Contenido

[1. Introducción 2](#_Toc463426523)

[1.1. Resumen ejecutivo 2](#_Toc463426524)

[1.2. Abstract 3](#_Toc463426525)

[2. Descripción del problema 4](#_Toc463426526)

[2.1 Resource and Time Constraint Project Scheduling 4](#_Toc463426527)

[2.2. Métodos para resolver el RCSP 6](#_Toc463426528)

[2.2.1. Heurísticos 6](#_Toc463426529)

[2.3 Regla de prioridad elegida 9](#_Toc463426530)

[2.4. Estructura de los datos 10](#_Toc463426531)

[2.5. Restricciones 11](#_Toc463426532)

[3.1 Algoritmo 1 12](#_Toc463426533)

[3.1.1 Pseudocódigo 12](#_Toc463426534)

# **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, 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 optimo, 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 “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 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.

# Introducción

## Presentación

La producción es la elaboración de un producto mediante el trabajo. En vistas del crecimiento de la población 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 correspondiente a las tareas a realizar de una serie de piezas terminadas. 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 optimo, 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.

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.

TAREA 6

TAREA 3

TAREA 4

TAREA 5

TAREA 2

TAREA 8

TAREA 7

TAREA 1

**Figura 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 más gruesos** 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 una regla de prioridad, GRPW, es decir, “Greatest Rank Position Weight”, que significa, mayor ranking según el peso. Esta regla consiste en asignar a la tarea que este en el conjunto de tareas que se pueden programar por haberse programado 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 capítulos posteriores.

3

8

15

TAREA 6

TAREA 3

TAREA 4

TAREA 5

TAREA 2

TAREA 8

TAREA 7

TAREA 1

3

5

11

18

10

**Figura 1**: Esquema del “case 8.3”. En esta imagen se pueden ver los valores de los pesos de cada tarea.

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. Además, esta resolución es impracticable para casos con más tareas y más restricciones. En este proyecto se resuelve el secuenciado de proyectos de hasta 204 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 programar de la tarea con mayor peso, si es posible, no tiene por que proveer la mejor solución. Una de las bondades del heurístico es que no es determinista, y permite explorar un amplio conjunto de soluciones, y quedarse 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.

# Descripción del problema

## 2.1 Resource and Time Constraint Project Scheduling

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

• **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.

Sin embargo, el TCSP puede considerarse una variante de la RCSP. 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

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

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

• W es un conjunto de actividades

• *p*es un vector de tiempos de procesamiento por actividad

•A es el conjunto de restricciones temporales

• R es el conjunto de recursos

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

• B es el vector de capacidad de recursos - para el RCSP

• 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 (*/*) 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 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.

Aunque en la práctica los plazos están a menudo limitados en los proyectos, TCSP se ha tratado muy pocas veces. En la mayoría de los casos, se ha considerado combinado con restricciones de recursos, como es el caso de programación con restricciones de recursos y tiempo.

## 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, los casos con más de 60 actividades son difíciles de resolver utilizando métodos exactos. Por lo tanto, una amplia gama de heurísticas y metaheurísticas métodos se han propuesto. En esta sección, ofrecemos un resumen de algunas de las propuestas.

### 2.2.1. Heurísticos

En 1963, Kelley [Kel63] publicó un primer heurístico de generación de programaciones. 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). 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.

**Schedule Generation Schemes**

Los programas de generación de esquemas (programaciones) (SGS) 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.

**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 REGLA ELIGE LA 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

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.

La heurística basada en prioridades es de uso frecuente en la práctica puesto que tienen un tiempo de funcionamiento pequeño en general y son fácil de implementar. Además, se utilizan a menudo para calcular límites superiores o inferiores y soluciones iniciales.

**Metaheurísticos**

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

**Algoritmos genéticos**

Primero promovido por John Holland en 1995, [Hol75], 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 definidoespacio 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 más comúnmente utilizados metaheurísticos para el problema. Dos ejemplos son los propuestos por Hartmann [Har98] y, más recientemente, un mejor algoritmo por Valls [VV03].

**Búsqueda tabú**

El concepto básico de búsqueda tabú fue descrito por Glover ([Glo89a] y [Glo89b]). 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 [AMR03], Klein [Kle00] y Nonobe y Ibaraki [KN02] han propuesto

algunos de los más recientes algoritmos de búsqueda tabú para la RCSP.

**Optimización de Colonia de hormigas**

Optimización de Colonia de hormigas (ACO) toma inspiración de la conducta de forrajeo de algunas especies de hormigas, [MD96]. 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 soluciones anteriores que incluyeron esa opción.

Merkle [DM6] presentó el primer ACO para un RCSP.

## 2.3 Regla de prioridad elegida

The best Priority Rules for the problem in my opinion are:

GRPW Greatest Rank Positional Weight: this rule selects activities by summing up the duration of the activity and the duration of all its successors.

MTS Most Total Successors: it counts all the successors the activity has.

