Sección de Matemáticas
Universidad de La Laguna

Ignacio Domínguez Espinosa

Una Aproximación a los Problemas de Planificación y Secuenciación

An Approach to Scheduling and Sequencing Problems

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2023

Dirigido por
David Alcaide López de Pablo

David Alcaide López de Pablo
Departamento de Matemáticas,
Estadística e Investigación
Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Quiero agradecer a familia y amigos por apoyarme a lo largo de toda mi etapa universitaria.

Indudables agradecimientos al director de este trabajo, el Profesor David Alcaide, por introducirme al mundo de los problemas de planificación, además de por su apoyo y su presencia en los momentos cruciales del desarrollo de este documento.

Ignacio Domínguez Espinosa
La Laguna, 10 de julio de 2023

Resumen · Abstract

Resumen

En este documento se realiza un acercamiento a los problemas de Planificación y Secuenciación. Se aplican herramientas que utiliza el campo de la Investigación Operativa para afrontar algunos de los problemas, trabajando además con una clasificación paramétrica de estos.
Se profundiza en el problema de Secuenciación y Ordenación Estocástico Jerárquico, enunciando diversos resultados y procedimientos algorítmicos relevantes en la literatura para resolver este problema.
Por último, se aporta una nueva programación en Python 3.8. de los procedimientos algorítmicos mencionados, sentando las bases para investigar otros problemas similares en un futuro próximo.

Palabras clave: *Planificación y Secuenciación – Planificación Determinística – Planificación Estocástica – Problema de Ordenación Secuencial Estocástico Jerárquico*

Abstract

This document provides an approach to the problems of Scheduling and Sequencing. It applies tools from the field of Operational Research to tackle some of these problems, while also working with a parametric classification of them.
It delves into the Hierarchical Stochastic Sequential Ordering problem, examining various relevant results and algorithmic procedures found in the literature to address this problem.
Furthermore, a new implementation in Python 3.8 is provided for the mentioned algorithmic procedures, laying the foundations for investigating other similar problems in the near future.

Keywords: *Scheduling and Sequencing – Deterministic Scheduling – Stochastic Scheduling – Hierarchical Stochastic Sequential Ordering Problem*

Contenido

Introducción

En una sociedad en la que organizar y optimizar tareas se está volviendo tendencia entre los diferentes sectores, inevitablemente aparecen dificultades en el ajuste de horarios entre máquinas, ordenadores o empleados. Muchas empresas deben abordar este tipo de impedimentos careciendo de las herramientas adecuadas y llegando a resultados incorrectos o ineficientes. En este contexto, surgen los problemas de Planificación y Secuenciación, cuya resolución permite superar mejor estas dificultades.

En el espíritu del denominado proceso de modelización [15] tan utilizado en la investigación científica, aparecen los modelos de Planificación y Secuenciación para resolver estos problemas. Estos constituyen un conjunto variado de modelos que tienen como objetivo principal asignar, a lo largo de un periodo de tiempo y de acuerdo a determinados criterios de optimización, diversas tareas a las máquinas o trabajadores para que las lleven a cabo dentro de un plazo. A partir de esta idea, surgen una infinidad de modelos que pretenden resolver problemas aplicados a la realidad con diferentes requisitos, limitaciones u objetivos.

Estos modelos se popularizaron con la creación Investigación Operativa después de la Segunda Guerra Mundial, y tienen su auge a partir de 1950 con los trabajos de Garey y Johnson (1954) [9] entre otros. Posteriormente, surgen autores como Baker (1974) [6], French (1982) [8], Graham et al. (1979) [10] y Lawler (1982) [12] que aportan las nociones básicas a esta rama, además de clarificar y explorar conceptos más complejos.

En este Trabajo de Fin de Grado se realiza un acercamiento a los problemas de Planificación y Secuenciación, iniciando con una definición formal de conceptos como las máquinas, los trabajos y los criterios de optimización, de acuerdo a los estándares más comunes existentes en la literatura. Con estos conceptos, se caracterizan los problemas utilizando varios criterios como el de $\alpha|\beta|\gamma$, procedentes de los artículos de Graham et al. [10].

Posteriormente, desarrollamos un modelo para el HSSOP ("Hierarchical Stochastic Sequential Ordering Problem") o Problema de Ordenación Secuen-

cial Estocástico Jerárquico, construyendo la versión determinística y concluyendo con el modelo estocástico, donde datos como los tiempos de procesamiento de los trabajos pasan a ser variables aleatorias. Previo a finalizar el capítulo, introducimos varios resultados relacionados con la complejidad computacional del problema, teoremas y estimaciones que ayudan al desarrollo de procedimientos para determinar soluciones óptimas.

A continuación, se citan y presentan dos algoritmos para la resolución del problema: un algoritmo exacto que explora todas las posibles combinaciones de los trabajos en busca del óptimo, y un algoritmo heurístico que trabaja con una solución inicial y, mediante diferentes reordenaciones de esta, se quiere llegar a otra mejor en un menor tiempo de ejecución. Estos algoritmos están publicados en Alcaide López de Pablo et al. (2003) [2].

Por último, se lleva a cabo una experiencia computacional del problema anterior, programando los algoritmos en Python, y realizando comparaciones para verificar el funcionamiento de la heurística.

Este trabajo está estructurado en los siguientes capítulos:

- **Capítulo 1.** Nociones Básicas de los Problemas de Planificación y Secuenciación 1. Se presentan los conceptos esenciales en este tipo de problemas, además de una clasificación en función de las máquinas, los trabajos y los criterios de optimalidad.

- **Capítulo 2.** Estudio detallado del HSSOP 2. Se enuncia el Problema de Ordenación Secuencial Estocástico Jerárquico, partiendo inicialmente del problema determinístico, y enunciando después el caso estocástico. Además, incluye un apartado de análisis de complejidad computacional, que justifica la necesidad de un algoritmo heurístico, y otro en el que se enuncian varios resultados que ayudan al desarrollo de dichos algoritmos.

- **Capítulo 3.** Algunas Estrategias de resolución del problema 3. Se citan y presentan dos algoritmos junto a sus pseudocódigos y explicaciones, que pretenden encontrar soluciones del HSSOP. Estos fueron creados y examinados en Alcaide López de Pablo et al. (2003) [2].

- **Capítulo 4.** Aplicación de un Modelo para resolver el HSSOP 4. Desarrollamos una experiencia computacional en la que se programan los algoritmos de resolución del problema. Se comparan el rendimiento y la precisión de ambos para verificar, de manera computacional, el funciónamiento de la heurística.

En el capítulo 4, se muestran una serie de tablas que confirman las similitudes de los algoritmos, y en el apéndice se proporciona parte del código programado para la construcción de los procedimientos.

1

Nociones Básicas de los Problemas de Planificación y Secuenciación

En este capítulo se desarrollan los conceptos utilizados comúnmente en la formulación y modelización de los problemas de Planificación y Secuenciación. Mostramos una clasificación clásica de los modelos, además de algunos ejemplos introductorios a esta rama de la Investigación Operativa.

1.1. Conceptos fundamentales

Los problemas reales de Planificación y Secuenciación son muy diversos y aparecen en múltiples situaciones en el entorno de trabajo; desde gestionar la jornada laboral de los trabajadores de una empresa, hasta la organización de la maquinaria en un proceso industrial.

Al igual que se describe en el boletín [4], este tipo de problemas se pueden resumir en tres componentes: ¿qué?, ¿quién? y ¿para qué? El ¿qué? explica qué acciones se tienen que llevar a cabo. El ¿quién? cuestiona qué o quiénes son los que llevan a cabo estas acciones. Por último, el ¿para qué? pretende definir los objetivos del problema, dando criterios para poder discernir la mejor entre las posibles soluciones.

Esta descripción da lugar a una gran variedad de problemas a resolver. En la literatura, las componenetes se suelen definir mediante las tres ideas siguientes:

- **Máquinas.** Definimos la noción de máquina como el ser u objeto que puede llevar a cabo los trabajos en los que se subdivide el problema que se quiere planificar. En la realidad, las maáquinas equivalen a los empleados, los ordenadores o los aparatos de una empresa que son capaces de completar dichas actividades. Este concepto es lo que anteriormente definimos como el ¿quién? dentro de los problemas.

  Si en el problema intervienen m máquinas, se suelen denotar como M_i, $\forall i \in \{1, 2, ..., m\}$.

- **Trabajos.** Identificamos la idea de trabajo como el conjunto de pasos o fases que se tienen que completar para finalizar el problema. Cada una de estas acciones las deben llevar a cabo las máquinas y pueden existir restricciones o limitaciones en función del problema. Esto equivale a el ¿qué? entre las componentes.

  Si se llevan a cabo n trabajos, los denotamos como J_j, $\forall j \in \{1, 2, ..., n\}$. Cabe destacar que cada uno de los trabajos se pueden subdividir en operaciones, donde O_{ij} indica la parte del trabajo j que se realiza en la máquina i, $\forall j \in \{1, 2, ..., n\}$, $\forall i \in \{1, 2, ..., m\}$.

  También se debe considerar si se permite o no realizar más de un trabajo en la misma máquina a la vez. La negación de esto se denomina hipótesis de no simultaneidad de trabajos.

- **Criterios de optimalidad.** Por último, al ser un problema originado en la Investigación de Operaciones, se necesita definir una serie de funciones a minimizar o maximizar, para así encontrar un óptimo. Cualquier problema que se quiere completar, siempre tiene un propósito, ya sea terminar en un plazo o minimizar costos; estos objetivos suelen ser modelizados mediante estas funciones o criterios. Concretamente, esta idea es el ¿para qué? descrito previamente.

  Se pueden tener en cuenta K criterios, donde a cada uno de ellos se le asigna una función de coste f_j^k, que depende de la ordenación de cada trabajo j y de cada criterio k. Estas funciones suelen estar definidas en cada instante de tiempo y son las que deciden cuál de todas las posibles planificaciones es la mejor.

1.2. Clasificación

A fin de comenzar con la modelización, es necesario primero llevar a cabo diversas clasificaciones que ayudan a caracterizar muchos de los problemas de Planificación y Secuenciación.

Una primera aproximación para intentar agrupar los problemas es en función de los datos y las variables que intervienen. Si estos parámetros son constantes conocidas o estimadas, nos encontramos ante un problema de carácter determinístico. Si por lo contrario se da la existencia de variables aleatorias dentro del problema, decimos que estamos ante un problema de Planificación y Secuenciación estocástico. En este último campo, caben destacar los trabajos de Pinedo (2002, 2005) [17] [18], quién ha contribuido mucho al estudio de los problemas de Planificación estocástica.

También podemos diferenciar estos problemas en función del número de criterios que se tienen que verificar. Hay autores como Hoogeveen (1992) [11] que plantean problemas de planificación multicriterio.

En muchas ocasiones, es posible distinguir entre los problemas de planificación cíclica o periódica, y aquellos problemas que no tienen esta propiedad. Con los problemas cíclicos, hay que decidir sobre una planificación que se repite de manera periódica cada cierto tiempo denominado periodo. En los procesos industriales se suelen combinar los problemas cíclicos con los que presentan robots o brazos mecánicos, que en muchas ocasiones, no se pueden considerar como máquinas en la planificación. Luego, la existencia o no de este tipo de condiciones se tiene que considerar y autores como Levner (2007, 2010) [13] [14] cubren eficazmente este tema.

Una última clasificación básica puede ser en base a la complejidad computacional de estos problemas. Al ser muchos y muy variados, la complejidad y el tiempo de resolución varían. Desde problemas que se resuelven en tiempo polinomial (P) hasta problemas más "difíciles"(NP-duros). Para saber más de esta rama de la computación se puede consultar Garey y Johnson (1979) [9].

1.2.1. Esquema $\alpha|\beta|\gamma$:

Con el objetivo de crear una clasificación general que cubra todos los problemas posibles, Graham et al. (1979) [10] confeccionaron un repertorio fundado en las tres ideas descritas en la sección 1.1, que catalogan las características de las máquinas con el parámetro α , las de los trabajos con el parámetro β y las de los criterios con el parámetro γ. Este catálogo se ha ido estudiando y actualizando con el paso de los años y con la aparición de nuevos problemas de planificación de interés.

Parámetro α:

Pretende describir las características de las máquinas. Estas se clasifican en función del número de máquinas que intervienen en el problema y de la tipología de estas.

Si tratamos de organizar los trabajos en una sola máquina, nos introducimos en los Problemas de Secuenciación y Ordenación, mientras que si trabajamos con más de una, tratamos con los Problemas de Planificación, en los que tenemos que tener en cuenta factores como el paralelismo, que plantea la idea de realizar diferentes trabajos simultáneamente en distintas máquinas. Si continuamos por esta segunda rama, hay dos alternativas: todas las máquinas son capaces de realizar todos los trabajos o existe especialización entre ellas.

Al tratar con máquinas no especializadas, estas pueden ser completamente idénticas entre si respecto de los tiempos de proceso de los trabajos. Otra opción

puede ser que existan diferencias en los tiempos, pero a causa del funcionamiento de la propia máquina, independientemente del trabajo que realice. A este tipo se les llama uniformes, y un ejemplo de ello pueden ser dos máquinas, una más antigua que la otra, y por tanto, esta puede tardar más en realizar sus tareas que la otra máquina más nueva, dándose estas diferencias. Una última opción son las máquinas no relacionadas, que valora cuándo existen diferencias en los tiempos de proceso que dependen tanto de la máquina como de los trabajos que esta realiza.

Por otra parte, cuando se habla de la existencia de especialización entre las máquinas, nos referimos a que no todas son capaces de completar todas las tareas de cada problema, extiste para cada tarea un subconjunto de máquinas que son capaces de llevarla a cabo. Se dan varias modalidades como los sistemas Flow shop, que son aquellos en los que el procesamiento de cada uno de los trabajos tiene que realizarse siguiendo un orden prefijado. Cada trabajo debe de pasar por cada una de las máquinas en orden y si existe alguna que no se utilice para este trabajo, decimos que el tiempo de proceso del trabajo en la máquina es nulo. Otra variante son los sistemas Open shop, en los que todos los trabajos deben de pasar por todas las máquinas, pero a diferencia del modelo anterior, en este no consideramos ningún orden de preferencia entre los trabajos. Por último, destaca la variante de Job shop, donde cada trabajo tiene un subconjunto de máquinas y el orden de los procesos en cada una no es relevante. Este grupo de problemas tiene la singularidad de que permiten que los trabajos puedan pasar varias veces por la misma máquina.

En la figura 1.1 se pueden ver la tipología de las máquinas. Nótese que ponemos entre paréntesis la letra por la que tenemos que sustituir α. Además, en la sección 1.2.1 vemos algunos ejemplos de la notación clásica de Graham.

Parámetro β:

Expone las reglas a las que están sujetos los trabajos dentro del problema. Cualidades como si se permiten interrupciones en los trabajos para continuarlos más adelante; si existen relaciones de precedencia entre trabajos, donde para poder comenzar un trabajo es necesario que se hayan finalizado otros; si hay instantes de disponibilidad o todos los trabajos se pueden empezar desde el inicio de la planificación, dando a problemas dinámicos si existen o problemas estáticos si no; se pueden dar cotas en el número de operaciones en el problema; y pueden existir recursos adicionales a tener en cuenta. Son algunos ejemplos de restricciones en los trabajos que añaden particularidad a cada problema. Se suele denotar substituyendo β por un vector con las diferentes condiciones.

En la figura 1.2 se denotan varias características de los trabajos. Al igual que con α, ponemos entre paréntesis los diferentes valores que puede tener el vector β.

