento anterior. En este método, se conoce una cota inferior o superior de los elementos que deben integrarse para pasar de un estado a otro. Se compara el mejor valor esperado a partir del estado considerado con el de una solución heurística, cancelando los estados que no ofrecen garantías de mejora.

4.3.2. Heurística y método heurístico

La información contenida en este apartado ha sido extraída de (http://es.slideshare.net/profjavierjuarez/metodo-heurstico-1.com, 2013).

La heurística es la ciencia que estudia los procesos de decisión respecto a un campo de conocimiento concreto, como son las estrategias cognitivas. Su contrapartida

formal en computación es el algoritmo. La palabra heurística proviene de la palabra griega “heuriskein” que significa descubrir, encontrar. Por heurística entendemos una estrategia, método, criterio o truco usado para hacer más sencilla la solución de problemas difíciles.

El conocimiento heurístico es un tipo especial de conocimiento usado por los humanos para resolver problemas complejos. En este caso el adjetivo heurístico significa medio para descubrir. Debido a la existencia de algunos problemas importantes con un gran interés práctico difíciles de resolver, comienzan a surgir algoritmos capaces de ofrecer posibles soluciones que, aunque no consiguen el resultado óptimo, sí que se acercan en un tiempo de cálculo razonable.

Estos algoritmos están basados en el conocimiento heurístico y por lo tanto reciben el nombre de algoritmos heurísticos. Por lo general, los algoritmos heurísticos encuentran buenas soluciones, aunque a veces no hay pruebas de que la solución pueda hallarse en un tiempo razonablemente corto o incluso de que no pueda ser errónea. Frecuentemente pueden encontrarse casos particulares del problema en los que la heurística obtendrá resultados muy malos o que tarde demasiado en encontrar una solución.

Un método heurístico es un conjunto de pasos que deben realizarse para identificar en el menor tiempo posible una solución de alta calidad para un determinado problema. Al principio esta forma de resolver problemas no fue bien vista en los círculos académicos, debido fundamentalmente a su escaso rigor matemático. Sin embargo, gracias a su interés práctico para solucionar problemas reales fue abriendo poco a poco las puertas de los métodos heurísticos, sobre todo a partir de los años 60.

El método heurístico conocido como “IDEAL”, formulado por Bransford y Stein (1984), incluye cinco pasos: Identificar el problema; definir y presentar el problema; explorar las estrategias viables; avanzar en las estrategias; y lograr la solución y volver para evaluar los efectos de las actividades (Bransford and Stein, 1984). El matemático Polya (Polya, 1957) también formuló un método heurístico para resolver problemas que se aproxima mucho al ciclo utilizado para programar computadores.

Actualmente las versiones matemáticas de métodos heurísticos están creciendo en su rango de aplicaciones, así como en su variedad de enfoques. Nuevas técnicas heurísticas son utilizadas a diario por científicos de computación, investigadores operativos y profesionales, para resolver problemas que antes eran demasiado complejos o grandes para las anteriores generaciones de este tipo de algoritmos

4.3.3. Ventajas e inconvenientes del método heurístico

Las principales ventajas y desventajas del método heurístico son:

1. Permite obtener un conjunto muy amplio de soluciones, quedándose con la mejor, siendo esta normalmente lo suficientemente cercana al óptimo.

2. Si no se conoce el óptimo del problema se puede obtener una solución del problema mediante el método heurístico demasiado alejada de un buen valor, siendo esto indeseable.

3. Si se necesita un tiempo demasiado elevado para la resolución del heurístico se estaría violando uno de los principios fundamentales de los algoritmos heurísticos, que su resolución se haga en un tiempo reducido.

4. Los métodos heurísticos generalmente por su tipo de búsqueda nos pueden conducir a errores u operaciones equivocadas, aunque raras veces aparecen los peores casos en la práctica.

5. Algunos heurísticos se pueden contradecir al aplicarse al mismo problema, creando con esto confusión

6. Las soluciones óptimas determinadas por el heurístico pueden hacer menos exhaustiva la búsqueda

4.4. Heurísticos

En 1963, Kelley (Kellley, 1963) publicó un primer heurístico de generación de secuenciados. Desde entonces, se han propuesto un gran número de técnicas de solución diferente. El núcleo de la mayoría de ellos es el "Schedule Generation Schemes" (SGS), es decir, sistema de generación de secuencias en serie. En algunos casos, la generación de los "schedule" o programas destaca por el uso de reglas de prioridad, dando lugar a un conjunto de heurísticos conocido como heurísticos RCSP basados en prioridad.

