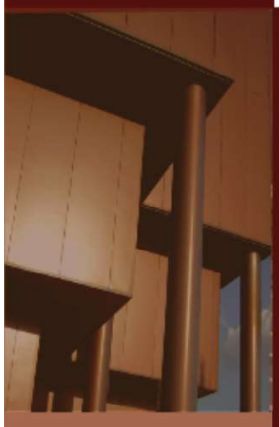
Trabajo Fin de Grado
Ingeniería de Organización Industrial

Integración de planificación de intervenciones en quirófanos y camas postoperatorias

Autor: África Nieto Medina
Tutor: Víctor Fernández-Viagas Escudero
Tutor Externo: José Manuel Molina Pariente



Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo de Fin de Grado
Ingeniería de Organización Industrial

Integración de planificación de intervenciones en quirófanos y camas postoperatorias

Autor:
África Nieto Medina

Tutor:
Víctor Fernández-Viagas Escudero
Profesor Titular

Tutor Externo:
José Manuel Molina Pariente
Profesor Ayudante Doctor

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Grado: Integración de planificación de intervenciones en quirófanos y camas postoperatorias

Autor: África Nieto Medina

Tutor: Víctor Fernández-Viagas Escudero

Tutor Externo: José Manuel Molina Pariente

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

Agradecimientos

A mi tutor, Víctor Fernández-Viagas Escudero, por haber confiado en mí incluso cuando yo no lo hacía. A mi familia, por enseñarme a superar las adversidades y apoyarme incondicionalmente. A Juan Carlos, por permanecer a mi lado en cada instante. Sin vosotros esto no hubiera sido posible. Esto solo acaba de empezar.

África Nieto Medina

Sevilla, 2021

Resumen

La pandemia provocada por el SARS-CoV-2 está llevando al sistema sanitario español a un entorno de gran variabilidad e incertidumbre que puede provocar la congestión y escasez de recursos sanitarios primarios, tales como quirófanos, camas hospitalarias o camas UCI.

Este Trabajo de Fin de Grado aborda de forma integrada el problema de planificación de intervenciones quirúrgicas en los quirófanos y recursos postoperatorios: camas PACU, camas ICU y camas hospitalarias. En la primera etapa, el problema determina, para un conjunto de pacientes en la lista de espera, una fecha de intervención en un quirófano determinado. En la segunda etapa, las camas postoperatorias se asignarán de acuerdo con las necesidades de los pacientes intervenidos. Para encontrar la solución se van a emplear dos modelos de programación lineal entera.

El objetivo del proyecto consiste en la organización de las intervenciones quirúrgicas con la mayor brevedad posible teniendo en cuenta el peso clínico de cada uno de los pacientes e integrando los recursos postoperatorios necesarios, en este caso, camas postoperatorias. De esta manera, se priorizarán aquellas intervenciones de mayor gravedad reduciendo el retraso de cada intervención con respecto a su fecha límite y se integrará la gestión de los recursos sanitarios postoperatorios necesarios para la realización de cada una de las intervenciones, evitando así la congestión de recursos sanitarios primarios.

Establecer una adecuada planificación facilitará la posterior toma de decisiones, que mejora la gestión de los recursos sanitarios tanto en periodos excepcionales de demanda provocados por el SARS-CoV-2, como en los distintos escenarios sanitarios emergentes. Los posteriores niveles, que implican una programación en un horizonte temporal más reducido, quedan fuera del ámbito del proyecto.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Solo sé que no sé nada

- Sócrates-

Hoy en día, las organizaciones de atención médica soportan una presión creciente con el objetivo de proporcionar sus servicios al menor coste posible (H. Fei, 2010) debido a presupuestos restrictivos (I. Marques, 2012), una población que envejece (B. Roland, 2010) y una creciente lista de personas en espera (B. González-Nusto, 1999).

Los quirófanos son uno de los recursos más caros que consumen alrededor del 10% del presupuesto (José. M. Molina-Pariente, 2015) y generan más del 40% de los ingresos del hospital (J. Bowers, 2005). Gestionarlos de forma adecuada tiene un impacto esencial en la calidad y el coste de la atención médica, logrando la reducción de los costes asociados a los servicios hospitalarios.

Cuando se planifican las intervenciones quirúrgicas en quirófanos, las preferencias y restricciones establecidas por los cirujanos y resto de personal sanitario son una consideración primaria, ignorándose el impacto en los recursos posteriores (C. Price, 2011). En la mayoría de los hospitales, no se considera la disponibilidad de los recursos postoperatorios necesarios en la planificación de las intervenciones quirúrgicas. La indisponibilidad de estos recursos origina bloqueos impidiendo que los pacientes sean enviados al postoperatorio, ocasionando un impacto negativo en la gestión de los quirófanos (A. Abedini, 2017).

Por lo general, los pacientes tras la cirugía pasan a la Unidad de Recuperación Post-Anestésica (PACU) o a la Unidad de Cuidados Intensivos (ICU). Sin embargo, como las intervenciones son programadas sin tener en cuenta la disponibilidad de los recursos postoperatorios, en ocasiones, no hay disponibilidad de camas en la unidad en la que el paciente debe ser atendido tras la cirugía. Esta situación desemboca en un bloqueo de quirófanos, que significa que el paciente permanecerá en el quirófano hasta que una cama, ICU o PACU según las necesidades del paciente, quede disponible, impidiendo la realización de intervenciones quirúrgicas posteriores (C. Price, 2011). Esta situación de bloqueo produce el consumo innecesario de uno de los recursos más caros del hospital, los quirófanos, como consecuencia de la planificación y programación de intervenciones quirúrgicas realizada sin integración de los recursos postoperatorios necesarios.