Parámetro γ:

En esta variable se almacena las funciones objetivos a minimizar utilizando variables como los tiempos de completación de los trabajos (Se suelen denotar como C_j), o bien cantidades que indican si estos son tardíos o no, es decir, si se terminan a tiempo o exceden su fecha límite de finalización. Las funciones más utilizadas son del tipo máximo ($\text{máx}_j\, f_j = f_{\text{máx}}$) o de tipo suma ($\sum_j f_j$).

Ejemplos:

A continuación, mostramos algunos ejemplos de como utilizar el esquema $\alpha|\beta|\gamma$. Uno de los problemas más sencillos que podemos plantear con esta notación es el problema de Secuenciación $1||C_{\text{máx}}$, en el que 1 indica que solo interviene una máquina, no hay condiciones respecto de los trabajos y el criterio a optimizar es minimizar el máximo de los tiempos de completación, es decir, minimizar la duración total del problema (pues el máximo de los tiempos de completación es el último trabajo que termina de procesarse y este coincide con la finalización del problema).

Otro ejemplo puede ser $O|pmtn, prec|\sum_j T_j$, que es un sistema Open shop de m máquinas (siendo m un número arbitrario, aunque se puede especificar), donde se permiten las interrupciones en los trabajos, existen relaciones de precedencia y se busca minimizar la suma de los T_j, que cada uno modeliza cuánto de tardío es el trabajo j.

Un último ejemplo puede ser, en el campo estocástico, $1|prec, rj|E[C_{\text{máx}}]$, donde tenemos una máquina, los trabajos tienen relaciones de precedencia y tiempos de disponibilidad y se busca minimizar el valor esperado del tiempo de completación del último trabajo. En los problemas estocásticos trabajamos en términos probabilísticos y en muchas ocasiones podemos incluir factores como la esperanza de las variables aleatorias en las funciones a optimizar. En concreto, en los próximos capítulos desarrollamos la teoría para resolver el HSSOP ("Hierarchical Stochastic Sequential Ordering Problem") o el Problema de Ordenación Secuencial Estocástico Jerárquico, que tiene cualidades muy parecidas a este último ejemplo.

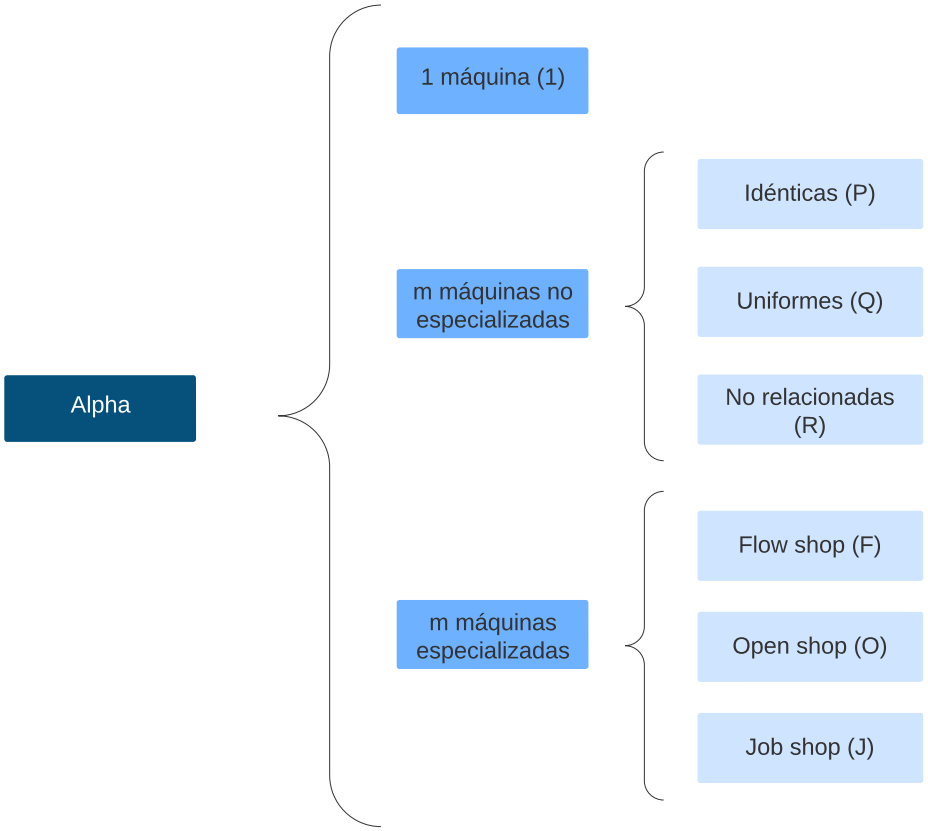

Figura 1.1. Tipología básica de las máquinas.

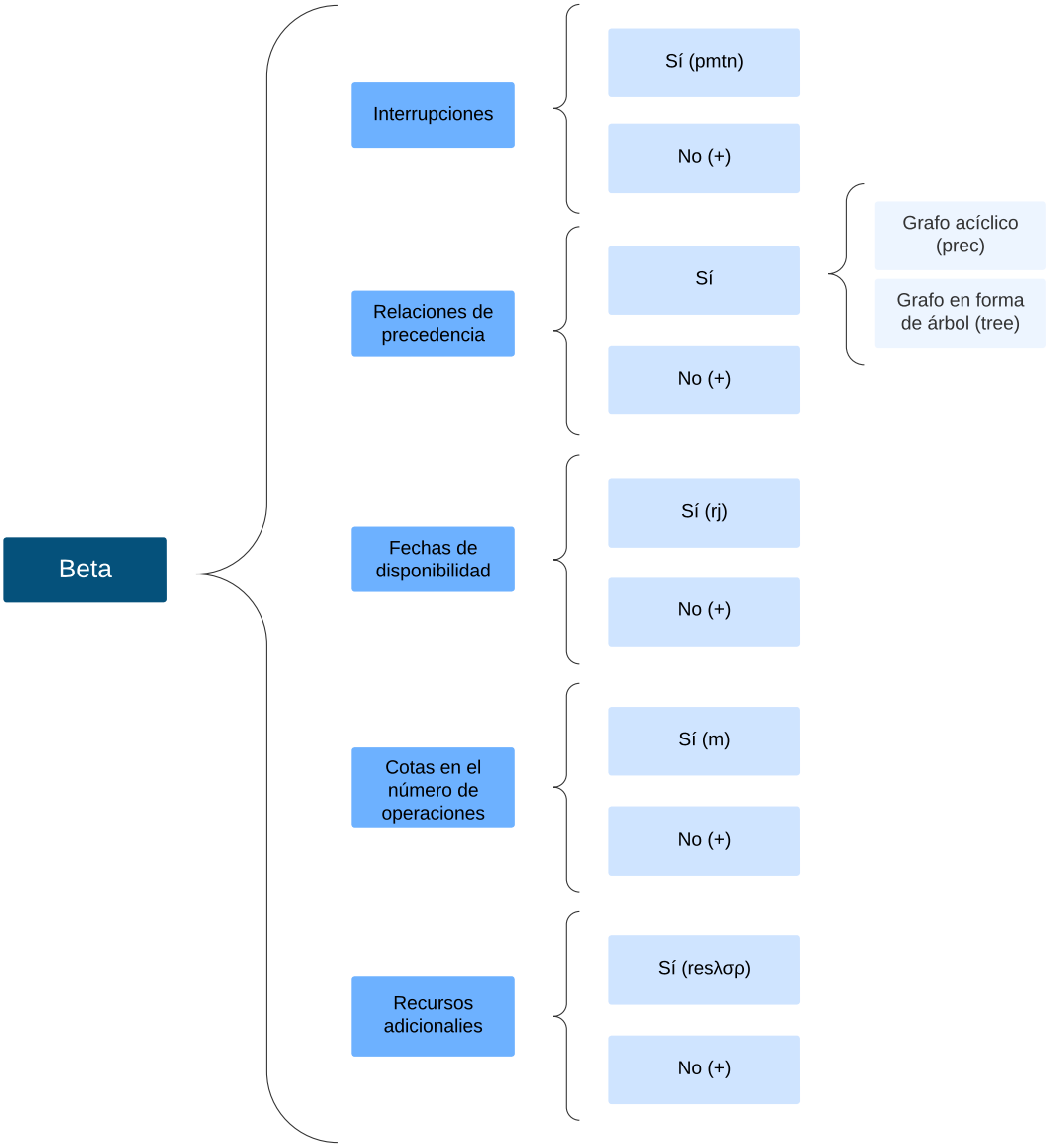

Figura 1.2. Algunas características de los trabajos.

2

Estudio detallado del HSSOP

En este capítulo, se desarrolla el HSSOP ("Hierarchical Stochastic Sequential Ordering Problem") o Problema de Secuenciación y Ordenación Estocástico Jerárquico. Trabajamos inicialmente con el caso determinístico del problema para posteriormente desarrollar la versión estocástica, con un criterio de factibilidad y otro para reducir el valor esperado del tiempo de completación del último trabajo de la secuencia. Además, se enuncian diversos resultados necesarios para los algoritmos de resolución del problema.

2.1. Motivación

Los problemas de Secuenciación y Ordenación o SOP ("Sequencing Ordering Problem") son una rama de los problemas de Planificación y Secuenciación que consisten en buscar la secuencia o el orden de ejecución de los diferentes trabajos en una única máquina. Aparecen en los procesos industriales o en los sistemas de manufacturación de fábricas y empresas. En general, son problemas que en la realidad surgen con frecuencia y están bastante estudiados y modelizados por su utilidad.

En muchas ocasiones, los problemas no se pueden resolver sin tener en cuenta la naturaleza aleatoria de las diversas variables. Por ejemplo, en los problemas donde se dan ventanas temporales, si se desconocen los tiempos de proceso de los trabajos, no se puede saber con presición si los trabajos de la secuencia se ajustan a las cotas, solo se puede enunciar en términos probabilísticos. Esto da pie a la confección de modelos estocásticos para la resolución de estos problemas.

En muchos problemas estocásticos con ventanas temporales en los trabajos, se suele no solo maximizar la probabilidad de factibilidad de la solución, sino también se tienen en cuenta otros criterios secundarios. De aquí surgen los problemas de Ordenación Secuencial Jerárquicos, que contemplan los diferentes criterios de optimalidad que estos problemas pueden tener.

Combinando las cualidades probabilísticas que pueden tener los datos y variables, junto a la jerarquía que puede darse en los criterios llegamos al problema que queremos resolver: El HSSOP ("Hierarchical Stochastic Sequential Ordering Problem") o Problema de Secuenciación y Ordenación Estocástico Jerárquico.

2.2. Planteamiento y Formulación del Modelo

Con objetivos de modelizar el problema, seguimos el mismo procedimiento desarrollado en la sección 2 del artículo Alcaide López de Pablo et al. (2003) [2], partiendo del problema clásico SOP determinístico, un problema de grafos comúnmente utilizado en el desarrollo y resolución del TSP ("Travelling Salesman Problem") o problema del Viajante de Comercio, y concluyendo con el problema a resolver y su posterior modelización.

2.2.1. Modelo determinístico

En este primer apartado desarrollamos la versión determinística del problema partiendo del, como se enunció anteriormente, problema determinístico SOP.

Sea un grafo dirigido $G = (V, A)$, donde V es el conjunto de vértices y A el de arcos, definimos un coste p_j asociado a cada vértice $j \in V$ y un coste c_a asociado a cada arco $a \in A$, siendo el objetivo encontrar el camino hamiltoniano S con coste mínimo, es decir, una secuencia de arcos que recorra cada uno de los vértices de V una única vez. Una forma de enunciar el objetivo del problema es la siguiente:

$$\text{mín}_S \left\{ \sum_{j \in J} p_j + \sum_{a \in S} c_a \right\}$$

Siendo S un camino hamiltoniano en el grafo $G = (V, A)$.

Además la solución debe estar sujeta a las siguientes restricciones:

- Cotas a los costes de los subcaminos. La suma de los costes desde el vértice inicial hasta el vértice j-ésimo (incluyendo en la suma los costes c_a correspondientes) debe estar acotado entre un valor mínimo y un máximo, $\forall j \in V$.

- Relaciones de precedencias entre vértices. La solución S del problema debe respetar las precedencias entre los trabajos, y las enunciamos mediante el grafo dirigido sin ciclos $R = (V, P)$, donde V son los vértices y P contiente los arcos con las relaciones de precedencia. Luego si $(i, j) \in P$, entonces el vértice i debe aparecer en la solución S antes que el vértice j.

Construyamos ahora un modelo de planificación para resolver este problema. El conjunto V contiene los trabajos a realizar, siendo p_j, $j = 1, 2, ..., n$ el tiempo de proceso del trabajo j, y el conjunto A contiene los pares de trabajos $a = (i, j)$, $i, j \in V$, que la máquina puede llevar a cabo de manera consecutiva, siendo c_a con $a = (i, j) \in A$ el tiempo que necesita la máquina en estar disponible para comenzar el trabajo j tras haber completado i, es decir, su tiempo de preparación. A fin de facilitar la modelización del problema, si $c_{ij} = c_{ji} = 0$, $i, j \in V$ entonces, decimos que los trabajos i y j pretenecen a la misma familia f, donde denotamos como f_i a la familia que contenga al trabajo i. Por último, la solución S de este problema es una secuencia de todos los trabajos de V, donde esta debe verificar las diferentes condiciones que presentamos en los siguientes párrafos.

Al igual que en el artículo [2], los trabajos j tienen sendos ventanas temporales, denotando por r_j como el instante de disponibilidad, que indica cuándo se puede empezar a procesar el trabajo j, $\forall j \in V$, y por otra parte, definimos d_j como las fechas límite, es decir, el tiempo máximo en el que se puede terminar el trabajo j sin que este sufra demoras, $\forall j \in V$. Con ambos parámetros definidos, tenemos que toda solución S del problema necesita que cada uno de sus trabajos se lleven a cabo dentro de sus correspondientes ventanas temporales $[r_j, d_j]$, $\forall j \in V$.

Consideramos también la existencia de relaciones de precedencia entre los trabajos. Las modelizamos con el diagrafo acíclico $R = (V, P)$, donde V son los trabajos del problema y el par $(i, j) \in P$ indica si el trabajo i se debe realizar antes que el j, y por tanto en la secuencia solución S debe aparecer i antes que j.

Con la finalidad de empezar a definir los objetivos del problema, introducimos el concepto de solución factible.

Definición 2.1. *Denotamos como Ω al conjunto de posibles secuencias S que cumplen las relaciones de precedencia descritas en $R = (V, P)$ y cada trabajo de S verifica los instantes de disponibilidad r_j, $\forall j \in V$.*

De este conjunto sale de manera intuitiva la idea de solución factible.

Definición 2.2. *Decimos que $S \in \Omega$ es una solución factible del problema si todos los trabajos se finalizan antes de sus fechas límites d_j, $\forall j \in S$.*

Otras variables que pueden ayudar al entendimiento y modelización del problema son los siguientes:

- Se denota por S_j la fecha de comienzo de j en la secuencia es decir, es el instante de tiempo donde da inicio el proceso del trabajo j en la máquina, $\forall j \in V$. Además, podemos denotar C_j como el tiempo de completación de j,

es decir, el momento donde la máquina finaliza de manera exitosa el trabajo j.