Los programas de generación de esquemas (programaciones) utilizan un enfoque paso a paso para construir un horario factible, a partir de uno parcial. Esta ampliación progresiva del horario puede hacerse siguiendo dos enfoques: "*activity-oriented series" SGS* donde una actividad es programada en cada paso, y "*time-oriented parallel" SGS* donde en cada paso, un instante de tiempo es considerado y varias actividades pueden ser incluidas en el programa.

4.5. Elección del sistema de generación de programaciones

Se debe elegir uno de entre los dos posibles sistemas de generación de programaciones. En ambos casos, una vez que una actividad ha sido introducida en la secuencia solución, nunca es resecuenciada. El orden de secuenciación puede obtenerse de forma aleatoria o mediante una función de prioridad tal como el requerimiento de recursos, duración de las actividades, entre otros. Cada regla de prioridad define una secuenciación diferente. Estos son los dos tipos de sistemas de generación de programaciones expuestos por Kolisch (Kolisch, 1996):

4.5.1. Sistema de generación de programaciones en serie:

En cada iteración una actividad es seleccionada y programada de entre el conjunto de tareas que están disponibles para programar, y una vez programada entra en el conjunto solución. El criterio para seleccionar la tarea a programar es una regla de prioridad. La

tarea se programa en el instante más temprano posible según la disponibilidad de recursos. Cuando todas las actividades están programadas termina el algoritmo.

4.5.2. Sistema de generación de programaciones en paralelo:

Se define una vez más un conjunto de candidatos, un conjunto de actividades que pueden programarse dado que todos sus predecesores han concluido. Difiere en el sistema de generación de programaciones en serie en el hecho de que en cada iteración, además de definirse este primer conjunto de actividades, se define a continuación un subconjunto dentro de ese conjunto de las actividades que se van a programar en función de la disponibilidad de recursos. De esta forma, en cada iteración es posible que se programen varias tareas, y no sólo una como en el caso de la generación de programaciones en serie.

Según Kolisch, (Kolisch, 1996), el método de secuenciación en serie es bueno para muestras grandes y restricciones moderadas de recursos mientras que el método en paralelo es el adecuado para tamaños pequeños y grandes restricciones de recursos. De esta forma, se concluye que el método de secuenciación adecuado para el proyecto que nos ocupa será el sistema de generación de programaciones en serie.

4.6. Reglas de prioridad

Al generar horarios factibles, con cualquiera de los SGS, normalmente usamos reglas de prioridad para asignar a cada actividad un valor para que la actividad con el valor mayor (o menor) sea elegido. También se debe definir una regla adicional, para utilizar en caso de empate.

Las reglas de prioridad pueden centrarse en diferentes características de la actividad. Los principales grupos son basados en la actividad, basados en la red, las reglas basadas en el camino crítico o las basadas en los recursos.

Tipo	**Siglas**	**Regla elige actividad con el:**
Basadas en actividades	SPT	Tiempo de procesamiento menor
Basadas en actividades	LPT	Tiempo de procesamiento más largo

Basadas en la red	MIS	Mayor número inmediatos sucesores
Basadas en la red	LIS	Menor número inmediatos sucesores
Basadas en la red	MTS	Mayor número de sucesores
Basadas en la red	LTS	Menor número de sucesores
Basadas en la red	GRPW	Mayor rango posicional por peso
Basadas en camino crítico	EST	Menor tiempo temprano de comienzo
Basadas en camino crítico	ECT	Menor tiempo temprano de final
Basadas en camino crítico	LST	Mayor tiempo temprano de comienzo
Basadas en camino crítico	LCT	Mayor tiempo temprano de final
Basadas en camino crítico	MSLK	Mínima holgura
Basadas en recursos	GRR	Mayores requerimientos de recursos

Tabla 3: Principales reglas de prioridad y su criterio de programación de tareas

Si el valor asignado a una tarea sigue siendo el mismo en todo el SGS, decimos que la regla de prioridad es estática y dinámica si es lo contrario.