La pandemia ocasionada por la COVID-19 está ocasionando un impacto en las listas de esperas en general, y en las quirúrgicas en particular. Las listas de espera en intervenciones quirúrgicas han incrementado significativamente, y la presión en las organizaciones de atención médica por proporcionar un servicio de calidad a menor coste es aún mayor. Por otro lado, la COVID-19 ha producido una paralización prácticamente total de la atención primaria, diagnóstica y especializada, a lo que cabe añadir que lo enfermos no han querido asistir a los hospitales por miedo al contagio, aguantando síntomas agudos y dolores en su domicilio, agravando la enfermedad que padecían. Estas circunstancias, también impactan de forma directa sobre las listas de espera de atención especializada y éstas, sin duda, sobre las quirúrgicas (L. de Pablos Escobar, 2021). Sin embargo, no todos los recursos quirúrgicos disponibles tienen la misma importancia ni influyen de la misma manera en la reducción de las personas en espera. Los recursos más significativos son las camas y los quirófanos (L. de Pablos Escobar, 2021).

La saturación de camas debido a la crisis sanitaria provocada por la COVID-19, en concreto en las camas de la Unidad de Cuidados Intensivos (ICU) y las camas hospitalarias, está ocasionando un bloqueo en las listas de espera. Al no haber disponibilidad de recursos postoperatorios, es decir, de camas postoperatorias, los hospitales no están planificando cirugías y se está retrasando el tratamiento de pacientes con gran prioridad médica. Sin embargo, algunas de esas cirugías que no se están planificando, no requieren recursos postoperatorios. Por ello, es esencial abordar el problema de la planificación de quirófanos integrando la disponibilidad de los recursos postoperatorios. De esta manera, si una cirugía no requiere recursos

postoperatorios, podrá ser programada a pesar de la saturación de recursos empleados en el postoperatorio ocasionada por la COVID-19.

Así, este Trabajo de Fin de Grado se basa en la aplicación de los métodos de planificación de la producción en el sistema sanitario para la elaboración de la planificación de intervenciones quirúrgicas y recursos postoperatorios. Es por ello, que son de conocimiento necesario los siguientes epígrafes expuestos para su elaboración.

1.1 Gestión de la producción

Este proyecto se enmarca en la gestión de la producción, que es el conjunto de toma de decisiones necesarias para acometer el proceso de la producción de forma adecuada, considerándose la planificación de la producción, el control de la producción, la secuenciación y la gestión de inventarios entre otros. Su objetivo es garantizar la entrega de un bien con la máxima calidad, al mínimo coste y plazo de entrega.

La toma de decisiones tiene gran impacto en la empresa y su objetivo. Dependiendo de la naturaleza de la decisión y el periodo en el que se tomen, se descomponen adoptando una estructura jerárquica en tres niveles según el impacto que provocan: Decisiones estratégicas, decisiones tácticas y decisiones operativas.

Figura 1-1: Niveles de toma de decisiones y su impacto. (Miranda, 2018)

En el nivel estratégico o planificación estratégica, la gerencia de la empresa toma decisiones a largo plazo relacionadas con el diseño estratégico de la estructura productiva, objetivos de la organización, recursos necesarios para implementar los planes de producción, selección de proveedores, localización y diseño de las plantas de producción (Jose M. Framiñan, 2014). Estas decisiones establecen, en un nivel agregado (usando periodos mensuales, por ejemplo), la capacidad de producción y la distribución de los recursos fijos de forma coordinada con los proveedores. Tienen un gran impacto en la empresa, su economía y sus resultados, y se basan en pronósticos del entorno (evolución de mercado y recursos fijos de la empresa). Los recursos físicos involucrados hacen que sea poco probable o incluso imposible la revisión de la planificación estratégica en un horizonte temporal inferior al año. Una vez diseñada la estructura de fabricación y distribución, se toman decisiones relativas a cómo hacer el mejor uso de esta estructura para que las acciones establecidas en la planificación estratégica sean alcanzadas (Jose M. Framiñan, 2014). Estas decisiones se descomponen a su vez, debido a la complejidad del proceso de fabricación y la diversidad y calidad de la información requerida para la toma de decisiones, en decisiones a medio y corto plazo: nivel táctico y nivel operativo respectivamente.

En el nivel táctico, se asocian de forma individual los productos a realizar a recursos productivos en un horizonte temporal menor. El resultado de este proceso es, idealmente, un plan en el que se especifica qué trabajo debe ser realizado por cada recurso y cuándo. Al tratarse de una planificación más específica, en este nivel de decisión se involucran los cargos entre la alta dirección y el nivel operacional. A pesar de su mayor grado de detalle, raramente la planificación de la producción se puede implementar tal y como fue diseñada debido a cambios de última hora e imprevistos (avería de maquinaria, absentismo de trabajadores, nuevas órdenes urgentes, cancelación de pedidos existentes, etc.) que deben implementarse (McClain, 1993). Por ello, los planes de producción deben ser reprogramados en el nivel operativo.

El nivel operativo, consiste en la realización de un nuevo cronograma más detallado que en el nivel anterior,

donde se contemplan los cambios previstos con respecto a la planificación realizada en el nivel táctico y, a menudo, se toman decisiones reactivas que tendrán impacto sobre las operaciones diarias, adaptando la planificación a la situación correspondiente. Todos los niveles de la organización están involucrados en este nivel, y deben garantizar que todas las tareas y operaciones programadas se ejecuten de acuerdo con lo establecido para alcanzar los objetivos inicialmente previstos.