- F_j denota la cantidad de tiempo transcurrida desde el instante de disponibilidad (r_j) hasta el tiempo de completación (C_j), $\forall j \in V$.
- Se puede considerar también la demora de un trabajo j como $L_j = C_j - d_j$, que si es positiva, indica el tiempo extra que se ha usado para finalizar el trabajo; o se puede tener en cuenta solo la tardanza de un trabajo j que se define como $T_j = \text{máx}\{0, L_j\}$; o incluso utilizar un indicador de trabajo tardío U_j, valiendo 1 si $C_j > d_j$ (el tiempo de completación supera la fecha límite) y 0 en otro caso.

Los datos y variables asociadas al trabajo j que acabamos de comentar pueden ilustrarse gráficamente como muestra la figura 2.1, donde el eje de abcisas es el eje de tiempo.

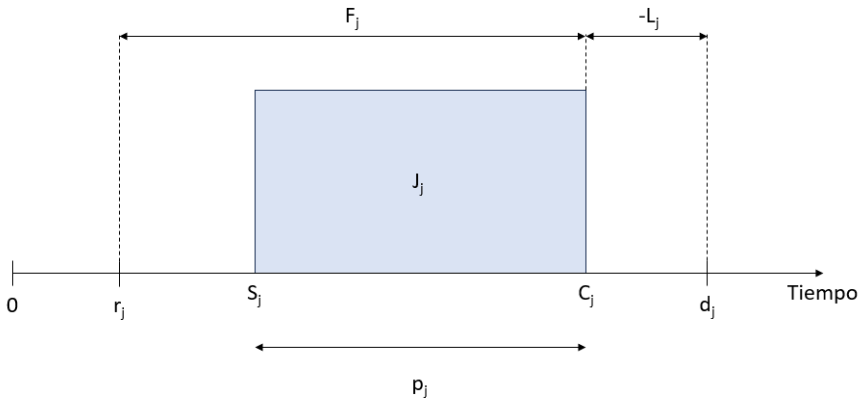

Figura 2.1. Trabajo con sus variables asociadas.

2.2.2. Modelo Estocástico

Los problemas estocásticos destacan principalmente por la exitencia de sucesos aleatorios que provocan el desconocimiento de algunos datos o variables dentro del problema. Como se muestra en la sección 2 del artículo [2], el factor aleatorio del problema estocástico puede localizarse en los tiempos de proceso p_j de cada trabajo j. Muchos problemas de Planificación y Secuenciación se moldelizan mejor con modelos estocásticos que con modelos determinísticos. Esto ocurre cuando algunos de los datos del problema no pueden ser representados como constantes conocidas o estimadas, sino que tienen un comportamiento

aleatorio o estocástico de acuerdo a alguna distribución de probabilidad. Este comportamiento aleatorio puede deberse a diversos factores como las averías en las máquinas, los imprevistos, factores humanos u otros motivos externos al problema.

Al asumir que los tiempos de proceso p_j son variables aleatorias, otras variables como los tiempos de completación de los trabajos C_j también son afectadas por este factor aleatorio, y por tanto, tenemos que trabajar en terminos probabilísticos para la resolución de nuestro problema. Dicho de otra manera, el carácter aleatorio de los datos de entrada hace que las funciones objetivo que modelizan los criterios de optimalidad también sean variables aleatorias.

Se pueden utilizar algunos conceptos (como el conjunto Ω) de la variante determinística de problema al intentar modelizar el Problema de Ordenación Secuencial Estocástico Jerárquico. Pero como desconocemos cuando terminan los trabajos, no se puede saber si una secuencia S es solución factible o no con exactitud. Esta dificultad da iniciativa a trabajar con la probabilidad de que una solución sea factible.

Definición 2.3. *Sea $S \in \Omega$ una secuencia de trabajos, la probabilidad de que la secuencia S sea una solución factible se denota por:*

$$P(S \text{ factible}) = P(C_1 \leq d_1, C_2 \leq d_2, ..., C_n \leq d_n)$$

Siendo n el número de trabajos que intervienen en el problema, d_j la fecha límite del trabajo j y C_j el tiempo de completación del trabajo j, $\forall j \in V$.

A partir de esta definición, enunciamos el primer criterio que consideramos para el HSSOP. Buscamos las soluciones con mayor probabilidad de ser factibles, es decir, aquellas secuencias que verifiquen las relaciones de precedencia, los instantes de disponibilidad r_j y cuya probabilidad de que los tiempos de completación de los trabajos no superen las fechas límite, sea máxima. Se puede denotar este objetivo de la siguiente forma:

$$\text{máx}\{P(S \text{ factible})/S \in \Omega\}$$

Como segundo criterio dentro de nuestro problema, buscamos entre las soluciones con un nivel de probabilidad de ser factibles suficientemente alto, aquellas en las que el tiempo de completación del último trabajo realizado sea menor. Es decir, en la jerarquía de criterios considerada, el primer criterio, y más importante es encontrar soluciones en el conjunto Ω de ordenaciones de trabajos cuya probabilidad de ser factibles (y, por tanto, respetar sus fechas límite d_j) sea suficientemente grande. Y entre dichas, encontrar aquellas en las que el valor esperado del tiempo de completación del último trabajo ($E[C_{\text{máx}}]$, con $C_{\text{máx}} = \text{máx}_{j=1,2,...,n}\{C_j\}$) sea menor. Esta segunda función objetivo se puede denotar de la siguiente forma:

$$\text{mín}\{E[C_{\text{máx}}(S)]/S \in \Omega \text{ y } P(S \text{ factible}) = P^*\}$$

Siendo P^* el máximo de la probabilidad de ser factible.

Cabe destacar que para trabajar con la factibilidad de S, siempre hay que garantizar que verifica las relaciones de precedencia y las fechas de disponibilidad r_j. Se puede definir la probabilidad de factibilidad de una secuencia que no verifique alguna de las condiciones anteriores como $P(S \text{ factible}) = 0$.

En muchas ocasiones el primer criterio es muy restrictivo, pues podemos descartar demasiadas secuencias al exigir que la probabilidad sea máxima. Esto puede afectar a la toma de decisiones en el segundo criterio de optimalidad. Una posible solución es establecer una cota o una tolerancia, donde la probabilidad de factibilidad sea lo suficientemente alta y que permita más variedad de soluciones para el otro criterio. Para ello, el artículo [2] introduce la definición siguiente:

Definición 2.4. *Una secuencia S que verifica las relaciones de precedencia decimos que es una secuencia factible de parámetro x (x-factible), con $0 \leq x \leq 1$, si $P(S \text{ factible}) \geq 1 - x$*

Con esto se puede fijar una nivel de tolerancia $\bar{x}$ cercana a cero y tomar todas las secuencias x-factibles, con $x \leq \bar{x}$ y minimizar $E[C_{\text{máx}}(S)]$.

2.3. Análisis del problema y resultados teóricos

2.3.1. Complejidad Computacional

En los próximos capítulos se citan y explican dos algoritmos para la resolución del HSSOP, uno exacto y uno heurístico provenientes de Alcaide López de Pablo et al. (2003) [2]. Una primera cuestión que el lector puede tener es: ¿Por qué necesitamos una heurística para este problema?

Si tuviesemos n trabajos que ordenar dentro del problema, el número de secuencias posibles equivale al número de permutaciones de los trabajos. En el peor de los casos tendremos que comparar $n!$ posibles soluciones e identificar la óptima. Para valores grandes de n, un algoritmo de busqueda tardará demasiado en encontrar la mejor solución. Esta dificultad da la iniciativa para utilizar una heurística.

En el siguiente teorema vemos que el HSSOP es un problema de dificultad máxima, demostrando las complicaciones para encontrar una solución óptima de este problema.

Teorema 2.5. *El problema HSSOP (Hierarchical Stochastic Sequential Ordering Problem) o Problema de Secuenciación y Ordenación Estocástico Jerárquico es NP-duro.*

Demostración. Vease el artículo [2], Sección 3.1, Teorema 1.

2.3.2. Resultados Teóricos

En esta sección se recogen algunos de los resultados teóricos que dan soporte a los algoritmos que estamos enunciando. Todos estos resultados y propiedades han sido obtenidas por Alcaide López de Pablo et al. (2003) [2]. En dicho artículo pueden verse también sus correspondientes demostraciones.

Factibilidad

Previo a iniciar con los teoremas y proposiciones, buscamos el modo de calcular la probabilidad de que una solución S sea factible, pues es necesaria para el primer criterio.

Sea una secuencia $S \in \Omega$, el tiempo de completación del trabajo en la posición j-ésima es:

$$C_1 = r_1 + p_1$$
$$C_j = \text{máx}\{r_j, C_{j-1} + c_{j-1,j}\} + p_j, \quad j \geq 2$$

Definimos la variable h_j, que indica el tiempo mínimo a partir del cual se puede empezar el trabajo de la posición j de la secuencia.

$$h_j = \text{máx}\left\{ \max_{1 \leq i \leq j-1} \{r_i + \sum_{k=1}^{j-1} c_{k,k+1}\}, r_j \right\}, \quad j = 1, 2, ..., n$$

Esta variable compara si se puede empezar el trabajo de la posición j en su instante de disponibilidad r_j o si hay que esperar a que terminen los trabajos en posiciones anteriores de la secuencia S para poder iniciarlo. Nótese que podemos escribir $h_1 = r_1$ y la formula anterior como:

$$h_j = \text{máx}\{r_j, h_{j-1} + c_{j-1,j}\}, \quad j \geq 2$$

Utilizando h_j, podemos calcular las funciones de distribución de los tiempos de completación C_j, pues las necesitamos para poder calcular la probabilidad de factibilidad y el valor esperado del tiempo de completación del último trabajo de la secuencia.

Proposición 2.6. *Dada una secuencia $S \in \Omega$, la función de distribución del tiempo de completación C_j del trabajo en la posición j-ésima en S es:*

$$F_{C_1}(t) = P(C_1 \leq t) = F_{p_1}(t - r_1), \quad t \geq r_1$$
$$F_{C_j}(t) = P(C_j \leq t) = \int_0^{t-h_j} F_{C_{j-1}}(t - c_{j-1,j} - y) f_{p_j}(y)\, \mathrm{d}y, \quad t > h_j,$$
para $j = 2, ..., n$.

Demostración. Véase el artículo [2], Sección 3.2, Proposición 1.

Ahora calculamos las probabilidades bivariantes $P(C_j \leq d_j,\, C_{j-1} \leq d_{j-1})$, para el trabajo en la posición j-ésima y el trabajo anterior, $j = 2, 3, ..., n$.

Proposición 2.7. *Dada una secuencia $S \in \Omega$, denotamos la probabilidad bivariante como $P(C_j \leq d_j, C_{j-1} \leq d_{j-1}) = p$, y para calcularla se pueden dar los siguientes casos:*

- *Caso 1: Si $d_{j-1} \geq d_j - c_{j-1,j}$, entonces*
  $p = F_{C_j}(d_j)$.

- *Caso 2: Si $r_j - c_{j-1,j} < d_{j-1} < d_j - c_{j-1,j}$, entonces*
  $p = F_{C_{j-1}}(d_{j-1})F_{p_j}(d_j - c_{j-1,j} - d_{j-1}) + F_{C_j}(d_j)$
  $- \displaystyle\int_0^{d_j - c_{j-1,j} - d_{j-1}} F_{C_{j-1}}(d_j - c_{j-1,j} - y) f_{p_j}(y)\,\mathrm{d}y.$

- *Caso 3: Si $d_{j-1} \leq r_j - c_{j-1,j}$ y $r_j > h_{j-1} + c_{j-1,j}$, entonces*
  $p = F_{C_{j-1}}(d_{j-1})F_{p_j}(d_j - r_j)$.

Demostración. Véase el artículo [2], Sección 3.2, Proposición 2.

Utilizando las probabilidades de las proposiciones anteriores, podemos calcular la probabilidad de que una secuencia S sea una solución factible del problema.

Teorema 2.8. *Sea una secuencia $S \in \Omega$, la probabilidad de que sea una solución factible es:*

$$P(S \text{ factible}) = P(C_1 \leq d_1), \quad n = 1.$$
$$P(S \text{ factible}) = P(C_1 \leq d_1, C_2 \leq d_2), \quad n = 2.$$
$$P(S \text{ factible}) = P(C_1 \leq d_1, C_2 \leq d_2) \times \prod_{j=3}^{n} \frac{P(C_{j-1} \leq d_{j-1}, C_j \leq d_j)}{P(C_{j-1} \leq d_{j-1})}, \quad n \geq 3.$$

Demostración. Véase el artículo [2], Sección 3.2, Teorema 2.

Esperanza del tiempo de completación del último trabajo de la secuencia

Ahora vemos como calcular el valor esperado del tiempo de completación del último trabajo de la secuencia. Para ello, utilizando las proposiciones anteriores se puede enunciar el siguiente teorema.

Teorema 2.9. *Sea una secuencia $S \in \Omega$, la esperanza de la variable C_j que modeliza el tiempo de completación del trabajo en la posición j en S es:*

$$E[C_1] = r_1 + E[p_1], \quad j = 1.$$
$$E[C_j] = h_j P(C_{j-1} \leq h_j - c_{j-1,j})$$
$$+ \int_{h_j - c_{j-1,j}}^{\infty} (t + c_{j-1,j}) f_{C_{j-1}}(t)\, \mathrm{d}t + E[p_j], \quad j \geq 2.$$

Demostración. Véase el artículo [2], Sección 3.3, Teorema 3.

Nótese que en un problema de n trabajos, el valor esperado del tiempo de completación del último trabajo es tiempo del completación del trabajo en la n-ésima posición, es decir, $C_{\text{máx}} = C_n$ (considerando n como el último trabajo de la secuencia sin perdida de generalidad), y por tanto, utilizando el teorema anterior:

$$E[C_{\text{máx}}] = h_n P(C_{n-1} \leq h_n - c_{n-1,n})$$
$$+ \int_{h_n - c_{n-1,n}}^{\infty} (t + c_{n-1,n}) f_{C_{n-1}}(t)\, \mathrm{d}t + E[p_n].$$

2.3.3. Aproximaciones

En esta sección se presentan algunas estimaciones a las probabilidades y esperanzas enunciadas en el apartado anterior. Al igual que las proposiciones, estos resultados provienen del artículo [2].

Factibilidad

Las integrales descritas en las proposiciones anteriores se vuelven más complejas a medida que las calculamos. Por ejemplo, en la primera proposición se puede observar que para obtener la función de distribución de C_j, es necesario haber calculado antes la de C_{j-1}. Esto implica que, para valores de j elevados, la función adquiere una expresión demasiado compleja como para poder ser evaluada en un corto periodo de tiempo, y esto dificulta a los algoritmos cuando van a calcular la probabilidad de que la secuencia S sea factible. Por lo tanto, necesitamos otra forma de comparar la factibilidad entre las soluciones.

De la fórmula de factibilidad de una secuencia $S \in \Omega$ se puede deducir lo siguiente:

$$P(C_1 \leq d_1, C_2 \leq d_2, ..., C_n \leq d_n) = P(C_1 \leq d_1) \cdot P(C_2 \leq d_2/C_1 \leq d_1) \cdot ...$$
$$\cdot P(C_n \leq d_n/C_1 \leq d_1, C_2 \leq d_2, ..., C_{n-1} \leq d_{n-1}).$$

Nos queda como producto de probabilidades condicionadas, que suelen ser complejas de calcular conforme aumentamos el número total de trabajos n. Como indica el artículo Alcaide López de Pablo et al. (2003) [2], a pesar de que los tiempos de completación C_j son variables aleatorias dependientes, en la práctica es razonable pensar que si la probabilidad marginal $P(C_j \leq d_j)$ es muy pequeña, entonces se puede esperar que la condicionada también lo sea. Concluimos así que otro criterio que se puede utilizar para intentar encontrar la solución óptima es tomar el mínimo de las probabilidades marginales:

$\text{mín}_{1 \leq j \leq n} P(C_j \leq d_j)$,

Donde j es la posición de los trabajos en la secuencia solución S.