Cooper (1976) positioned these priority rules between the first positions of a ranking (of the best priority rules) he did with the results of a study about the sampling applications of the serial method. He employed a sample size of 100. Information extracted from Rainer Kolisch’s “Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation” 1996

The paper by Davies & Patterson (1975) is a computational study on RCPS heuristics. The authors compared heuristic and optimal solutions for RCPS problems. Global results of 25 algorithms on 144 projects (with sample sizes of 27, 51 and 103 activities) showed that GRPW average distance to optimum was 3.30% and MTS average distance to optimum was 3.55%. GRPW was the best priority rule in this aspect and MTS, the third. Information extracted from Slowinski’s “Advances in project scheduling” 1989.

There are another rules that could be good too. For instance, CUMRED CUMulative Resource Equivalent Duration, but its implementation seems difficult. However, I may try it.

## 2.4. Estructura de los datos

El formato en el que se han volcado los datos desde Excel es .txt. Esto permite su lectura de manera sencilla en el programa desarrollado en C. Estas instancias contienen todos los datos que necesita el programa para sacar una solución correcta, respetando todas las restricciones.

Los datos de las instancias, necesarios para resolver este problema de secuencia con recursos limitados, son:

* Número de tareas: número de tareas a secuenciar
* Número de áreas: número de áreas distintas, cada tarea debe realizarse en el área que tiene asignada
* Número de trabajadores: número de trabajadores distintos, cada tarea debe realizarla el tipo de trabajador que tiene asignado
* Capacidad de área: cada área tiene una capacidad máxima asignada, es decir, el máximo de operarios que pueden trabajar simultáneamente en ella
* Capacidad de trabajadores: cada tipo de trabajador tiene una capacidad máxima asignada, es decir, el número de trabajadores de un tipo determinado no puede superar su capacidad máxima en ningún momento
* Mínimo de operarios por tarea: para la realización de cada tarea será necesario una cantidad mínima de operarios trabajando simultáneamente
* Máximo de operarios por tarea: para la realización de cada tarea no será posible superar este límite en ningún momento
* Precedencia: número de sucesores y sucesores de una determinada tarea
* “Non-parallel”: tareas que no pueden solapar los intervalos temporales en los que se programan
* Restricción “Consecutive”: tareas que deben programarse inmediatamente después de otras
* Modos: cada tarea puede tener hasta cuatro modos de resolverse. En cada modo se especifica la duración de la tarea y los operarios y capacidad de área necesarios para acometerla.

## 2.5. Restricciones

Como ya se ha comentado, este problema de secuenciado está enfocado a resolver el problema con el mínimo coste, pues hoy en día la industria aeronáutica, tiene su Lead Time fijado por la producción esperada en lugar de por la demanda del cliente. Este enfoque no es el habitual. Por tanto, no existe una restricción para el “Makespam” y lo que si hay son restricciones en los recursos, de cara a abaratar costes. Las restricciones de las que consta el problema son las siguientes:

* Número mínimo y máximo de operarios de un tipo determinado por tarea
* Número máximo de operarios por tipo
* Número máximo de operarios que pueden ocupar un área de trabajo determinada
* Precedencias
* “Non-parallel”, es decir, tareas que no pueden solapar los intervalos temporales en los que se programan
* “Time-lag=0”, es decir, tareas que deben programarse inmediatamente después de otras

## 3.1 Algoritmo 1

### 3.1.1 Pseudocódigo

Pseudocódigo para el TCSP. Programa de generación de programaciones en serie (usando un solo paso, es decir, una sola regla de prioridad) con reglas de prioridad y aleatorización sesgada.

1. DURANTE\_2 1 hasta solucionesdeseadas (siendo solucionesdeseadas la cantidad de soluciones que queremos obtener antes de que el código se detenga):
2. Siendo E1 el set de actividades sin predecessor:
3. DURANTE\_1 1 hasta n (siendo n el número total de actividades/tareas) HACER:
4. Escoger una actividad del set. Esto se realiza empleando 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 realizer 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.
5. MIENTRAS 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 coonsecutivas, es decir, la de lapso de tiempo cero) durante un tiempo, HACER:
6. Calcular el menor tiempo en el que la actividad puede ser programada.
7. FINALDELMIENTRAS
8. Programar la actividad en el intervalo (respetando duración de la actividad…)
9. Actualizar los recursos mediante la disminución de los recursos empleados por la actividad en el intervalo en el que la actividad ha sido programada.
10. 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.
11. FINALDELDURANTE\_1
12. SI el tiempo de realización del proyecto es el menor hasta ahora, almacenar los resultados en la soluciónfinal (siendo soluciónfinal la salida del programa)
13. SI se han realizado más de 200 iteraciones sin una mejora del tiempo de realización del proyecto, salir de este bucle (FINDELDURANTE\_2)
14. Volcar en un fichero .txt la soluciónfinal para representarla gráficamente mediante ggplot2 en R