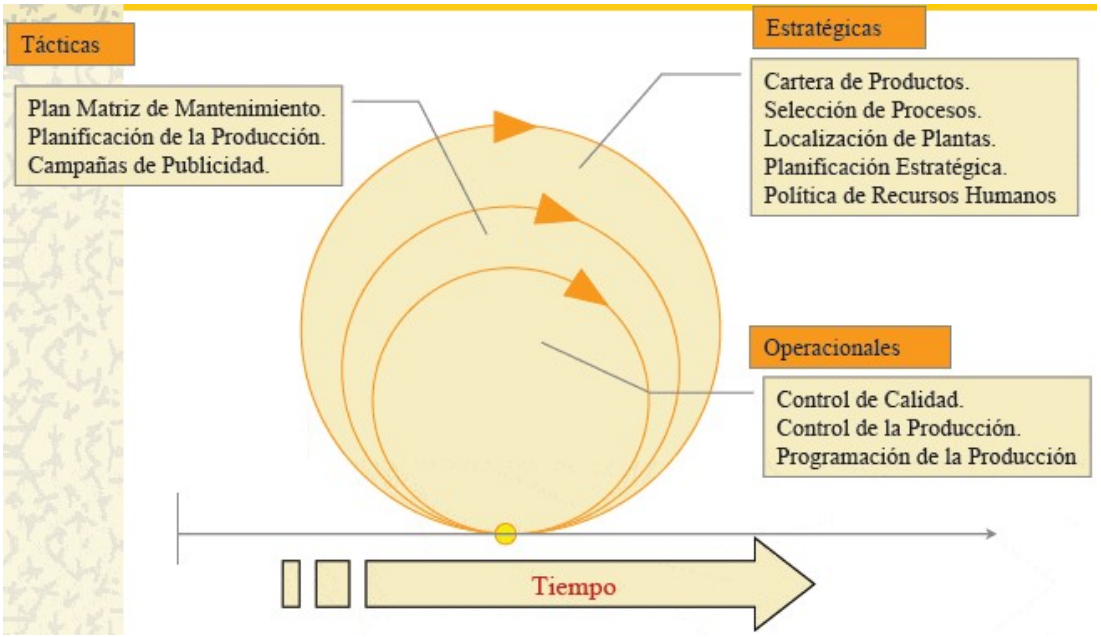


Figura 1-2: Nivel de decisión en función del tiempo. (Tutoriales, 12)

La diferencia entre la planificación realizada en el nivel táctico y la programación realizada en el nivel operativo radica básicamente en lo objetivos perseguidos en cada nivel. Si bien la planificación tiene como objetivo mejorar el rendimiento de la producción, programar tiene como objetivo minimizar perturbaciones en la producción debido a la llegada de eventos inesperados (Jose M. Framiñan, 2014).

Para lograr los objetivos establecidos en la planificación estratégica, es necesario establecer de forma correcta los planes de los niveles táctico y operativo, pues es un proceso integrado e interdependiente donde todos los niveles son necesarios. Por ello, las planificaciones deben involucrar a todo el personal de la empresa, trabajando de forma coordinada hacia un objetivo común.

1.2 Niveles de decisión en el ámbito sanitario

En el ámbito de la gestión de quirófanos, las decisiones de descomponen en tres niveles jerárquicos: Decisiones estratégicas, decisiones tácticas y decisiones operativas (B. Cardoen E. D., 2010).

En el nivel estratégico, las variables de decisión son el número y tipo de cirugías que se planifican en el horizonte de planificación y el número de recursos requeridos. En el nivel táctico, los recursos de quirófano se asignan en un horizonte de planificación de varias semanas estableciendo un programa quirúrgico maestro. En el nivel operativo, se establecerá la fecha de la cirugía, el quirófano asignado, y el tiempo indicado para la realización de cada cirugía (José. M. Molina-Pariente, 2015).

El nivel operativo, está compuesto por dos niveles: el nivel *off-line* y el *on-line* (José M. Molina-Pariente, 2015). El nivel operativo *off-line* se resuelve en dos pasos, en el primer paso, se determina el día y el quirófano asignado a cada cirugía, y en el segundo paso, se obtiene la lista de cirugías para cada quirófano en cada día (José. M. Molina-Pariente, 2015). El nivel *on-line* operativo implica mecanismos de control para reaccionar ante posibles imprevistos, como discrepancias entere la duración prevista y la duración real de las cirugías (José M. Molina-Pariente, 2015).

1.3 Introducción a los modelos de planificación

La planificación es un proceso de toma de decisiones donde se obtiene un modelo formal con el objetivo de resolver un problema real. Un modelo es una representación simplificada y abstracta de una amplia realidad compleja y detallada. En él, se modelan todos los recursos productivos relevantes como máquinas, personal de trabajo, herramientas y áreas de almacenaje considerando el sistema de tareas/operaciones, las restricciones del proceso y los criterios de decisión (Jose M. Framiñan, 2014).

La resolución de problemas de programación/planificación reales se lleva a cabo en tres pasos: primero modelándolos (de ahí la necesidad de modelos), segundo encontrando métodos que brinden soluciones para estos modelos, y no para el problema del mundo real, y tercero transfiriendo estas soluciones a el mundo real (Jose M. Framiñan, 2014).