Otra dificultad que puede surgir es el cálculo de las probabilidades marginales. A pesar de utilizar el otro criterio del mínimo para evitar calcular las probabilidades bivariantes, sigue estando el problema de que las fórmulas de las $P(C_j \leq d_j)$ se vuelven muy complejas y difíciles de computar cuando tratamos con muchos trabajos. Por tanto, otra estrategia que se puede usar es intentar acotar las probabilidades y utilizar estimaciones.

Hemos visto anteriormente que las variables C_j se pueden escribir de la siguiente forma:

$$C_1 = r_1 + p_1$$
$$C_j = \text{máx}\{r_j, C_{j-1} + c_{j-1,j}\} + p_j, \quad j \geq 2$$

Luego de manera generalizada las podemos denotar como

$$C_j = \text{máx}\left\{ r_j + p_j, r_{j-1} + c_{j-1,j} + p_j, ..., r_1 + \sum_{i=1}^{j} p_i + \sum_{i=1}^{j-1} c_{i,i+1} \right\},$$
$$j = 1, 2, ..., n.$$

Entonces, se pueden acotar los tiempos de completación del trabajo en la posición j con las siguientes cotas:

$$UB(C_j) = \max_{i=1,...,j} \{r_i\} + \sum_{i=1}^{j} p_i + \sum_{i=1}^{j-1} c_{i,i+1}$$

$$LB(C_j) = r_1 + \sum_{i=1}^{j} p_i + \sum_{i=1}^{j-1} c_{i,i+1}$$

Donde UB ("Upper Bound") es la cota superior y LB ("Lower Bound") es la cota inferior, que verifican que $LB(C_j) \leq C_j \leq UB(C_j)$, y por tanto, $P(LB(C_j) \leq d_j) \leq P(C_j \leq d_j) \leq P(UB(C_j) \leq d_j)$, $\forall j = 1, ..., n$.

Estas probabilidades solo dependen de la variable aleatoria $\sum_{i=1}^{j} p_i$ que suelen ser mejor computables que las distribuciones de C_j cuando j es grande.

Gracias a estas cotas podemos estimar la probabilidad marginal $P(C_j \leq d_j)$ como:

$$\hat{P}C_j \leq d_j) = w_{1j}P(UB(C_j) \leq d_j) + w_{2j}P(LB(C_j) \leq d_j),$$
$$\text{con } w_{1j},\ w_{2j} \geq 0,\ w_{1j} + w_{2j} = 1,\ j = 1, ..., n.$$

En el capítulo 4, se confirma que los intervalos que generan las cotas son muy pequeños, haciendo útiles las estimaciones.

Esperanza del tiempo de completación del último trabajo de la secuencia

Con objetivos de también intentar acotar $E[C_{\text{máx}}(S)]$, partimos de las desigualdades anteriores y las tomamos para $C_j = C_n$ (el trabajo de S en la última posición), es decir, para $C_{\text{máx}}$.

$$LB(C_n) = LB(C_{\text{máx}}) \leq C_{\text{máx}} \leq UB(C_{\text{máx}}) = UB(C_n).$$

Por tanto, aplicando la esperanza en la desigualdad llegamos a que

$$LB_1 = E[LB(C_{\text{máx}})] \leq E[C_{\text{máx}}] \leq E[UB(C_{\text{máx}})] = UB_1$$

Utilizando propiedades de la esperanza, la cota inferior LB_1 se puede mejorar, llegando a la siguiente expresión:

$$LB_2 = \text{máx}\{r_n + E[p_n], r_{n-1} + E[p_{n-1}] + c_{n-1,n} + E[p_n], ...,$$
$$r1 + \sum_{i=1}^{n} E[p_i] + \sum_{i=1}^{n-1} c_{i,i+1}\},$$

Con $LB_1 \leq LB_2 \leq E[C_{\text{máx}}] \leq UB_1$.

Y se concluye con las cotas $LB_2 \leq E[C_{\text{máx}}] \leq UB_1$. Siguiendo el mismo razonamiento que con las probabilidades marginales constuimos una estimación de la misma forma:

$$\hat{E}[C_{\text{máx}}] = w_1' LB_2 + w_2' UB_1, \text{ con } w_1',\ w_2' \geq 0,\ w_1' + w_2' = 1.$$

Al igual que en el apartado anterior, en el capítulo de aplicaciones revisamos que estos intervalos formados por las cotas son pequeños haciendo buenas las estimaciones.

3

Algunas Estrategias de resolución del problema

En este capítulo se presentan dos algoritmos para la resolución del problema de Secuenciación y Ordenación Estocástico Jerárquico. Estos provienen del artículo [2], Secciones 4 y 5.

3.1. Algoritmo exacto

Este primer algoritmo es un procedimiento exhaustivo, que pretende analizar todas las posibles secuencias del conjunto Ω, calculando primero su probabilidad de ser factible mediante los teoremas y las integrales del capítulo anterior, tomando aquellas soluciones con probabilidad máxima. Luego, se elige de entre ellas, aquella cuyo valor esperado del tiempo de completación del último trabajo de la secuencia sea mínimo.

El algoritmo tiene una primera fase de inicialización en la que se define U que contiene las soluciones no analizadas ("Untested"), Σ como el conjunto de soluciones con probabilidad máxima de ser factibles, y se toma una secuencia inicial y su probabilidad de ser factible.

Se prosigue con una fase de factibilidad en la que analizan mediante un bucle "while" todas las soluciones. Si encuentra otra solución con misma probabilidad de ser factible, esta se añade a Σ, si tiene mayor probabilidad, se reinicia el valor de Σ y se le añade esta solución, y en otro caso se descarta.

Luego de analizar todo Ω, realizamos un proceso parecido para el segundo criterio. Definimos M como el conjunto de soluciones con esperanza mínima, se toma una secuencia inicial con su esperanza inicial y se realiza un bucle idéntico a la fase de factibilidad, pero en lugar de comparar la probabilidad de que la solución sea factible, se utiliza valor esperado del tiempo de completación del último trabajo.

El algoritmo concluye con el conjunto de soluciones óptimas, con probabilidad máxima y esperanza mínima. De este se puede elegir cualquiera de las soluciones para aplicarla en la planificación del problema.

El pseudocódigo del algoritmo exacto es el siguiente:

Procedimiento Exacto
/ Fase de Inicialización */*
$U = \Omega$;
Tomamos una solución inicial $S_1 \in U$;
$P^* = P(S_1$ factible);
$\Sigma = \{S_1\}$;
$U = U - \{S_1\}$;
/ Fase de Factibilidad */*
while $U \neq \emptyset\{$
Elegimos una secuencia $S \in U$;
if $P(S$ factible$) > P^*\{$
$P^* = P(S$ factible$)$;
$\Sigma = \{S\}$;
$\}$
else if $P(S$ factible$) = P^*\{$
$\Sigma = \Sigma \cup \{S\}$;
$U = U - \{S\}$;
$\}$ */* Fin del while */*
/ Fase de Esperanza del tiempo de completación del último trabajo */*
$U = \Sigma$
Tomamos una solución inicial $S_2 \in U$;
$E^* = E[C_{\text{máx}}(S_2)]$;
$M = \{S_2\}$;
$U = U - \{S_2\}$;
while $U \neq \emptyset\{$
Elegimos una secuencia $S \in U$;
if $E[C_{\text{máx}}(S)] > E^*\{$
$E^* = E[C_{\text{máx}}(S_2)]$;
$M = \{S\}$;
$\}$
else if $E[C_{\text{máx}}(S)] = E^*\{$
$M = M \cup \{S\}$;
$U = U - \{S\}$;
$\}$ */* Fin del while */*
/ Fin del algoritmo */*

3.2. Procedimiento heurístico

En esta sección presentaremos la heurística diseñada por Alcaide López de Pablo et al. (2003) [2] para encontrar una solución aproximada del problema. Esta consta de una fase de preproceso, dos fases en la que se intenta mejorar la probabilidad de ser factible y una última centrada en el valor esperado del tiempo de completación del último trabajo de la secuencia, donde en cada uno de los módulos se intenta alterar el orden de los trabajos de la solución S para mejorarla.

Un pseudocódigo de este procedimiento es el siguiente:

Algoritmo Heurístico
/ Preproceso */*
Actualización de las ventanas de tiempo;
/ Fase 1 */*
Tolerancia $\overline{\alpha}$;
Solución inicial EDD S;
Probabilidad de ser factible inicial α;
while $\alpha > \overline{\alpha}${
Módulo de reordenación;
if Encontramos solución factible de parámetro α
Reducimos α
else
Módulo de intercambio;
if Encontramos solución factible de parámetro α
Reducimos α
}
/ Fase 2 */*
Reducir tiempo ocioso;
/ Fase 3 */*
Minimizar Esperanza del tiempo de completación del último trabajo;
/ Fin del algoritmo */*

En las siguientes secciones se explica el funcionamiento de los diferentes módulos del algoritmo heurístico.

3.2.1. Preproceso

Se inicia con una fase preliminar que tiene el objetivo de reducir las ventanas de tiempo utilizando los instantes de disponibilidad r_j, las fechas límite d_j y las relaciones de precedencia $R = (V, P)$.

Sea un trabajo j, que tiene un conjunto de trabajos predecesores definidos mediante las relaciones de precedencia en R, encontramos el instante de tiempo θ_j tal que si empezamos el trabajo j antes, alguno de sus predecesores no está completo antes de su fecha de entrega. Luego se actualizan los instantes de disponibilidad del trabajo j tomando el máximo entre r_j y θ_j, es decir, $r'_j = \text{máx}\{r_j, \theta_j\}$.

Sea un trabajo i, que tiene un conjunto de trabajos sucesores definidos mediante las relaciones de precedencia en R, encontramos el instante de tiempo δ_j tal que si empezamos el trabajo j después, alguno de sus sucesores no está completo antes de su fecha de entrega. Luego se actualizan las fechas límite del trabajo j tomando el mínimo entre d_j y δ_j, es decir, $d'_j = \text{mín}\{d_j, \delta_j\}$.

Después de este ajuste, el algoritmo toma como solución inicial aquella que sigue la regla EDD ("Earliest Due Date"), es decir, se ordenan los trabajos de menor a mayor en función se las nuevas fechas de entrega d'_j. Nótese que a pesar de haber cambiado el orden de la solución, las relaciones de precedencia se verifican gracias al preproceso anterior.

3.2.2. Fase 1: Módulos de reordenación e intercambio

En esta parte del algortimo, definimos una probabilidad de factibilidad inicial $\overline{\alpha}$, cuyo valor dependerá del problema de planificación y de la tolerancia mínima que exijamos a la solución para considerarla factible. Se inicia un proceso iterativo en el que se busca adelantar o permutar los trabajos de la solución hasta satisfacer este valor mínimo o hasta que ambos procesos no sean capaces de mejorar la solución.

Módulo de reordenación

Antes de iniciar el módulo, revisamos la probabilidad de que la solución inicial sea factible. Si esta ya satisface el mínimo $\overline{\alpha}$, iremos directamente a la Fase 2, de lo contrario se inicializa el bucle de mejora de la solución. Sea $1-\alpha$ la probabilidad de que la solución actual sea factible, vamos a examinar la secuencia S en busca del primer trabajo cuya probabilidad de ser factible no alcance el valor $1-\alpha$, es decir, el primer trabajo de S tal que $P(C_j \leq d_j) < 1-\alpha$. Si no encontramos ningún trabajo, reducimos el valor de α una cantidad ϵ y buscamos de nuevo el primer trabajo no factible a esa probabilidad. Se realiza

esta reducción hasta que α sea lo suficientemente pequeño, es decir, cuando lleguemos a la tolerancia $1 - \overline{\alpha}$, y por tanto, iremos a la Fase 2.

Si encontramos el trabajo j cuya probabilidad no alcanza el valor $1 - \alpha$, intentamos realizarlo antes sin que los trabajos afectados por este cambio dejen de ser factibles con probabilidad $1 - \alpha$. Consideramos la subsecuencia de S que comienza en el trabajo inicial de S, que denotamos como k, y termina en el trabajo anterior a j, que denotamos como $y(j)$, y buscamos un trabajo i en esta subsecuencia tal que podemos poner j justo después de i, verificando que los trabajos de esta nueva subsecuencia son factibles con probabilidad $1 - \alpha$. Si existen varios candidatos i, buscaremos aquel que maximice el mínimo de las probabilidades $P(C_l \leq d_l)$ con l en la subsecuencia. Si no encontramos ningún trabajo i, fracasamos al mejorar la solución y procedemos con el módulo de intercambio. Si conseguimos anteponer j detrás de i, hemos tenido éxito en mejorar la factibilidad de la solución, y por tanto, pasamos al siguiente módulo. Es importante remarcar si se ha alterado o no la secuencia, pues no queremos entrar en bucle si ninguno de los dos módulos de la fase 1 son capaces de mejorar la solución.

Módulo de intercambio

En esta sección del algoritmo, se busca permutar pares de trabajos adyacentes, tal que la suma de los tiempos de preparación c_a decrezca. Vimos en secciones anteriores que f_j denota la familia del trabajo j, es decir, contiene todos los trabajos i cuyos tiempos de preparación c_{ij} y c_{ji} son cero. Buscaremos las tuplas de 4 trabajos consecutivos (i, j, k, l) dentro de la secuencia S tales que $f_i \neq f_j \neq f_k \neq f_l$, y queremos permutar los trabajos k y j siempre que: el cambio sea compatible con las relaciones de precedencia, la suma de los tiempos de preparación decrezca y la factibilidad tras el intercambio mejore.

Si conseguimos realizar el intercambio, vemos la factibilidad de la solución y en función de si alcanza o no el nivel de factibilidad $1 - \overline{\alpha}$ proseguiremos con la Fase 2 o volvemos a ejecutar módulo de reordenación. Si no llevamos a cabo el intercambio, fracasamos al intentar aplicar este módulo.

Nótese que si en cualquier momento, fracasamos al intentar aplicar los dos módulos de manera consecutiva, esto indica que la solución no se puede mejorar más, y por tanto iremos a la Fase 2 directamente.

3.2.3. Fase 2: Reducción del tiempo ocioso

Tras aplicar los módulos anteriores, la solución obtenida puede presentar tiempo ocioso entre los trabajos, es decir, pueden existir periodos de tiempo entre las ejecuciones de los trabajos, en los que la máquina no está llevando

a cabo niguna actividad. La idea fundamental de este módulo es, utilizando una estrategia parecida a la del módulo de reordenación, alterar el orden de los trabajos en la secuencia de tal forma que reducimos la probabilidad de que existan tiempos ociosos, mejorando así la probabilidad de que la solución sea factible.

Definimos I_i como la variable aleatoria que determina la cantidad de tiempo ocioso que hay entre los trabajos en la posición $i-1$ y posición i en la secuencia S.

$$I_i = \text{máx}\{0, r_i - (C_{i-1} + c_{i-1,i})\}$$

Elegimos los trabajos j de mayor a menor en función de los instantes de disponibilidad r_j y vamos a intentar anticipar el trabajo j a la posición i, elegida de mayor a menor en función de la probabilidad $P(I_i > 0)$ (que indica la probabilidad de que exista tiempo ocioso entre los trabajos en las posiciones $i-1$ e i).