El heurístico basado en prioridades es de uso frecuente en la práctica puesto que tienen un tiempo de funcionamiento pequeño en general y son fáciles de implementar. Además, se utilizan a menudo para calcular límites superiores o inferiores y soluciones iniciales. Este enfoque, el de heurístico basado en prioridades, es el empleado en este proyecto. A continuación, se expondrá los motivos para la elección de la regla de prioridad GRPW.

4.7. Elección de la regla de prioridad

Tras el análisis del problema a resolver, y el estudio de la literatura disponible, se hace patente que las mejores reglas de prioridad serán:

GRPW Greatest Rank Position Weight: este criterio selecciona actividades mediante la suma de la duración de la actividad y la de todos sus sucesores.

MTS Most Total Successors: este criterio selecciona actividades mediante la suma de todos los sucesores de la actividad

Cooper (Cooper, 1976) posiciona estas dos reglas de prioridad entre las primeras del ranking que hizo con los resultados de un estudio sobre las aplicaciones del secuenciado en serie. Empleó un tamaño muestral de cien. Información extraída de un "paper" de Kolisch (Kolisch, 1996).

Davies y Patterson hicieron un estudio computacional de los heurísticos RCPS (Davies and Patterson, 1975). Los autores comparan heurístico y soluciones óptimas de problemas tipo RCPS. Los resultados globales de 25 algoritmos, es decir, 25 reglas de prioridad, y 144 proyectos, de tamaños muestrales de 27, 51 y 103 actividades, muestran que la distancia media al óptimo para la GRPW fue del 3,3% y la distancia media de la MTS al óptimo fue 3.55%. La GPRW fue la mejor regla de prioridad en este aspecto mientras que la MTS fue la tercera. Información extraída del libro de Slowinski, (Slowinski, 1989).

Hay otras reglas que parecen buenas, como CUMRED CUMulative Resource Equivalent Duration, pero su implementación parece complicada.

Por todo esto, la regla de prioridad seleccionada, es la GRPW.

4.8. Metaheurísticos

Entre todos los metaheurísticos, los más comunes para la RCSP son algoritmos genéticos, búsqueda tabú, recocido simulado y colonia de hormigas.

4.8.1. Algoritmos genéticos

Primero promovido por John Holland, (Holland, 1995), los algoritmos genéticos (GAs) están basados en las ideas evolucionistas de la selección natural y genética. GAs está diseñado para simular procesos en un sistema natural necesarios para la evolución, específicamente aquellos que siguen los principios establecidos por Charles Darwin de la supervivencia del más fuerte. Como tales, representan una explotación inteligente de una búsqueda al azar dentro de un definido espacio de búsqueda para resolver un problema.

Estos algoritmos consideran un conjunto de programas factibles, o población. A partir de éstos, nuevas soluciones se calculan por el apareamiento de dos ya existentes (cruce) o por modificar uno ya existente (mutación). Una vez que se producen las nuevas soluciones, la mejor de las soluciones son elegidas, según el valor de la función objetivo. Los más aptos (mejores) soluciones sobreviven, convirtiéndose en la próxima generación y el resto se eliminan.

Ha sido uno de los metaheurísticos más comúnmente utilizados para resolver el problema. Dos ejemplos son los propuestos por Hartmann (Harmann, 1998) y, más recientemente, un mejor algoritmo por Valls (Valls, 2003).

4.8.2. Búsqueda tabú

El concepto básico de búsqueda tabú fue descrito por Glover, (Glover, 1989). Se trata de un método de investigación que evalúa todas las soluciones de un área y elige la mejor opción, con el fin de proceder de él. Este método tendría el riesgo de volverse cíclico, volviendo a la situación previa. Para evitar este problema, una lista de tabú se configura como una forma de memoria para el proceso de búsqueda. Esta lista puede prevalecer solamente si el área correspondiente dará lugar a una nueva solución global mejor.

Artigues (Artigues, 2003), Klein (Klein, 2000) y Nonobe y Ibaraki (Nonobe and Ibaraki, 2002) han propuesto algunos de los más recientes algoritmos de búsqueda tabú para la RCSP.