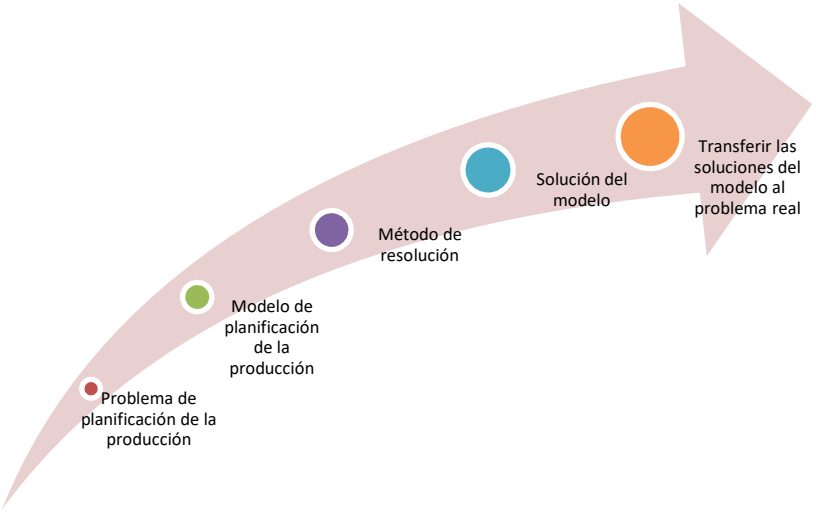


Figura 1-3: Procedimiento de resolución de problemas de planificación (Elaboración propia)

Los modelos tienen las siguientes características y beneficios (Jose M. Framiñan, 2014):

- Enfocados en determinados aspectos de la realidad.
- Siempre relacionados con la realidad que se modela. El modelo debe aproximarse lo máximo posible a la realidad para lograr su objetivo previsto.
- Un modelo es diferente de la realidad, es una simplificación. Si el modelo representa la realidad de manera exacta, no es un modelo sino una copia.
- Los modelos, o mejor, el resultado obtenido de la experimentación del modelo, pueden ser medidos con facilidad.
- Pueden ser empleados para el estudio de realidades alternativas. Una vez validado el modelo para la realidad presente, pueden emplearse modelos alterados para estudiar escenarios en diferentes realidades, con un grado de confianza, que de otra forma sería imposible estudiar.
- Una vez validados los modelos, pueden explicar eventos pasados, predecir observaciones futuras y/o aprobar o rechazar hipótesis sobre la realidad modelada.
- Deben tener un fácil manejo. La realidad se modela debido a su complejidad. Un modelo difícil de interpretar o trabajar con él, no es un modelo válido.
- Los modelos están llenos de limitaciones. La interpretación de los resultados del modelo es válida si se lleva a cabo de manera científica. El modelo tiene que ser validado y los resultados examinados minuciosamente.

Para la resolución de problemas de planificación/programación, se emplean los siguientes tipos de modelos o metodologías (Jose M. Framiñan, 2014):

- Modelos matemáticos
- Modelos gráficos

- Modelos estadísticos
- Modelos de simulación
- Modelos de algoritmos

En este proyecto, se va a abordar la resolución del problema de planificación mediante un modelo de algoritmos tal y como se presenta en la sección 1.5.

1.3.1 Datos básicos del modelo

A continuación se definen una serie de términos de necesario conocimiento para la elaboración de los modelos.

- Máquinas: Número finito de máquinas conocido.
- Trabajos: Número finito de trabajos a procesar conocido.
- Operación o actividad: Cada trabajo tiene un número predeterminado de operaciones o actividades, llevadas a cabo en al menos una máquina cada una de ellas.
- Ruta de proceso: Cada trabajo tiene una ruta predefinida a través de la planta de producción.
- Tiempo de proceso: Tiempo requerido por la máquina *i* para procesar el trabajo *j*.
- Fecha de llegada *(release date)*: Instante de tiempo a partir del cual un trabajo puede ser procesado.
- Fecha de entrega *(due date)*: Instante de tiempo en el que un trabajo debe estar terminado, es decir, en el que un paciente debe haber sido intervenido.
- Peso: Prioridad que establece una jerarquía relativa entre el conjunto de trabajos.

1.3.2 Clasificación de los modelos

En la literatura se clasifican los modelos de planificación de la producción de acuerdo a sus características principales: objeto de flujo, constantes del proceso y medidas de rendimiento. Algunos autores proponen para los modelos una clasificación basada en tres elementos (Jose M. Framiñan, 2014): α, β, γ.

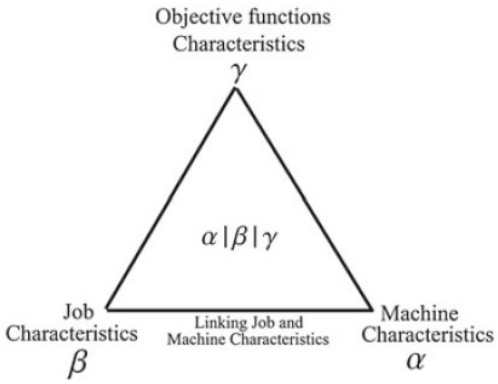


Figura 1-4: Clasificación de los modelos (Jose M. Framiñan, 2014)

El parámetro α define el entorno del proceso: características del proceso productivo y número de máquinas. Por otro lado, β define las restricciones del proceso. Finalmente, γ proporciona información sobre la función objetivo del problema de estudio.