Para estimar la probabilidad, utilizaremos el mismo método que en el artículo [2], que se desarrolla de la siguiente forma: $P(I_i > 0) = P(C_{i-1} < r_i - c_{i-1,i})$, teniendo en cuenta que $LB(C_{i-1}) \leq C_{i-1} \leq UB(C_{i-1})$ entonces $P(UB(C_{i-1}) < r_i - c_{i-1,i}) \leq P(C_{i-1} < r_i - c_{i-1,i}) \leq P(LB(C_{i-1}) < r_i - c_{i-1,i})$. Denotando por $P(UB(I_i) > 0) = P(UB(C_{i-1}) < r_i - c_{i-1,i})$ y por $P(LB(I_i) > 0) = P(LB(C_{i-1}) < r_i - c_{i-1,i})$, podemos acotar $P(I_i > 0)$ con $P(UB(I_i) > 0) \leq P(I_i > 0) \leq P(LB(I_i) > 0)$ y se puede ralizar la estimación:

$$\hat{P}(I_i > 0) = w_1 P(UB(I_i) > 0) + w_2 P(LB(I_i) > 0)$$

Con $w_1, w_2 \leq 0, w_1 + w_2 = 1$.

Al igual que en los anteriores módulos si existiesen varias posiciones donde el trabajo pudiese ser adelantado, eligiremos aquella que produzca un mayor incremento en la probabilidad de factibilidad de la secuencia.

3.2.4. Fase 3: Minimizar el valor esperado del tiempo de completación del último trabajo de la secuencia

Tras las mejoras anteriores de la factibilidad de la solución, en esta última fase nos centramos en reducir $E[C_{\text{máx}}]$. Para ello intentamos anteponer los trabajos j, elegidos de mayor a menor en función del tamaño de su ventana temporal $(d_j - r_j)$, a la posición i, elegida, al igual que en la Fase 2, de mayor a menor en función de las probabilidades $P(I_i > 0)$. Nótese que cada vez que llevemos a cabo un cambio en la secuencia, debemos comprobar que la probabilidad de ser factible alcanzada en los módulos anteriores no decrece.

Con esta última fase creamos nuestra solución heurística sin tener que revisar cada una de las posibles combinaciones de los trabajos.

Todas las veces que hablamos de factibilidad en los procedimientos anteriores, puede referirse tanto a la probabilidad exacta de que la secuencia sea factible, como al criterio del mínimo de las probabilidades marginales. Además, cuando calculamos las probabilidades marginales, se pueden hacer mediante las fórmulas originales o utilizando las estimaciones con las cotas. Por último, el valor esperado del tiempo de completación del último trabajo de la secuencia se puede también calcular con la ecuación exacta o con la estimación vista en la sección 2.3.2.

En el capítulo siguiente comparamos tanto los algoritmos como las estimaciones para verificar su validez y utilidad.

4

Aplicación de un Modelo para resolver el HSSOP

En la sección siguiente se desarrolla una experiencia computacional para el HSSOP, donde se computan y comparan los algoritmos presentados en el capítulo anterior.

Se pretende complementar y actualizar a nuevos lenguajes de programación la experiencia computacional desarrollada en el artículo [2] con una posible perspectiva de desarrollar este mismo modelo para otros criterios como minimizar $E[T_{\text{máx}}]$. En aquella ocasión, el autor trabajó con el lenguaje C y ahora se ha utilizado la versión 3.8 de Python para la creación del código.

4.1. Generación de parámetros de entrada para contrastar las estrategias de resolución

Es importante destacar que los algoritmos presentados no dependen de las distribuciones de las variables aleatorias. En nuestro caso hemos considerado los tiempos de proceso p_j como variables independientes e idénticamente distribuidas como una exponencial de parámetro λ. Se han planteado problemas con $n = 5$, 10, 20, 50, y 100 trabajos a secuenciar, y para cada caso se han generado 20 pruebas que varían en función de las fechas límite d_j. El resto de datos han sido generados de manera aleatoria utilizando la librería random de Python siguiendo los siguientes criterios:

- Para cada n, se ha generado $\frac{1}{\lambda}$ siguiendo una distribución uniforme $U[0, 100]$.
- Los instantes de disponibilidad y los tiempos de preparación se emularon utilizando las siguientes distribuciones: $r_j \sim U[0, 50]$ y $c_{i,j} \sim U[0, 20]$, cuando $i \neq j$, siendo $c_{ii} = 0$.
- Para las fechas límite d_j, se han utilizado varios recursos para intentar generar 20 pruebas diferentes. Consideramos su distribución como

  $d_j \sim U[r_j + L_1, r_j + L_2]$

donde:

$L_1 = n\frac{1}{\lambda}(2 - TF - \frac{RDD}{2})$.

$L_2 = n\frac{1}{\lambda}(2 - TF + \frac{RDD}{2})$.

Siendo TF ("Tardiness Factor") un valor que modeliza cuando un trabajo pueda ser tardío o no, y RDD ("Range Due Date") un dato que amplía o disminuye el rango donde puede darse la fecha de entrega del trabajo. TF toma los valores $\{0, 0.2, 0.4, 0.6, 0.8\}$ y RDD varía entre los valores $\{0.2, 0.4, 0.6, 0.8\}$, siendo por tanto, 20 pruebas diferentes las que se pueden hacer para cada valor posible de TF y RDD.

Los parámetros de entrada se generan de esta forma pues se quiere probar el funcionamiento y la precisión de los algoritmos, para un rango amplio de casos posibles. Gracias a la variabilidad del dato d_j, podemos verificar la eficacia de los procedimientos.

Al igual que en el artículo [2], no hemos considerado relaciones de precedencia en la aplicación de los algoritmos del modelo.

Para realizar las comparaciones de los algoritmos, se ha llevado a cabo un código en el editor de texto VSCode, generando una serie de tablas con las que podemos comparar los tiempos de ejecución y la precisión de la heurística.

4.2. Tablas de resultados

En esta sección se presentan una serie de tablas en las que se comparan los algoritmos en función de la factibilidad, esperanza y tiempos de proceso finales.

La tabla 4.1 representa las 20 pruebas para $n = 5$ trabajos y está dividida en dos secciones, una que presenta los resultados del algoritmo exacto y otra los del heurístico. En las columnas del algoritmo exacto, aparece una columna con una de las soluciones generadas por el algoritmo (Sol) (tomamos en general la solución en la primera posición), tenemos otra columna con la probabilidad óptima de que la secuencia sea factible (Factibilidad), la columna del valor esperado del tiempo de completación del último trabajo de la secuencia del problema (Esperanza) y el tiempo en segundos de ejecución (Tiempo) para calcular el conjunto de soluciones óptimas. Por otra parte, en las columnas del algoritmo heurístico, aparecen las mismas que en el exacto (Nótese que la columna de soluciones en este caso representa la única solución calculada por el procedimiento heurístico), añadiendo una columna de mejoras. Diremos que se da una mejora cuando alguno de los módulos de reordenación, intercambio, de reducción de tiempo ocioso o de minimización del valor esperado del tiempo

	Exacto					Heurística				
Pruebas	Sol	Factibilidad	Esperanza	Tiempo		Sol	Mejoras	Factibilidad	Esperanza	Tiempo
0	(1, 4, 2, 3, 0)	0.97446386	373.583137	2097.168151		(1, 3, 2, 4, 0)	0	0.974270503	380.7189006	231.4087829
1	(2, 3, 4, 0, 1)	0.964846204	150.657694	1114.479244	*	(2, 3, 4, 0, 1)	1	0.964846204	150.6576939	168.3954369
2	(2, 0, 3, 1, 4)	0.976293412	201.295099	1878.357156	*	(2, 0, 3, 1, 4)	0	0.976293412	201.2950987	365.5281052
3	(1, 0, 2, 3, 4)	0.983849265	483.638507	1857.986266		(1, 0, 2, 4, 3)	2	0.983681799	514.7004994	260.7979626
4	(1, 2, 0, 4, 3)	0.957144674	439.261154	1077.189381		(2, 0, 1, 3, 4)	2	0.946993455	461.453539	122.5678139
5	(1, 2, 0, 4, 3)	0.922631587	634.689732	557.8293917	*	(1, 2, 0, 4, 3)	1	0.922631587	634.6897322	137.3160519
6	(3, 0, 2, 1, 4)	0.972654017	456.976319	1025.366712		(3, 1, 0, 2, 4)	1	0.971669303	464.8645576	167.9860639
7	(4, 0, 3, 2, 1)	0.978978457	126.216921	1166.83824	*	(4, 0 3, 2, 1)	1	0.978978457	126.2169208	120.2606405
8	(4, 0, 2, 1, 3)	0.901240896	373.523451	1033.754698		(3, 0, 1, 4, 2)	2	0.896970392	378.4597537	175.9681162
9	(1, 0, 2, 3, 4)	0.917461356	284.546476	1085.55348		(2, 3, 0, 1, 4)	3	0.892282878	299.7825085	173.7066715
10	(2, 1, 3, 0, 4)	0.917214144	532.624747	1162.454343	*	(2, 1, 3, 0, 4)	1	0.917214144	532.6247467	220.488906
11	(3, 4, 0, 1, 2)	0.943699326	334.414886	1147.674163		(4, 0, 3, 1, 2)	0	0.923700405	363.7975624	205.6211383
12	(2, 4, 0, 1, 3)	0.87758853	326.760650	1048.504652	*	(2, 4, 0, 1, 3)	1	0.87758853	326.760650	124.7216004
13	(1, 0, 2, 4, 3)	0.87226587	184.380272	851.624739		(0, 1, 2, 4, 3)	0	0.869952621	185.2946925	143.2870687
14	(0, 1, 4, 3, 2)	0.871736559	978.48134	920.6139851	*	(0, 1, 4, 3, 2)	2	0.871736559	978.4813367	227.8135597
15	(2, 4, 3, 1, 0)	0.901501814	361.135610	1265.37899		(2, 4, 3, 0, 1)	0	0.898621207	377.2459149	221.5244943
16	(3, 1, 0, 2, 4)	0.810674507	676.616832	862.219891	*	(3, 1, 0, 2, 4)	0	0.810674507	676.616832	162.7877643
17	(0, 4, 1, 2, 3)	0.683307223	257.036206	1111.986374		(4, 0, 1, 2, 3)	1	0.673501975	260.0499379	183.8314696
18	(1, 0, 4, 2, 3)	0.805913251	506.09675	1765.159255	*	(1, 0, 4, 2, 3)	3	0.805913251	506.0967456	372.9573962
19	(2, 4, 0, 3, 1)	0.865778165	451.637429	1403.481297		(2, 3, 4, 0, 1)	1	0.863844656	468.1279857	252.5952907

Tabla 4.1. $n = 5$ trabajos con $P(S$ factible$)$ y $E[C_{\text{máx}}]$ como criterios.

de completación del último trabajo alteran el orden de los trabajos de la solución para mejorarla. Entre ambas secciones de los algoritmos, hemos añadido una columna extra con asteriscos, donde marcamos las pruebas en las que las soluciones de ambos procedimientos era la misma.

Las probabilidades de ser factible y los valores esperados del tiempo de completación del último trabajo de las secuencias de cada prueba se calcularon en ambos algoritmos usando las fórmulas exactas. En la tabla se puede comprobar que hay muchas coincidencias entre el algoritmo exacto y el heurístico, y muchas de las pruebas que no coinciden se puede ver que las diferencias no son excesivas. En el peor de los casos, en las pruebas 9 y 11 la probabilidad de ser factible difiere por aproximadamente 0.02 y en la columna de la esperanza el error no excede de 50 en la mayoría de casos. Esto demuestra de manera empírica que el procedimiento heurístico puede dar soluciones bastante buenas en mucho menos tiempo.

En la tabla 4.2, al igual que la anterior, presenta las 20 pruebas para $n = 5$ con la única diferencia de que en lugar de usar el criterio de la formula exacta para calcular la probabilidad de ser factible, se utiliza el mínimo de las probabilidades marginales para ambos algoritmos. Gracias a este cambio podemos comparar ambos algoritmos utilizando este criterio, y además, dado que los 20 problemas son los mismos que los generados en la tabla anterior, podemos comparar los criterios entre ellos para ambos algoritmos.

Vemos una reducción clara de los tiempos de ejecución en ambos procedimientos. En muchas ocasiones se tiende a llegar al óptimo, aunque hay casos como la prueba 4 en el que las soluciones exactas son diferentes. Esto puede

	Exacto					Heurística				
Pruebas	Sol	Factibilidad	Esperanza	Tiempo		Sol	Mejoras	Factibilidad	Esperanza	Tiempo
0	(1, 4, 2, 3, 0)	0.975687033	373.5831366	154.2740942		(1, 3, 2, 4, 0)	0	0.974302799	380.7189006	34.65129
1	(2, 3, 4, 0, 1)	0.965278893	150.6576939	155.7892435	*	(2, 3, 4, 0, 1)	1	0.965278893	150.6576939	35.41235
2	(2, 0, 3, 1, 4)	0.979838392	201.2950987	163.6414166	*	(2, 0, 3, 1, 4)	0	0.979838392	201.2950987	36.79261
3	(1, 0, 2, 3, 4)	0.987387957	483.6385074	187.3798039	*	(1, 0, 2, 3, 4)	1	0.987387957	483.6385074	57.33332
4	(1, 3, 0, 2, 4)	0.957502318	445.2464386	227.4106422		(2, 0, 1, 3, 4)	2	0.947391924	461.453539	62.03799
5	(1, 2, 0, 4, 3)	0.928866865	63.46897322	114.7257792	*	(1, 2, 0, 4, 3)	1	0.928866865	63.46897322	54.17810
6	(3, 0, 2, 1, 4)	0.976759447	456.9763187	239.3827021		(3, 1, 0, 2, 4)	1	0.975312029	464.8645576	68.33361
7	(4, 0, 3, 2, 1)	0.97970158	126.2169208	240.4430273	*	(4, 0, 3, 2, 1)	1	0.97970158	126.2169208	65.46079
8	(4, 0, 2, 1, 3)	0.907269433	373.523451	252.39468		(3, 0, 1, 4, 2)	2	0.89966898	378.4597537	75.19860
9	(1, 2, 4, 0, 3)	0.925834288	280.1349971	250.9304649		(4, 2, 0, 1, 3)	1	0.899672212	303.6768141	70.66533
10	(2, 1, 3, 0, 4)	0.936641977	532.6247467	252.4936753	*	(2, 1, 3, 0, 4)	1	0.936641977	532.6247467	59.17317
11	(0, 1, 4, 3, 2)	0.955400087	323.8969026	254.0465129		(4, 1, 0, 3, 2)	1	0.932918171	350.0330699	57.92227
12	(2, 4, 0, 1, 3)	0.879371675	326.76065	252.5544669	*	(2, 4, 0, 1, 3)	1	0.879371675	326.76065	69.43936
13	(1, 0, 2, 4, 3)	0.872986632	184.3802717	247.1202798		(0, 1, 2, 4, 3)	0	0.870688928	185.2946925	51.52292
14	(0, 1, 4, 3, 2)	0.877623015	97.84813367	228.0166845		(0, 1, 3, 4, 2)	4	0.770647885	113.8729042	75.82303
15	(0, 2, 4, 3, 1)	0.918910354	368.3166019	232.3171453		(2, 4, 3, 1, 0)	1	0.918530587	361.1356096	61.77858
16	(3, 1, 0, 2, 4)	0.824407183	67.6616832	23.5052392	*	(3, 1, 0, 2, 4)	0	0.824407183	67.6616832	54.51047
17	(0, 4, 1, 2, 3)	0.690052258	257.036206	296.7487309		(4, 0, 1, 2, 3)	1	0.679054761	260.0499379	77.26122
18	(1, 0, 4, 2, 3)	0.838716551	506.0967456	343.3520215		(1, 2, 4, 0, 3)	1	0.815064607	529.8891417	101.3571
19	(2, 4, 0, 3, 1)	0.896481979	451.6374289	355.2681496		(2, 3, 0, 4, 1)	2	0.886844556	463.7820303	95.24531

Tabla 4.2. $n = 5$ trabajos con $\min_{1 \leq j \leq n} P(C_j \leq d_j)$ y $E[C_{\max}]$ como criterios.

deberse al cambio de criterio o a que existan varias soluciones óptimas. A pesar de las diferencias, los valores de la probabilidad de factibilidad y de la esperanza del tiempo de completación del último trabajo no varían excesivamente al igual que en la tabla anterior.