4.8.3. Optimización de Colonia de hormigas

Optimización de Colonia de hormigas (ACO) toma inspiración de la conducta de forrajeo de algunas especies de hormigas, (Marco Dorigo, 1996). Estas hormigas depositan feromonas en el suelo para marcar algún camino favorable que debe ser seguido por otros miembros de la colonia. La Optimización de Colonia de hormigas explota un mecanismo similar para resolver problemas de optimización.

En el caso de RCSP, cada solución está representada por una lista de actividades, que describe el orden en que las actividades se han incluido en la solución durante el SGS. El valor de feromona en este caso está relacionado con lo prometedor que parece poner la actividad *w* en la programación, teniendo en cuenta el valor objetivo de la función de las anteriores que incluyeron esa opción.

Merkle (Merkle, 2006) presentó el primer ACO para un RCSP.

5. Descripción del modelo

En el anterior epígrafe se expuso el estado del arte y el método de resolución elegido. Para que el software pueda llevar a cabo los correspondientes cálculos, los correspondientes requerimientos han de ser traducidos a código.

A lo largo de este capítulo, se expone el pseudocódigo.

5.1. Explicación extendida del pseudocódigo

Mientras el número de iteraciones sea menor que “solucionesdeseadas” (siendo “solucionesdeseadas” la cantidad de soluciones que queremos obtener antes de que el código se detenga) se realiza la mayor parte de tareas del código

S_0 es el set de actividades iniciales a programar, es decir, las que no tienen predecesor. Mientras que i sea menor que n (siendo n el número total de actividades/tareas e i el número de las tareas programadas hasta el momento). Se ejecutan una serie de tareas.

Se escoge una actividad del set S_i. Esto se realiza usando una regla de prioridad. La regla de prioridad GRPW asigna un peso a cada actividad de cara a decidir la probabilidad con la que ésta se puede programar. Esto se lleva a cabo para poder realizar la aleatorización sesgada, es decir, cuanto mayor es el peso (o menor, para ciertas reglas de prioridad), mayor es la probabilidad de que la actividad sea escogida para su programación.

Mientras que la cantidad necesaria de recursos para hacer la actividad estén disponibles (Trabajadores mecánicos o estructurales, las áreas A o B), y las restricciones, se cumplan (la restricción que impide que dos tareas se ejecuten en paralelo y la restricción de tareas consecutivas, es decir, la de lapso de tiempo cero) durante un tiempo se hacen estas tareas.

Cálculo del menor tiempo en el que la actividad puede ser programada. Se programa la actividad en el intervalo (respetando duración de la actividad…). Se actualizan los recursos mediante la disminución de los recursos empleados por la actividad en el intervalo en el que la actividad ha sido programada.

Añadir al siguiente set de actividades todos los sucesores cuyos predecesores ya hayan sido programados, de la actividad que acaba de ser programada. Eliminar la actividad que acaba de ser programada del conjunto de tareas programar y añadirla al conjunto solución.

Si el tiempo de realización del proyecto es el menor hasta ahora, almacenar los resultados en la "solucionfinal" (siendo "solucionfinal" la salida del programa)

Si se han realizado más de un número de iteraciones sin una mejora del tiempo de realización del proyecto, se vuelca la salida en un .txt. Esto es el criterio de parada.

Se vuelca en un fichero .txt la solución final para representarla gráficamente mediante ggplot2 en R con lo que finaliza el programa.

5.2. Pseudocódigo

WHILE iteraciones menor que "solucionesdeseadas":

WHILE i menor que n:

DO escoger una actividad del set S_i empleando la regla de prioridad GRPW

WHILE existan recursos y las restricciones se cumplan durante un tiempo:

DO calcular el menor tiempo en el que la actividad puede ser programada.

ENDWHILE.

Programar la actividad en el intervalo y actualizar los recursos, enviando ésta al conjunto solución. Añadir al siguiente set las actividades a programar.

ENDWHILE.

IF tiempo de realización del proyecto es el menor hasta ahora, almacenar los resultados en la "soluciónfinal"

IF se han realizado más de un número de iteraciones sin una mejora del tiempo de realización del proyecto, salir de este bucle.

ENDWHILE.

Volcar en un fichero .txt la solución final para representarla gráficamente mediante R.

6. Implementación

7. Resultados

8. Análisis de Resultados

9. Futuras mejoras