1.3.2.1 Entorno del proceso (α)

A lo largo de la literatura de programación de la producción, el entorno del proceso está definido según la disposición de las máquinas y las especificaciones de las diferentes rutas de procesamiento de trabajo (Jose M. Framiñan, 2014). Las principales disposiciones son:

- Una máquina (*Single Machine*)

- Máquinas en paralelo (*Parallel Machine*): Compuesto por varias máquinas individuales trabajando en paralelo. Pueden ser máquinas idénticas, uniformes (con distintas velocidades) o diferentes entre sí, siendo este último caso el más común.
- Taller de flujo (*Flow Shop*): Compuesto por un conjunto de máquinas individuales que trabajan en serie. Cada trabajo debe ser desempeñado por todas y cada una de las máquinas en un orden especificado, es decir, todos siguen la misma ruta.
- Taller de trabajo (*Job Shop*): Compuesto por un conjunto de máquinas dispuestas en serie y, en las que a diferencia del *Flow Shop*, la ruta a seguir dependerá del trabajo realizado.
- Taller abierto (*Open Shop*): Compuesto por un conjunto de máquinas dispuestas en serie en las que los trabajos pueden ser ejecutados de forma arbitraria sin seguir un orden especificado.
- Híbridos (*Hybrid*): Compuesto por etapas de producción, donde en cada una de ellas hay un número, que puede ser distinto, de máquinas dispuestas en paralelo. Los trabajos van de etapa en etapa y todas las rutas son posibles.

1.3.2.2 Restricciones del problema (β)

Existen multitud de restricciones que limitan la planificación de la producción. En el sector industrial, se clasifican generalmente en cuatro grupos: restricciones del proceso, operaciones, transporte y almacenaje. Estas restricciones pueden ser extrapoladas al sector sanitario. Algunas de las restricciones más empleadas son:

1.3.2.2.1 Restricciones del proceso

- Elegibilidad de la máquina: En los sistemas híbridos con máquinas en paralelo se asume que cualquier máquina es capaz de desempeñar cualquier trabajo. Sin embargo, esto no siempre ocurre.
- Disponibilidad de las máquinas: Se asume que las máquinas están continuamente disponibles en el horizonte temporal.
- Tiempo de inactividad: Factor no deseable especialmente en recursos caros.

1.3.2.2.2 Restricciones de las operaciones o actividades

- Interrupción de los trabajos: Una vez que se comienza un trabajo, en este caso una intervención quirúrgica, no puede ser interrumpida y debe ser procesada hasta completarse.
- Fecha de llegada (*release date*) y fecha de entrega (*due date*).
- Solapamiento de actividades
- Tiempo especial de proceso: Los tiempos de proceso se suponen conocidos y fijados de antemano. Sin embargo, hay muchas situaciones que podrían afectar a esta suposición, que por lo demás, sería sólida.

1.3.2.2.3 Restricciones de almacenamiento

La capacidad de almacenaje de la planta de producción debe ser considerada.

1.3.2.3 Función objetivo (γ)

En la gestión de la producción, los objetivos se suelen clasificar en: costes, tiempo, calidad y flexibilidad (Jose M. Framiñan, 2014). Estas categorías son interdependientes, y aunque en ocasiones se empleen en paralelo, esto suele resultar conflictivo.

Los objetivos empleados en los modelos de planificación de la producción están relacionados con la optimización de algún indicador. Estos indicadores, se denominan medida del rendimiento y algunos de los más importantes en función del tiempo son (Jose M. Framiñan, 2014):

- Tiempo de finalización del trabajo j o *Completion time of job j* (C_j): Instante de tiempo en el que el trabajo j termina.
- Tiempo de flujo del trabajo j o *Flow time of job j* (F_j): Periodo de tiempo en el que el trabajo permanece en el taller, ya sea siendo procesado o en espera.
- Retraso del trabajo j o *Lateness* (L_j): Retraso del trabajo j con respecto a su fecha de entrega.

- Tardanza del trabajo j o *Tardiness* (T_j): Indicador que toma valor positivo únicamente para trabajos terminados posteriormente a su fecha de entrega.
- Adelanto del trabajo j o *Earliness* (E_j): Similar al *tardiness*. Se calcula únicamente para los trabajos adelantados, siendo la diferencia (negativa) entre la fecha de finalización y fecha de entrega.
- Trabajo tarde o *Tardy job* (U_j): Tomará el valor 1 si el trabajo finaliza tarde y 0 en caso contrario.
- Trabajo temprano o *Early job* (U_j): Tomará el valor 1 si el trabajo finaliza temprano y 0 en caso contrario.
- Adelanto-Tardanza del trabajo j o *Earliness-Tardiness of job j* (ET_j): Suma del *tardiness* y *earliness* del trabajo j.
- Trabajo justo a tiempo o *Just-In-Time job* (JIT_j): Tomará el valor 1 si el trabajo j no finaliza ni tarde ni temprano, es decir, justo a tiempo, y 0 en caso contrario.

Considerando las similitudes entre ambos sistemas estudiadas anteriormente, los indicadores mencionados con anterioridad son fácilmente extrapolables a la planificación de quirófanos y camas postoperatorias.

1.4 Validación y verificación de los modelos

Todos aquellos modelos teóricos que representen un problema del entorno real, deben ser validados y verificados.

La verificación de un modelo prueba si la formulación del modelo (y su procedimiento de resolución) cumple los requisitos del modelo dentro de la esfera del modelo (Jose M. Framiñan, 2014). Se realiza con el objetivo de verificar que el modelo y su proceso de resolución actúan según lo requerido. Para ello, es necesario comprobar que el diseño del modelo teórico y la implementación de los algoritmos ha sido realizada de manera correcta.

La validación del modelo es complementaria a la verificación y comprueba la adecuación del modelo con respecto al problema real. Garantiza que el modelo cumple los requisitos previstos en términos de métodos empleados y resultados obtenidos en relación con el entorno real (Jose M. Framiñan, 2014).