Luego, verificamos que el procedimiento heurístico genera soluciones muy parecidas a las óptimas del algoritmo exhaustivo, siendo un método fiable para encontrar soluciones en el resto de tablas. Además, el cambio de criterio no afecta en exceso a la elección de la solución óptima.

En las tablas restantes del capítulo consideramos únicamente el algoritmo heurístico con el criterio del mínimo de las probabilidades marginales. Además, utilizamos las estimaciónes de dichas probabilidades y las de la esperanza del tiempo de completación del último trabajo de la secuencia. Las tablas van en orden creciente de trabajos como hemos enunciado anteriormente: 5, 10, 20, 50, y 100 trabajos respectivamente. En las tablas incluiremos dos columnas con las estimaciones del mínimo y la media de las probabilidades marginales, y otra con la longitud media de los intervalos de estas aproximaciones. Respecto de la esperanza, se crea una columna con el valor estimado y otra con los intervalos de las cotas. También se incluyen las columnas con los tiempos de ejecución y las mejoras de la solución.

En esta primera tabla incluimos los datos con las probabilidades marginales y esperanzas exactas, pues los tenemos de la tabla anterior. Así podemos comprobar la eficacia de las estimaciones calculadas. En el resto de tablas solo tendremos los valores aproximados.

	Exacta				Aproximada					
Pruebas	Mejoras	Fact_Min	Esperanza	Tiempo	Fact_Min	Fact_Media	Long_Media	Esperanza	Intervalos	Tiempo
0	0	0.974302799	381	75.3020527	0.968960696	0.990617048	0.003657779	396.4845898	[377.222174462801, 415.7470050391497]	0.1806917
1	1	0.965278893	150.6576939	79.7647364	0.957015854	0.986225196	0.006733719	155.3961532	[147.6111868659032 8, 163.1811945795795]	0.1518331
2	0	0.979838392	201.2950987	85.6404975	0.972229034	0.988528258	0.006972128	213.5357057	[196.9752915492476, 230.09611987477507]	0.156757
3	1	0.987387957	483.6385074	62.2158118	0.986430196	0.994922246	0.000772088	492.7380593	[483.6007593329342, 501.8753593662876]	0.1852182
4	2	0.947391924	461.453539	60.0912584	0.946019604	0.983662131	0.000899714	464.6727517	[461.3514735612093, 467.9940298289447]	0.1848619
5	1	0.928866865	63.46897322	74.6306814	0.774714164	0.893213582	0.188447894	71.08182519	[55.54050584128255, 86.6231445441897]	0.1274632
6	1	0.975312029	464.8645576	89.198169	0.970567156	0.988437421	0.003868643	485.7913137	[463.90472560763124, 507.67790181647763]	0.2010643
7	1	0.97970158	126.2169208	89.4724216	0.958362919	0.985263685	0.015202635	155.1011538	[134.10545258172726, 176.09685503573616]	0.2349165
8	2	0.89966898	378.4597537	93.7004031	0.897121389	0.962344656	0.002049765	381.3852404	[378.4597536755167, 384.31072705449617]	0.1972442
9	1	0.899672212	303.6768141	88.5467226	0.89390544	0.955613389	0.002306709	308.1172299	[303.6768140869582, 312.5576458105563]	0.1828209
10	1	0.936641977	532.6247467	53.1849129	0.931915472	0.966040591	0.004833409	542.7936036	[532.1403287177546, 553.4468785744139]	0.1369082
11	1	0.932918171	350.0330699	51.3319813	0.932918171	0.969664732	0	350.0330699	[350.0330699037349, 350.0330699037349]	0.1362086
12	1	0.879371675	326.76065	53.7065591	0.862448471	0.952613547	0.014817365	338.0033814	[322.95947118175474, 353.04729166352325]	0.1819667
13	0	0.870688928	185	33.1576918	0.799758967	0.93151556	0.04752456	206.341624	[185.0342538413631, 227.64899412227157]	0.1628498
14	4	0.770647885	113.8729042	47.2123699	0.656601858	0.838589849	0.082851867	148.5859279	[133.7631068087059, 163.40874901933557]	0.1765102
15	1	0.918530587	361.1356096	39.840111	0.902480951	0.962678107	0.012677841	378.9971731	[359.89925721921657, 398.0950889653327]	0.1826432
16	0	0.824407183	67.6616832	37.2765336	0.486339682	0.733580894	0.511966495	68.22053451	[45.67571250293588, 90.76535652228226]	0.2162406
17	1	0.679054761	260.0499379	32.7434195	0.611110999	0.849082315	0.0652967	276.5611234	[259.40316324821345, 293.7190834848128]	0.1288039
18	1	0.815064607	529.8891417	40.0617493	0.804211687	0.914936303	0.010196491	539.9981223	[529.7403674600283, 550.2558771671386]	0.2099775
19	2	0.886844556	463.7820303	39.7172529	0.873463033	0.942088521	0.01209	479.9421573	[463.29893472066556, 496.58537979449034]	0.1888251

Tabla 4.3. $n = 5$ trabajos.

De nuevo podemos distinguir que las diferencias entre las probabilidades de ser factible y las esperanzas, no varían mucho si utilizamos las fórmulas exactas o las estimaciones. Luego esto justifica que en el resto de tablas se puedan utilizar solo estas aproximaciones para llegar a buenas soluciones.

	Mejoras	Fact_Min	Fact_Media	Long_Media	Esperanza	Intervalos	Tiempo
0	3	0.985949625	0.997826043	0.002046423	286.9818036	[267.7193883211919, 306.24421889754063]	1.6005
1	1	0.995321174	0.999125155	0.000170592	662.7729532	[653.795922334487, 671.7499840974783]	0.6444241
2	0	0.998162012	0.999536253	0.00014134	906.8946688	[887.7130176175125, 926.0763200337672]	0.7415505
3	0	0.993617231	0.99878269	0.000386989	675.3085865	[658.7616824501765, 691.8554905035776]	0.6440213
4	1	0.986116048	0.997519833	0.000132863	821.4974043	[817.9623676645226, 825.0324408768821]	1.5409888
5	1	0.989744906	0.997848481	8.76527E-05	1074.227337	[1070.0652837953091, 1078.3893904102233]	1.2619279
6	0	0.993351326	0.997815147	0.000162207	1037.375082	[1031.889254717201, 1042.8609101224797]	0.7483299
7	1	0.99310286	0.998413026	8.01065E-05	563.5423475	[561.4814609100632, 565.6032341365291]	0.6830645
8	1	0.965785705	0.993012795	0.001969412	935.7568687	[913.5522788708415, 957.961458590893]	1.5198788
9	4	0.952962999	0.989721981	0.003046878	380.1789423	[370.09449401302015, 390.2633906164318]	1.7662169
10	0	0.9255963	0.974652418	4.69015E-06	450.1975353	[427.61281480334654, 472.78225584857296]	0.8941643
11	5	0.964214127	0.993616416	0.00347469	308.0911631	[295.9958295078717, 320.18649672327933]	2.2036867
12	1	0.911608647	0.966971849	0.149826984	391.0520514	[373.16113044969603, 508.94297234332964]	1.6818975
13	2	0.899187228	0.974181447	0.00011364	582.1766393	[581.8871698909522, 582.4661086163578]	1.7990795
14	3	0.958962838	0.987666449	0.003754757	793.813957	[774.8238398005989, 812.8040741368409]	2.106672
15	5	0.85594562	0.954159562	0.015236566	421.0072477	[407.2427763774317, 434.7717190098663]	3.3102784
16	2	0.755951721	0.932366782	0.013680764	784.0236608	[767.9161036958668, 800.1312179361717]	1.6447063
17	2	0.826538903	0.940804813	0.009917394	761.6832472	[749.6293695206984, 773.7371249560613]	1.9838971
18	1	0.751689612	0.941998559	0.004295296	391.9843182	[389.4632897038771, 394.5053466791509]	1.4505316
19	1	0.800045576	0.931182787	0.000363129	717.426724	[716.6768495954863, 718.1765984302821]	1.6315813

Tabla 4.4. $n = 10$ trabajos.

En general, podemos observar la longitud media de los intervalos de las probabilidades son bastante pequeñas, haciendo que la estimaciones del mínimo sean bastante fiables. Asimismo, los intervalos de la esperanza estimada son bastante estrechos, dando lugar también a buenas aproximaciones.

En las últimas tablas, denotamos por, por ejemplo, $2.95918E{-}08$ al número en notación científica 2.95918×10^{-8} y en las ocasiones en las que aparezcan 1 o 0

Pruebas	Mejoras	Fact_Min	Fact_Media	Long_Media	Esperanza	Intervalos	Tiempo
0	0	0.99907093	0.999915299	2.73871E-05	1003.613715	[991.1281223658732, 1016.0993067431351]	3.7687598
1	5	0.783727796	0.965222028	0.052481095	299.9922256	[279.549362997902, 320.43508826839866]	7.5855682
2	0	0.999966675	0.999995669	1.29655E-06	1675.522252	[1656.7198370759006, 1694.324666171391]	3.5946599
3	0	0.999979063	0.999998086	6.41918E-07	1651.131565	[1630.9856188140784, 1671.2775111424394]	4.1376191
4	1	0.936005243	0.992579298	0.00940244	410.835855	[387.46188699784955, 434.209823087052]	8.2797305
5	2	0.996193819	0.999588356	0.000299006	758.1952584	[736.1604682619451, 780.2300485719663]	5.8643228
6	0	0.999657479	0.999960875	9.41513E-06	1953.2866	[1934.1412162712459, 1972.431984263272]	3.6364092
7	0	0.998405737	0.99978822	5.31445E-05	1063.677059	[1053.7693118646464, 1073.5848070061984]	3.531656
8	3	0.989039286	0.998749705	0.000149101	1610.595482	[1601.4991083533632, 1619.6918548205847]	9.9051419
9	0	0.990475083	0.999290749	0.094321704	328.5701453	[308.72193251722587, 348.4183579966357]	6.1367209
10	0	0.99673251	0.999470115	5.99189E-05	1386.799552	[1380.2409443797114, 1393.3581606039102]	3.729701
11	1	0.996583021	0.999242152	0.000475914	745.4134032	[727.6504047246787, 763.1764016410932]	3.6646243
12	6	0.95697812	0.993904874	0.000845563	1721.842998	[1708.779541628175, 1734.906454096885]	15.9489386
13	2	0.976718074	0.995585087	0.000332434	2144.298126	[2136.0397647607033, 2152.5564864389253]	7.9398312
14	0	0.990143276	0.998786371	9.41192E-05	1828.826994	[1822.1328372751182, 1835.5211516187392]	3.6871876
15	1	0.99508272	0.999160049	0.00015571	1667.643959	[1654.2784113120797, 1681.0095066482681]	3.578931
16	2	0.687002955	0.948689378	0.022653022	748.4943372	[725.2653590281459, 771.723315444073]	7.6586743
17	1	0.097954573	0.736776903	0.162230233	343.9742743	[321.2772421713722, 366.6713064753419]	7.6069104
18	1	0.946204673	0.990855215	0.001265795	1982.187411	[1966.8759811925715, 1997.4988402813478]	9.3835588
19	5	0.6730282	0.945495754	0.023263585	660.0245211	[644.4778203747916, 675.5712218191252]	17.1013986

Tabla 4.5. $n = 20$ trabajos.

	Mejoras	Fact_Min	Fact_Media	Long_Media	Esperanza	Intervalos	Tiempo
0	6	0.999999962	0.999999998	5.85763E-10	3742.753557	[3719.022562164154, 3766.4845517111225]	308.4923084
1	11	1	0.999231659	4.54E-08	725.8117509	[702.9341996944544, 748.6893020267902]	488.0660314
2	10	1	1	4.15E-13	4974.459548	[4970.108988974609, 4978.810107563045]	446.6823909
3	14	1	1	1.2192E-13	5323.266444	[5298.771148944684, 5347.76173844722]	545.482962
4	11	0.610308392	0.97116615	0.034699988	638.7572255	[614.0135842097571, 663.5008668868696]	501.722146
5	7	0.99999928	0.999999957	1.38092E-08	3693.359699	[3672.6140888643786, 3714.1053096288083]	346.5310599
6	13	0.999999972	0.999999998	4.38155E-10	4398.259492	[4376.7756853457595, 4419.743298216039]	446.8884855
7	12	0.999999998	1	3.83944E-12	5199.148819	[5194.627449447829, 5203.6701880310375]	450.8469031
8	8	0.998722884	0.999933839	1.7342E-05	3112.263585	[3094.56867389007, 3129.9584962304925]	346.6787742
9	10	0.999969369	0.999998324	2.59709E-07	4271.81797	[4258.676075944203, 4284.959864756295]	447.2286231
10	6	0.999324676	0.999959381	2.71547E-05	1672.519825	[1651.3401831708888, 1693.699467163638]	247.120549
11	0	0.999999495	0.999999955	7.50685E-09	3882.997764	[3871.3518943367135, 3894.643632842976]	46.5870918
12	11	0.993701369	0.999638484	5.24115E-05	4794.645409	[4776.33464302012, 4812.956175507141]	351.7915066
13	4	0.99070775	0.999354804	0.000299744	1994.873316	[1971.7585619734896, 2017.9880690350901]	101.259039
14	1	0.998212177	0.99981416	6.22812E-06	2478.634239	[2476.8637711149368, 2480.4047078291333]	54.4498845
15	5	0.999228435	0.999923419	3.66202E-05	2147.253579	[2126.360182150753, 2168.146976584638]	154.5052617
16	10	0.823936603	0.984507624	0.003337546	2821.009087	[2798.4013491241003, 2843.616825040114]	404.7889904
17	6	2.84941E-08	0.599978383	0.061675173	853.3112139	[828.9892089986408, 877.6332188272039]	172.6797596
18	4	1.51309E-08	0.600407301	0.060922429	887.0374411	[864.1945948396749, 909.880287421109]	92.4852646
19	0	0.996325362	0.999708527	3.67567E-05	4434.708576	[4421.251700461642, 4448.165451605575]	53.379287

Tabla 4.6. $n = 50$ trabajos.

en las tablas es porque exceden el límite de precisión de Python y este redondea los números.

Nótese que a diferencia de en el artículo [2], los tiempos de ejecución de las tablas son mucho mayores. Esto se debe al uso de diferentes lenguajes de programación en la creación de la esperiencia computacional. Artículos como [16] tratan este tema.

