Contenido

1. Introducción

1.1. Resumen ejecutivo

La industria aeronáutica ha sufrido, en los últimos veinte años, un cambio profundo en su contexto industrial. Debido a la entrada en el mercado civil de nuevos actores y unido a la reducción del gasto militar que acompañó el final de la guerra fría, la eficiencia y la competitividad han ganado importancia para la permanencia en el sector. Los sistemas de producción lean y la estandarización se han utilizado masivamente como método de mejora continua. Dentro de este contexto y debido a su impacto en todos los ámbitos de la empresa, la mejora de la planificación a todos los niveles es un facilitador fundamental. Por tanto, ser capaz de producir una planificación de detalle factible y exacta se ha convertido en prioritario.

Además, el secuenciado de tareas con recursos limitados es un problema de optimización NP-complejo. Más aún, es uno de los problemas más difíciles de tratar. Tanto por su relevancia industrial como por su dificultad técnica, la resolución de problemas de secuenciado con recursos limitados está siendo objeto de numerosas investigaciones. A pesar de esto, los problemas de secuenciado que suelen ser estudiados en la literatura no cubren todas las características de muchos problemas reales, entre los que se encuentra el secuenciado de tareas dentro de las plataformas de montaje aeronáuticas.

En este trabajo se ha utilizado el programa C para resolver un problema de secuenciado de actividades multimodo en un entorno de tiempo limitado.
Se trata de formulaciones de aplicabilidad directa a casos reales de diferentes industrias, entre las que se encuentra la industria aeronáutica.

1.2. Abstract

2. 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 $Y \subseteq X$

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 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 (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 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.

2.2.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