La validación y verificación del modelo debe abordarse de igual manera en el entorno sanitario, garantizando que la formulación de modelo teórico realizada cumple con los requerimientos y es aplicable al problema de planificación de intervenciones quirúrgicas y camas postoperatorias real. Contiene alguno de los siguientes aspectos (Jose M. Framiñan, 2014):

- Las restricciones del modelo deben ser flexibles, no redundantes y acordes al problema real.
- La función objetivo debe ser acorde a los objetivos del problema real.
- El espacio de decisión del modelo debe estar en línea con las decisiones reales.
- El modelo formulado no debe contener errores.
- La influencia subjetiva del encargado de la toma de decisiones debe ser minimizada o en la línea con las condiciones reales del ámbito de aplicación del problema.

1.5 Presentación de los posibles métodos de resolución

Un método de resolución es un procedimiento formal aplicable a cualquier instancia de un modelo de programación/planificación con el objetivo de obtener una solución factible que (probablemente) obtenga buenos resultados con respecto al objetivo buscado (Jose M. Framiñan, 2014). Toma una instancia de un modelo de programación como entrada para producir (al menos) una programación como salida.

El procedimiento se puede describir usando un número finito de pasos, y por lo tanto, puede ser codificado y ejecutado por una computadora. El nombre técnico de este procedimiento finito es *algoritmo*.

1.5.1 Algoritmos de programación

Los algoritmos de programación se clasifican en algoritmos exactos y algoritmos aproximados. Los algoritmos exactos son aquellos que garantizan que la solución obtenida para el modelo es la solución óptima. Sin embargo, los algoritmos aproximados no presentan esta garantía.

1.5.1.1 Algoritmos exactos

En los problemas de programación, los algoritmos exactos se clasifican en:

- Algoritmos constructivos exactos (polinomiales): Son algoritmos que explotan algunas propiedades de un modelo específico para construir una solución óptima garantizada. Estos procedimientos constructivos son simples y manejables a nivel computacional, sin embargo, cubren solo un espectro muy limitado de problemas bastante simples. Por ello, se pueden usar como submódulos en la programación de problemas más complejos.
- Algoritmos enumerativos (no polinomiales): Aquellos algoritmos que garantizan, de forma implícita o explícita, la evaluación de todas las posibles soluciones del modelo. Incluyen métodos como el *branch and bound*, la enumeración completa y el método de plano de corte. Este tipo de algoritmo proporciona una solución óptima exacta de manera segura y cómoda, sin embargo, el esfuerzo computacional crece de manera exponencial con el tamaño del problema, impidiendo en los problemas de grandes dimensiones la resolución.

1.5.1.2 Algoritmos aproximados

Aquellos que proporcionan una solución de calidad al problema de optimización, no garantizando la optimalidad de la solución obtenida. Los algoritmos aproximados se clasifican en: heurísticas (heurísticas constructivas y heurísticas de mejora) y metaheurísticas.

1.5.2 Método de resolución: Programación lineal entera (ILP)

La programación entera es un algoritmo exacto enumerativo que puede ser considerado como un tipo particular de *branch and bound* y formularse como programa entero o binario. Para este Trabajo de Fin de Grado se ha elegido el método de programación lineal entera. Consiste fundamentalmente en dos pasos:

1. Formulación del modelo de programación en términos de programación lineal entera (ILP).
2. Aplicación del software de programación entera (IP) al modelo de programación lineal entera (ILP) generado en el paso anterior.

Si la formulación del modelo de programación lineal entera (ILP) ha sido realizada en el primer paso de forma inteligente, el esfuerzo computacional se reducirá significativamente en el segundo paso.

1.6 Evaluación de los métodos

La evaluación del rendimiento de un método de programación/planificación se puede lograr de (al menos) dos maneras (Jose M. Framiñan, 2014):

- Comprobando la “adecuación formal”.
- Comprobando la “adecuación del entorno real”.

1.6.1 Adecuación del entorno real

Evalúa la contribución del método desarrollado al problema real, es decir, cómo de bien se ajusta el modelo teórico propuesto a la resolución del problema real. Será adecuada siempre que la solución implementada cumpla con los requerimientos del encargado de la toma de decisiones y/o los requerimientos del problema real (Jose M. Framiñan, 2014).

1.6.2 Adecuación formal

Evalúa la calidad de la solución ofrecida por el modelo teórico dentro del marco teórico (Jose M. Framiñan, 2014). Para ello, se suelen emplear al menos dos indicadores: el valor de la función objetivo y el tiempo de ejecución, que realizan una evaluación multidimensional de los algoritmos. Generalmente, la evaluación se puede realizar de dos maneras (Jose M. Framiñan, 2014):

- Mediante un indicador de desempeño
- Comparando dos o más algoritmos mediante una prueba formal o un estudio de simulación.

En este Trabajo de Fin de Grado vamos a evaluar la adecuación formal de los métodos propuestos mediante un estudio de simulación, por lo que vamos a desarrollar en qué consiste en la siguiente subsección.

1.6.2.1 Estudio de simulación

Es el método más extendido para la evaluación del desempeño de los algoritmos. Emplea un conjunto de instancias de prueba para la evaluación del método correspondiente. Puede resumirse en dos pasos (Jose M. Framiñan, 2014):

1. Generación del banco de pruebas (*testbed*): Consiste en la generación de un conjunto de instancias, ya sea de forma específica para el modelo o adaptando las empleadas para otros modelos existentes. La adaptación puede realizarse de dos maneras: referida únicamente al método de generar instancias de prueba mientras que las instancias mismas se generan de nuevo, o bien referida a instancias que se toman directamente de referencias o bases de datos públicas.
2. Análisis de los algoritmos: Basado en la ejecución en cada instancia de los algoritmos a comparar, empleando métricas o análisis para establecer el rendimiento relativo de cada algoritmo en el banco de pruebas. De este modo, si un algoritmo supera a otro en el banco de pruebas, su rendimiento es mejor.