De todo el análisis podemos concluir que la heurística es efectiva y proporciona soluciones, que en muchos casos, son equivalentes a las óptimas, en mucho menos tiempo que el procedimiento estándar. Al mismo tiempo, las estimaciones

Pruebas	Mejoras	Fact_Min	Fact_Media	Long_Media	Esperanza	Intervalos	Tiempo
0	15	1	1	1.15463E-16	7310.518626	[7292.959604021416, 7328.0776471167355]	4156.124437
1	12	0.999999747	0.999999994	4.45426E-09	2796.691585	[2777.9525898782636, 2815.4305802341996]	4083.071709
2	10	0.999995616	0.999999875	1.37425E-07	2199.634004	[2179.12893955527, 2220.139067586211]	3477.131526
3	17	1	1	0	3825.437322	[3824.9587068984215, 3825.915936938486]	4465.338692
4	11	0.999998284	0.999999953	2.55321E-08	3751.389637	[3731.1372102048704, 3771.6420631827377]	3440.421045
5	16	1	1	1.22125E-17	9002.083109	[9002.056340291238, 9002.109877506511]	4487.67328
6	9	1	1	8.99281E-17	8050.721649	[8044.140508560219, 8057.302788494681]	2988.758164
7	14	1	1	0	10854.27635	[10839.85146417193, 10868.701238518073]	3493.39371
8	17	0.989693983	0.999663888	0.000259186	2230.95765	[2206.213671388987, 2255.7016277137413]	4813.567734
9	12	0.999999992	1	1.02173E-11	8799.433518	[8795.981003974852, 8802.886031598704]	3315.5724
10	12	1	1	8.14073E-13	7887.151033	[7878.308230925801, 7895.993835276891]	3180.092544
11	10	0.999999999	1	3.98715E-12	4849.082867	[4846.310194178615, 4851.855539884343]	3119.844822
12	21	0.999700251	0.999989592	1.18328E-06	10010.08062	[9993.941582665437, 10026.219667320156]	4986.256784
13	20	0.09428984	0.880465455	0.030388491	1808.790545	[1787.573569614998, 1830.0075209299298]	5167.471276
14	11	0.999818332	0.999992502	3.10183E-06	4604.484566	[4581.45019829274, 4627.518933696339]	3398.084565
15	20	0.999999992	1	3.86637E-11	9790.342101	[9777.523758230032, 9803.16044397012]	4451.279962
16	16	0.979346359	0.999060579	0.000108563	9619.944348	[9597.979174359718, 9641.909520870964]	3584.315136
17	11	0.985152022	0.999352185	0.000141063	5380.0567	[5359.398379806188, 5400.715020266836]	2970.313729
18	14	0.990259438	0.999465054	0.000106094	4098.27762	[4085.180239254065, 4111.375001636695]	2534.723919
19	9	0.999991817	0.999999724	2.95918E-08	10865.13173	[10850.5756259644, 10879.687826858368]	2252.688948

Tabla 4.7. $n = 100$ trabajos y $w'_1 = w'_2 = 0.5$ en las estimaciones de la esperanza.

Pruebas	Mejoras	Fact_min	Fact_Media	Long_Media	Esperanza	Intervalos	Tiempo
0	10	1	1	1.15463E-16	7301.739115	[7292.959604021416, 7328.0776471167355]	3006.219962
1	14	0.999999747	0.999999994	4.45426E-09	2787.322087	[2777.9525898782636, 2815.4305802341996]	4343.55401
2	12	0.999995616	0.999999875	1.37425E-07	2189.381472	[2179.12893955527, 2220.139067586211]	3580.858679
3	13	1	1	0	3825.198014	[3824.9587068984215, 3825.915936938486]	3968.992777
4	10	0.999998284	0.999999953	2.55321E-08	3741.263423	[3731.1372102048704, 3771.6420631827377]	3156.333796
5	17	1	1	1.22125E-17	9002.069725	[9002.056340291238, 9002.109877506511]	4347.19827
6	15	1	1	8.99281E-17	8047.431079	[8044.140508560219, 8057.302788494681]	3721.574279
7	12	1	1	0	10847.06391	[10839.85146417193, 10868.701238518073]	2607.429763
8	17	0.989693983	0.999663888	0.000259186	2218.58566	[2206.213671388987, 2255.7016277137413]	4885.633217
9	10	0.999999992	1	1.02173E-11	8797.707261	[8795.981003974852, 8802.886031598704]	2599.797251
10	16	1	1	8.14073E-13	7882.729632	[7878.308230925801, 7895.993835276891]	3648.761535
11	18	0.999999999	1	3.98715E-12	4847.696531	[4846.310194178615, 4851.855539884343]	3560.168916
12	7	0.999700251	0.999989592	1.18328E-06	10002.0111	[9993.941582665437, 10026.219667320156]	898.8613562
13	15	0.09428984	0.880465455	0.030388491	1798.182057	[1787.573569614998, 1830.0075209299298]	3890.713188
14	10	0.999818332	0.999992502	3.10183E-06	4592.967382	[4581.45019829274, 4627.518933696339]	3292.021359
15	21	0.999999992	1	3.86637E-11	9783.93293	[9777.523758230032, 9803.16044397012]	4311.827887
16	16	0.979346359	0.999060579	0.000108563	9608.961761	[9597.979174359718, 9641.909520870964]	2488.119029
17	11	0.985152022	0.999352185	0.000141063	5369.72754	[5359.398379806188, 5400.715020266836]	2383.057038
18	14	0.990259438	0.999465054	0.000106094	4091.72893	[4085.180239254065, 4111.375001636695]	3753.432575
19	9	0.999991817	0.999999724	2.95918E-08	10857.85368	[10850.5756259644, 10879.687826858368]	970.8024401

Tabla 4.8. $n = 100$ trabajos y $w'_1 = 0.75$, $w'_2 = 0.25$ en las estimaciones de la esperanza.

generadas también son bastante fiables pudiendo intercambiarlas por los valores exactos y así evitar la naturaleza NP-dura del problema HSSOP.

Conclusiones

En este Trabajo de Fin de Grado se ha logrado realizar un análisis general de los diferentes problemas de Planificaón y Secuenciación, describiendo las ideas principales de estos, algunas de las herramientas matemáticas como la modelización para su resolución y diversas clasificaciones que pueden crearse para esquematizar y catalogar los problemas.

Con el objetivo de explorar en profundidad alguno de estos problemas, se desarrolló el HSSOP o problema de Secuenciación y Ordenación Estocástico Jerárquico, presentando el modelo, algunas propiedades y teoremas enunciados y demostrados en el artículo [2].

Se prosiguió con el desarrollo teórico de dos algoritmos: uno exacto, que sigue un método exhaustivo, y la heurística creada en el artículo [2].

Por último, se programaron los dos procedimientos en Python 3.8 para no solo emular la experiencia computacional hecha en el artículo [2], sino también para poder desarrollar otros problemas de Ordenación Secuencial Estocásticos Jerárquicos con distintos objetivos en el futuro. Además, se llegaron a resultados similares a los del artículo original, probando de nuevo la efectividad de la heurística.

A

Apéndice

A.1. Implementación de los algoritmos en Python

En este apéndice se muestran algunos extractos del código utilizado en la aplicación del problema HSSOP con las explicaciones de su funcionamiento.

A.1.1. Preliminares

Las librerías utilizadas en el programa son:

```
# libraries:
import os
import sys
from sympy import *
import numpy as np
from scipy.stats import gamma
import random as rand
import pandas as pd
import itertools
from timeit import default_timer
import warnings
```

Previo al inicio del algoritmo, se calcula de manera simbólica (si es necesario) las probabilidades marginales exactas y la integral de la esperanza del tiempo de completación del último trabajo.

```
def Prob_Cj_v2(m:int, t=Symbol('t')):
    """ It calculates the marginal probability for P(C_m <= t)

    Args:
        m (int): The m-th work
        t (_type_, optional): Variable used for recursion.
        Defaults to Symbol('t').

    Returns:
        _type_: Symbolic expression of P(C_m <= t)
    """
    r = Symbol('r0')
    lamb = Symbol('lamb')
```

```
        q = Symbol('q')
        costs = []
        for i in range(m + 1):
            row = []
            for j in range(m + 1):
                row.append(Symbol(f'c{i}{j}'))
            costs.append(row)

        dist_exp = 1 - exp(-lamb*q)
        dens_exp = lamb*exp(-lamb*q)

        if m == 0:
            return dist_exp.subs(q, t - r)

        else:
            h = Symbol(f'h{m}')
            x = Symbol(f'x{m}')
            fpm = dens_exp.subs(q, x)
            int_fun = fpm*Prob_Cj_v2(m - 1, t - costs[m - 1][m] - x)
            f = integrate( int_fun, (Symbol(f'x{m}'), 0, t - h))
            return f

    def ExactMakespan(m:int):
        """ Calculates the symbolic exact expected makespan

        Args:
            m (int): job number

        Returns:
            _type_: Symbolic expression of E[Cmax(S)]
        """

        Fn = Prob_Cj_v2(m - 2)
        h = Symbol(f'h{m - 1}')
        lamb = Symbol('lamb')
        t = Symbol('t')
        c = Symbol(f'c{m - 2}{m - 1}')
        int_fun = (t + c)*(diff(Fn, t))
        return h*Fn.subs(t, h - c) + integrate(int_fun, (t, h - c, oo))
        + 1/lamb


    def Preprocessing(m:int) -> list:
        Prob_fun = []
        for j in range(m):
            Prob_fun.append(Prob_Cj_v2(j))

        return Prob_fun, ExactMakespan(m)
```

En la práctica, gracias a la librería Sympy, se ha comprobado de manera empírica que el tiempo de ejecución del algoritmo disminuye si se calculan las fórmulas primero. Cada vez que se quiera ver la probabilidad de factibilidad de una secuencia o el valor esperado del tiempo de completación del último trabajo, solo hay que sustituir los valores en las fórmulas ya calculadas.

Como también vamos a trabajar con dos tipos de índices: el número del trabajo y su posición actual en la secuencia S, también se necesita una función auxiliar que contemple los cambios de orden de los tiempos de preparación (o costes) c_a dentro de la solución.

```
    def Cost_reorder(Sol: pd.DataFrame, Original_Costs: pd.DataFrame)
    -> pd.DataFrame:
        """ Reorder costs acording to its current position

        Args:
            Sol (pd.DataFrame): Current solution
            Original_Costs (pd.DataFrame): Original costs without
    change

        Returns:
            pd.DataFrame: Changed costs
        """
        new_costs = Original_Costs.copy()
        Changes = Sol['Jobs'].to_dict()
        for i in range(len(Original_Costs)):
            for j in range(len(Original_Costs)):
                i_new = Changes[i]
                j_new = Changes[j]
                new_costs.iloc[i_new, j_new] =
                Original_Costs.iloc[i][j]

        return new_costs
```

Por último vamos a generar las 20 pruebas para los valores de n, siendo n el número total de trabajos a secuenciar. Para ello, definimos primero las variables TF ("Tardiness Factor") y RDD ("Range Due Date"), y luego una función en la que se pueda fijar la semilla para regular y comprobar que todo se genera correctamente (a la hora de generar las tablas de la experiencia computacional, la semilla se deja libre).

```
    # Global variables:
    tardiness_factor = [0,0.2,0.4,0.6,0.8] # Tardiness Factor
    range_due_date = [0.2,0.4,0.6,0.8] # Range of due date

    # Functions:
    def initial_solution(jobs_num:int, seed=None) -> list:
        """ Generates 20 Cases of Jobs_num jobs with their costs
        and the value of lambda.

        Args:
            Jobs_num (int): Number of jobs
            seed (int, optional): Seed for random (if we want one).
            Defaults to np.nan.

        Returns:
            list: List with pairs of costs,
            lambda and Inicial solutions.
        """

```

```
        # Fix seed
        if seed:
            rand.seed(seed)
            np.random.seed(seed=seed)

        # Jobs
        jobs = list(range(0,jobs_num))

        sol_list = []


        for i in tardiness_factor:
            for j in range_due_date:

                # We generate rj as uniforms [0, 50]
                release_times = [rand.uniform(0, 50)
                for x in range(jobs_num)]

                # Generamos 1/lambda y los pj com exp(lambda)
                lambda_inv = rand.uniform(0, 100)
                lamb = 1 / lambda_inv

                processing_time = [rand.expovariate(lamb)
                for x in range(jobs_num)]

                # Generamos los cij como uniformes[0, 20]
                setup_costs = pd.DataFrame(np.random.uniform(0, 20,
                size=(jobs_num, jobs_num)))
                # Nota: cii = 0
                for k in range(jobs_num):
                    setup_costs.iloc[k][k] = 0

                # Generamos los dj
                L1 = jobs_num * 1/lamb * (2 - i - j/2)
                L2 = jobs_num * 1/lamb * (2 - i + j/2)
                due_dates = []
                for k in release_times:
                    due_dates.append(rand.uniform(k + L1, k + L2))

                zipped = list(zip(jobs, release_times,
                processing_time, due_dates))
                pre = pd.DataFrame(zipped,
                columns=['Jobs', 'rj', 'pj', 'dj'])
                sol_list.append([pre, setup_costs, lamb])

        return sol_list
```

A.1.2. Algoritmo exacto

Con las funciones anteriores, se inserta el código utlilizado para la implementación del algortimo exacto ya definido en el capítulo 3.

Los parámetros de entrada del algortimo son la solución inicial, los costes originales c_a (que equivalen a los tiempos de preparación), el parámetro λ, las

funciones de las probabilidades marginales exactas calculadas en los preliminares, al igual que la función de la esperanza del tiempo de completación del último trabajo de la secuencia, y por último una serie de enteros que valen 1 o 0 en función de la tabla que estemos calculando.

El primer entero ("Fes-option") indica si queremos utilizar la probabilidad de factibilidad exacta (1) o si queremos utilizar el criterio del mínimo de las probabilidades marginales (0). El segundo ("dj-option") cuestiona si se utilizan las probabilidades marginales exactas (1) o si se toman las estimaciones (0). El último ("Emax-option"), al igual que el segundo revisa si se parte de la fórmula teórica de la esperanza del tiempo de completación del último trabajo (1) o si se toman las estimaciones (0).

Otras funciones que aparecen en el código son las de "Feasible" y la de "ExpectedMakespan", que calculan respectivamente la probabilidad de ser factible y la esperanza del tiempo de completación del último trabajo de la solución introducida. Estas funciones se basan en los desarrollos hechos en el capítulo 3 y dado que este apéndice es meramente ilustrativo, no vamos a incluir sus códigos en este trabajo.

```
1    def Exact_Procedure(Sol_inicial:pd.DataFrame, Original_Costs:pd.
     DataFrame, lamb:float, fun_prob:list, Ex_fun, Fes_option:int,
     dj_option:int, Emax_option:int) -> list:
2
3        # Start timer
4        start = default_timer()
5
6        ## Initialization phase:
7        Sol_Costs = Cost_reorder(Sol_inicial, Original_Costs)
8        P_flag = Feasible(Sol_inicial, Sol_Costs, lamb, fun_prob,
9        Fes_option, dj_option)[0]
10
11       # We calculate all the permutations (all possible solutions)
12       Jobs = Sol_inicial['Jobs'].to_list()
13       Soluciones = []
14       Unknown = list(itertools.permutations(Jobs))
15
16       # Here we can add a module for the verification
17       # of the precedence relationships
18
19       print('Factibility fase:')
20       ## Feasibility probability phase
21       for i in Unknown:
22           # Reorder Sol_inicial and its costs
23           Sol_i = Sol_inicial.reindex(i).reset_index(drop=True)
24           Cost_i = Cost_reorder(Sol_i, Original_Costs)
25
26           # feasibility
27           P_i = Feasible(Sol_i, Cost_i, lamb,
28           fun_prob, Fes_option, dj_option)[0]
29
```

```
            if P_i > P_flag:
                P_flag = P_i
                Soluciones = [i]

            elif P_flag - P_i <= 0.00001:
                Soluciones.append(i)


        print('Secundary fase:')
        ## Expected makespan phase
        Unknown = Soluciones.copy()
        Soluciones = []
        E_flag = 10000000000000
        for i in Unknown:
            # Reorder Sol_inicial and its costs
            Sol_i = Sol_inicial.reindex(i).reset_index(drop=True)
            Cost_i = Cost_reorder(Sol_i, Original_Costs)

            # Calculate expected makespan
            Emax_i = ExpectedMakespan(Sol_i, Cost_i,
            lamb, Ex_fun, Emax_option)[0]

            if Emax_i < E_flag:
                E_flag = Emax_i
                Soluciones = [i]

            elif Emax_i - E_flag <= 0.00001:
                Soluciones.append(i)

        Time_opt = default_timer() - start

        Optimal_Solution = [Soluciones, P_flag, E_flag, Time_opt]
        return Optimal_Solution
```

Los resultados que genera esta función son las soluciones óptimas, los valores de probabilidad de factibilidad y esperanza del tiempo de completación del último trabajo óptimos, y el tiempo de ejecución del algortimo.