Para la generación del banco de pruebas que se realizará en la sección 7.2, se deben considerar previamente los siguientes aspectos (Jose M. Framiñan, 2014):

- El tamaño de las instancias del problema debe ser acorde a las dimensiones del problema real y/o a las dimensiones de las instancias empleadas en estudios similares.
- El número de instancias del problema generadas debe ser suficiente para obtener resultados útiles.
- Los principales parámetros del modelo deben ser seleccionados de forma adecuada en función del objetivo perseguido y basados en la observación del problema real.
- Cuando se amplía/adapta un banco de pruebas generado para otro modelos existente, deben considerarse las características adicionales del modelo en cuestión.

1.7 Ámbito de aplicación del proyecto

El Trabajo de Fin de Grado se enmarca en la Unidad de Gestión Clínica de Cirugía Plástica y Grandes Quemados del Hospital Universitario Virgen del Rocío (Sevilla, España). El Hospital Universitario Virgen del Rocío es uno de los hospitales referentes a nivel nacional. Cuenta con más de 8.000 profesionales, 54 quirófanos y 1.291 camas instaladas. Dada su relevante importancia en el panorama nacional, la correcta planificación y gestión de sus recursos es necesaria.

De forma más concreta, en la Unidad de Gestión Clínica de Cirugía Plástica y Grandes Quemados, se realizan alrededor de 3.000 cirugías de forma anual (José M. Molina-Pariente, 2015). La planificación de las cirugías en esta unidad de gestión clínica se realiza sin el apoyo de ninguna herramienta de soporte a la toma de decisiones basadas en la optimización, basada únicamente en la experiencia del responsable de la toma de decisiones. Esto conduce a un bajo rendimiento en términos de tiempo de acceso de los pacientes y una utilización ineficiente de los quirófanos (José M. Molina-Pariente, 2015).

El objetivo de este Trabajo de Fin de Grado es proporcionar a través de un modelo de programación lineal entera, una herramienta administrativa que soporte las decisiones de planificación de cirugías integrando los

recursos postoperatorios necesarios en el nivel operativo *off-line*. Esta herramienta proporcionará la lista de intervenciones a realizar en un horizonte de planificación, en el orden adecuado, teniendo en cuenta la prioridad médica de cada uno de los pacientes, el tiempo de pacientes en la lista de espera y la disponibilidad de los recursos necesarios tras las intervenciones.

El trabajo está organizado de la siguiente forma. En la sección 2 se contextualiza la naturaleza del problema de decisión dentro del marco de gestión de operaciones. A través de una comparación entre la gestión de la producción en el sistema industrial y la gestión de operaciones en el sistema sanitario, se concluye que lo métodos existentes para el primero pueden ser aplicados de forma similar al segundo. La complejidad de los modelos se define en la sección 3. En las secciones 4 y 5 se abordará de manera respectiva la revisión de la literatura y la descripción del problema abordado. La adaptación e implementación de la metodología empleada para resolver el problema de mediante programación lineal entera (ILP) aparece descrita en la sección 6, y la implementación de los modelos y los resultados experimentales obtenidos se presentan en la sección 7. Finalmente, en la sección 8 se presentan las conclusiones y se plantean posibles líneas de futuro.

En último lugar, se adjunta el anexo, que contiene las funciones desarrolladas en código C# para la resolución del problema abordado en este Trabajo de Fin de Grado.

2 COMPARACIÓN DE LA GESTIÓN DE OPERACIONES ENTRE EL SISTEMA INDUSTRIAL Y EL SANITARIO

El objetivo de este capítulo es enmarcar el problema de decisión referente a la planificación de quirófanos y camas postoperatorias dentro de la gestión de operaciones. Para ello, se realizará una comparación entre el sistema industrial y sanitario, donde se justificará la posible adaptación de los métodos estudiados para la planificación de la producción en el ámbito de la salud.

2.1 Gestión de operaciones en el ámbito sanitario

La gestión de la producción/operaciones es el proceso que combina y transforma los recursos utilizados en los subsistemas de producción de la organización en productos/servicios con valor añadido (S.A. Kumar, 2006). Esta definición es aplicable a la gestión de operaciones en el ámbito sanitario, donde se transforman y combinan los recursos utilizados en servicios con valor añadido, es decir, servicios sanitarios. Considerando de forma individual la consulta paciente/doctor, la entrada al proceso es un paciente que requiere atención sanitaria, y la salida, el diagnóstico del paciente, enviado a un proceso sanitario posterior, o curado. Los recursos que tienen que ser gestionados para transformar las entradas en salidas son aquellos asociados a la atención médica proporcionada por el doctor: por ejemplo, su tiempo, cualquier diagnóstico o cualquier servicio terapéutico empleado (J. Vissers, 2005).

En el ejemplo anterior, el proveedor 'individual' es el doctor. Sin embargo, el proveedor 'individual' puede ser, por ejemplo, un departamento del hospital (el departamento de rayos, por ejemplo), un hospital, una red de hospitales o una comunidad de servicios (J. Vissers, 2005).