Procedimiento heurístico

En esta sección se presentará el código utilizado en la creación y programación del segundo algortimo resolutivo del HSSOP.

Este tomará los mismos parámetros de entrada que el algoritmo exacto, añadiéndole "Tol", que mide el nivel de probabilidad de ser factible mínimo que exigimos a la solución (1 - Tol $= 1 - \overline{\alpha}$ y alpha es $1 - \alpha$ en la teoría del capítulo 3), también "eps"que es el ϵ de la fase 1, y w_1 y w_2 que se utilizan como parámetros en la estimación del $E[C_{\text{máx}}(S)]$.

En la práctica, el valor de tolerancia de $1 - \overline{\alpha}$ es del 0.995, es decir, las secuencias cuya probabilidad de que sean factibles pasen de 0.995 irán directamente a la Fase 2, y el valor de ϵ es 0.0001.

A lo largo del programa aparece la variable "impr"(y relacionadas como "Red-impr"o "Mine-impr"). Estas almacenan el número de mejoras que realiza el algortimo, es decir, el número de veces que altera el orden de los trabajos en los diferentes módulos para mejorar la solución.

El resto de funciones que aparecen en el programa ("RearrangeModule", "ExchangeModule", "ReduceIdleTime"y "MimizeExpectedMakespan") llevan a cabo los módulos explicados en el capítulo 3 (son el módulo de reordenación, módulo de intercambio, reducción del tiempo ocioso y minimizar la esperanza del tiempo de completación del último trabajo de la secuencia respectivamente), donde se añaden algunos booleanos para ver si el módulo tuvo éxito ("Success") en la mejora de la solución o si se necesita sumar el ϵ.

```
    def Heuristic_Procedure(Sol_inicial:pd.DataFrame,
                        Original_Costs:pd.DataFrame, lamb:float,
                        fun_prob:list, Ex_fun,
                        Fes_option:int, dj_option:int, Emax_option:int,
                        Tol:float, eps:float,
                        w_1:float=0.5, w_2:float=0.5) -> list:


    start = default_timer()
    impr = 0

    # Fase 1:
    print('FASE 1:')

    # We create the initial solution using the EDD rule
    Sol = Sol_inicial.sort_values(by = ['dj']).reset_index(drop=True)
    Sol_Costs = Cost_reorder(Sol, Original_Costs)
    alpha = Feasible(Sol, Sol_Costs, lamb, fun_prob, Fes_option,
    dj_option)[0]

    while 1 - Tol > alpha:
        print(alpha)
        RM_Success, EM_Success = True, True
        print('RearrangeModule:')
        Sol, alpha, RM_Success, Is_Fes = RearrangeModule(Sol,
    Original_Costs, lamb,
                                                         alpha, fun_prob,
    Fes_option, dj_option)

        if RM_Success:
            impr += 1

        if Is_Fes:
            alpha += eps
        else:
            print('ExchangeModule:')
            Sol, alpha, EM_Success, Is_Fes = ExchangeModule(Sol, alpha,
     Original_Costs, lamb,
                                                            fun_prob,
    Fes_option, dj_option)
```

```
36
37              if EM_Success:
38                  impr += 1
39
40              if Is_Fes:
41                  alpha +=eps
42
43          if not (RM_Success or EM_Success):
44              break
45
46      Sol_Costs = Cost_reorder(Sol, Original_Costs)
47      alpha = Feasible(Sol, Sol_Costs, lamb, fun_prob, Fes_option,
        dj_option)[0]
48      # Fase 2:
49      print('FASE 2')
50      Sol, alpha, Red_impr = ReduceIdleTime(Sol, alpha, Original_Costs,
        lamb, fun_prob, Fes_option, dj_option)
51      impr += Red_impr
52
53      # Fase 3:
54      print('FASE 3:')
55      Sol, alpha, Emax, Mine_impr = MinimizeExpectedMakespan(Sol,
        Original_Costs, lamb, fun_prob, Ex_fun, alpha, Fes_option, dj_option
        , Emax_option, w_1, w_2)
56      impr += Mine_impr
57
58      print(Red_impr, Mine_impr)
59
60      Sol_Costs = Cost_reorder(Sol, Original_Costs)
61      alpha = Feasible(Sol, Sol_Costs, lamb, fun_prob, Fes_option,
        dj_option)
62      Emax = ExpectedMakespan(Sol, Sol_Costs, lamb, Ex_fun, Emax_option,
        w_1, w_2)
63      Job_secuence_aprox = Sol['Jobs'].tolist()
64      Time_aprox = default_timer() - start
65
66      return Job_secuence_aprox, impr, alpha, Emax, Time_aprox
```

Los resultados que da este algoritmo son: la secuencia resultado calculada, el número de mejoras de la solución, el valor de probabilidad de factibilidad (junto con otros parámetros como la longitud media de los intervalos si trabajamos con las estimaciones), la esperanza del tiempo de completación del último trabajo (junto a sus cotas si es la estimada) y el tiempo de ejecución.

Gracias a estas funciones, se pueden definir los parámetros iniciales de estudio y calcular los resultados necesarios para la creación de las tablas del capítulo 4.

Bibliografía

[1] Alcaide López de Pablo, D. *Problemas de Planificación y Secuenciación Determinística: Modelización y Técnicas de Resolución* Tesis Doctoral. Departamento de Estadística, Investigación Operativa y Computación. Universidad de La Laguna. Tenerife, España (1995). Publicado también por el Servicio de Publicaciones de la Universidad de La Laguna. Soportes Audiovisuales e Informáticos. Serie Tesis Doctorales. Curso 1995/96. Ciencias y Tecnoligías. Vol. 10. Servicio de Publicaciones Universidad de La Laguna, 2004.

[2] Alcaide, D., Rodríguez-González Á., Sicilia J. An approach to solve a hierarchical stochastic sequential ordering problem. *Omega, The International Journal of Management Science*, 2003, vol. 31, pp. 169–187. https://doi.org/10.1016/S0305-0483(03)00027-6

[3] Alcaide, D., Rodríguez-González Á., Sicilia J. A heuristic approach to minimize expected makespan in open shops subject to stochastic processing times and failures, *International Journal and Flexible Manufacturing Systems*, 17, pp. 201–226 (2005). https://doi.org/10.1007/s10696-006-8819-1

[4] Alcaide, D. On Scheduling Models. *Boletín de la Sociedad de Estadística e Investigación Operativa*, 2008,Vol. 24, número 2, pp. 11–21.

[5] Alcaide, D., C. Chu, V. Kats, E. Levner, G. Sierksma. Cyclic multiple-robot scheduling with time-window constraints using a critical path approach. *European Journal of Operational Research*, 2007, vol. 177, pp. 147–162. European Journal of Operational Research, ISSN 0377-2217. https://doi.org/10.1016/j.ejor.2005.11.019.

[6] Baker, K.R. *Introduction to Sequencing and Scheduling*, John Wiley, 1974.

[7] Crama, T., V. Kats, J. van de Jlundert, E. Levner. Cyclic scheduling in robotic flowshops. *Annals of Operations Research*, 2000 vol. 96, pp. 97–124. https://doi.org/10.1023/A:1018995317468.

[8] French, S. *Sequencing and Scheduling an Introduction to the Mathematics of the Job Shop.*, Ellis Horwood Series, 1982. vol. 96, 1974, pp. 97–124.

https://doi.org/10.1002/net.3230130218.

[9] Garey, M., D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.*, W.H. Freeman y colaboradores, San Francisco, 1979. https://doi.org/10.1137/1024022

[10] Graham, R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics.*, 5, 1979, pp. 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

[11] Hoogeveen, J.A. *Single-Machine Bicriteria Scheduling*, PhD Thesis. CWI, The Netherlands Technology, Amsterdam, 1992.

[12] Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan. Recent developments in deterministic sequencing and scheduling: a survey. *Sempster, M.A.H., J.K. Lenstra, A.H.G Rinnooy Kan (eds.) Deterministic and stochastic scheduling. NATO Advanced Study Institutes Series, serie C*, vol. 84, 1982.

[13] Levner, E., V. Kats, D. Alcaide López de Pablo. Cyclic scheduling in robotic cells: An extension of basic models in machine scheduling problems. *E. Levner, ed. Multiprocessor Scheduling, Theory and Applications. I-Tech. Publishers, Viena, Austria*, 2007, ISBN 978-3-902613-02-8, 436 pp., doi: 10.5772/5212

[14] Levner, E., V. Kats, S. Alcaide López de Pablo, T.C.E. Cheng. *Complexity of cyclic scheduling problems: A state-of-the-art survey.* Computers & Industrial Engineering, Volume 59, Issue 2, 2010, pp. 352-361, ISSN 0360-8352. https://doi.org/10.1016/j.cie.2010.03.013.

[15] Miguel Sánchez García. *El Proceso de modelización en la investigación actual: discurso inaugural del curso 1981-1982* , Universidad de La Laguna. Secretariado de Publicaciones, 1982, 66 pp.

[16] Oden L., "Lessons learned from comparing C-CUDA and Python-Numba for GPU-Computing,"2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Västerås, Sweden, 2020, pp. 216-223, doi: 10.1109/PDP50117.2020.00041.

[17] Pinedo, M. *Scheduling: Theory, Algorithms and Systems*,Prentice Hall, N.J., 2002. https://doi.org/10.1007/978-3-319-26580-3

[18] Pinedo, M. *Planning and Scheduling in Manufacturing and Services*, Springer, New York, 2005. https://doi.org/10.1007/978-1-4419-0910-7

[19] Python Core Team: *Python: A dynamic, open source programming language.*. Python Software Foundation, 2021, https://www.python.org/. Python version 3.8.

[20] Romero-López, C. *Técnicas de Planificación y Control de Proyectos*, Editorial Pirámide. Madrid, 1983.

An Approach to Scheduling and Sequencing Problems

Ignacio Domínguez Espinosa
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu0101316420@ull.edu.es

Abstract

THIS document provides an approach to the problems of Scheduling and Sequencing. It applies tools from the field of Operational Research to tackle some of these problems, while also working with a parametric classification of them.

It delves into the Hierarchical Stochastic Sequential Ordering problem, examining various relevant results and algorithmic procedures found in the literature to address this problem.

Furthermore, a new implementation in Python 3.8 is provided for the mentioned algorithmic procedures, laying the foundations for investigating other similar problems in the near future.

1. Introduction

SCHEDULING AND SEQUENCING PROBLEMS are a branch of Operations Research that involve the management and allocation of different tasks or phases that make up a problem to machines or workers in order to optimize some criteria (e.g., completing it on time). Due to the variety of problems, a tripartite classification $\alpha|\beta|\gamma$ was created to categorize and summarize many of them, based on who performs the tasks (machines, people, etc.), what needs to be done (jobs, activities, etc.), and why it needs to be done (verify a timeframe, avoid delays, etc.). In the $\alpha|\beta|\gamma$ scheme:

- α contains the characteristics of the machines, such as the number involved, specialization, specific processing times, etc.
- β specifies the jobs, including precedence relationships, time windows, or allowances for interruptions, among others.
- γ determines the various objective functions that these optimization problems can have. They are often sums, maxima, or minima

Furthermore, these problems can be classified as stochastic or deterministic depending on the presence of random variables. Multiple criteria can be simultaneously considered, and cyclic planning can be implemented where jobs are repeated in the same order over time.

In the modeling of these problems, various variables and data are typically considered to quantify the qualities of each job j. In the following graph, we can observe how jobs are commonly represented in the literature of this type of problems:

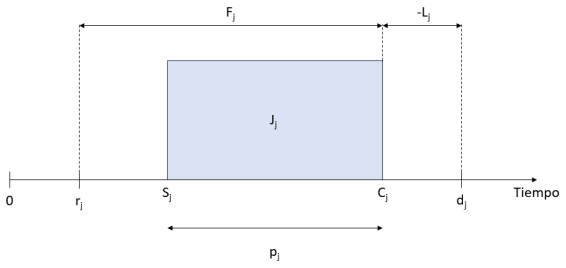

Figure 1: *Job J_j with its variables and data, where the x-axis represents the planning time.*

Where, $\forall$ jobs j:

- Availability times are r_j.
- Due dates are d_j.
- Starting time of the jobs are S_j.
- Completion time of the jobs are C_j.
- Processing times are p_j.
- Possible delays are L_j.

among others.

2. Outline

In this document, we will delve into the HSSOP (Hierarchical Stochastic Sequential Ordering Problem), where we work with a single machine, jobs have preference relationships and time windows, and there are random variables, specifically the job processing times and therefore the completion times. The stochastic factor requires the use of probabilistic tools to assess the feasibility of a solution. In our case $P(S \text{ feasible}) = P(C_1 \leq d_1, C_2 \leq d_2, ..., C_n \leq d_n)$

To solve the problem, the [1] model is developed, aiming to find the solution(s) with maximum feasibility and minimum expected makespan, i.e. $\max\{P(S \text{ feasible})\}$ & $\min\{E[C_{\max}(S)]/P(S \text{ feasible}) = P^*\}$, where S is a sequence of jobs that satisfies the priority orders and time windows and P^* is the maximum feasibility.

3. Analysis and resolution methods

In the article [1], it is proven that this problem is NP-hard in terms of the number of jobs, meaning that as the number of jobs increases, the execution time increases even more. Therefore, to solve the HSSOP, two algorithms will be proposed: an exhaustive one that analyzes all possible combinations of the problems in search of optima, and a heuristic one that starts with an initial solution and alters the order of jobs to improve the solution.

4. Applications of the HSSOP model

To build upon the work of [1], the algorithms will be implemented in Python. This will not only replicate the computational experience of the previous article but also allow for the implementation of other criteria in the future.

References

[1] Alcaide, D., Rodríguez-González Á., Sicilia J. An approach to solve a hierarchical stochastic sequential ordering problem. *Omega, The International Journal of Management Science*, 2003, vol. 31, pp. 169–187.

[2] Alcaide, D., Rodríguez-González Á., Sicilia J. *A heuristic approach to minimize expected makespan in open shops subjecto to stochastic processing times and failures*, Springer, 2006.

[3] Alcaide, D. On Scheduling Models. *Boletín de la Sociedad de Estadística e Investigación Operativa*, 2008,Vol. 24, número 2, pp. 11–21.