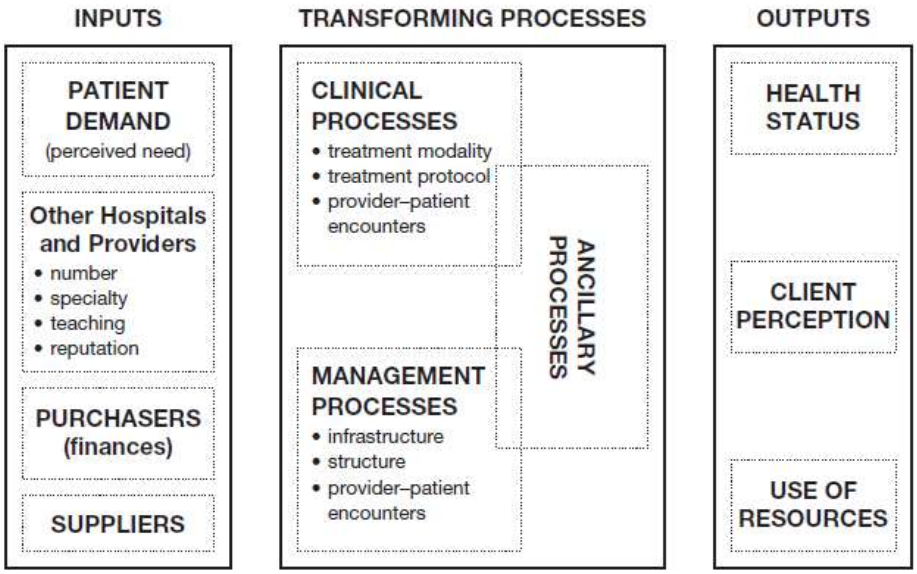


Figura 2-1: Proceso de prestación de atención médica en un hospital (J. Vissers, 2005)

Existen tres procesos genéricos para transformar las entradas en salidas: proceso clínico, proceso de gestión y procesos auxiliares. Si bien los procesos clínicos son los más importantes para llevar a cabo el diagnóstico y tratamiento de los pacientes, los procesos de gestión y los auxiliares son necesarios como soporte (aprovisionamiento de material médico, planificación de los servicios de limpieza).

En la figura anterior, podemos observar cómo el proceso de prestación de atención médica en un hospital está

formado por entradas, procesos de transformación y salidas al igual que el proceso de producción en un sistema industrial. Luego sí, existe similitud entre ambos entornos.

2.2 Similitudes y diferencias entre la gestión de operaciones en el sistema industrial y en el sanitario

En el sistema industrial, la revolución en tecnologías informáticas y la creciente competencia entre las distintas empresas, ha acelerado en la innovación de productos. La competitividad ocasiona una alta presión en el rendimiento empresarial en términos de calidad, eficiencia y flexibilidad con el objetivo de reducir los tiempos de entrega y los costes y aumentar el rendimiento, los ingresos y las ganancias.

El sistema sanitario se enfrenta a desafíos similares, como (J. Vissers, 2005):

- Aumento de la complejidad de los procesos debido a estancias más cortas de los pacientes, intercambio de tratamientos, nuevas tecnologías y mayor especialización.
- Utilización eficiente de los recursos y disminución de los costes: Primero, porque el tratamiento se realiza en un periodo de tiempo muy reducido, y segundo, por la presión política que genera el control de gastos en el Sistema Nacional de Salud.
- Aumento de presión con el objetivo de mejorar la calidad de los servicios, disminuyendo las listas de espera y los tiempos de espera en procesos.
- Controlar la carga de trabajo del personal sanitario para evitar efectos adversos en sus condiciones laborales.

Sin embargo, presenta las siguientes diferencias con respecto al sistema industrial (J. Vissers, 2005):

- En el sistema industrial, el control de la producción se centra en el flujo de material. En el sistema sanitario, se centra en el flujo de pacientes que necesitan tratamiento, mientras que el flujo de materiales pasa a ser secundario.
- En el sistema sanitario, es mucho menor la iteración precio-rendimiento, mientras que en los entornos productivos está siempre presente.
- Los métodos de control de la producción requieren especificaciones completas y explícitas acerca del producto final y sus requisitos de entrega, mientras que, en el sistema sanitario, las especificaciones de los productos son subjetivas e imprecisas.
- En las organizaciones de atención médica no existe una única línea de mando, se caracterizan por un delicado equilibrio de poder entre los distintos grupos de interés (Médicos, enfermeros, paramédicos, jefes de servicio), donde cada tiene su propia idea acerca del objetivo de la organización.
- El personal clave del proceso son profesionales altamente cualificados (médicos especialistas), que generan las peticiones de servicio y se involucran en la entrega de este.
- La asistencia médica no es una mercancía que pueda ser almacenada. El hospital es una organización dedicada a la prestación de servicios.

Estas diferencias y similitudes entre el sistema industrial y el sanitario se recogen de forma resumida en la tabla 2-1.

Tabla 2-1: Similitudes y diferencias entre el sistema industrial y el sanitario (J. Vissers, 2005)

Características	Sistema Industrial	Sistema Sanitario
Objeto de flujo	Material	Paciente
Requisitos del producto terminado	Especificado de antemano	Subjetivos y difusos
Medios de producción	Equipamiento y plantilla	Equipamiento y plantilla
Buffers	Stocks o plazos de entrega	Listas de espera y plazos de atención
Objetivo final	Beneficio	Control de coste
Entorno de mercado	Mercado competitivo	Mercado competitivo limitado

Teniendo en cuenta las diferencias y similitudes, podemos definir el control de la producción en las organizaciones de atención médica como:

> El diseño, planificación, implementación y control de mecanismos coordinados entre flujo de pacientes y actividades terapéuticas y diagnósticas en los servicios de atención médica para maximizar el rendimiento con los recursos disponibles, teniendo en cuenta los diferentes requisitos en la flexibilidad de entrega (electiva/cita, semi-urgente, urgente), las normas aceptables en cuanto a fiabilidad de la entrega (listas de espera, tiempos de espera) y los resultados médicos aceptables.
>
> (Vissers, 1994)

3 COMPLEJIDAD

En teoría de sistemas, la complejidad de un sistema viene dado por el número y la diversidad de objetos en el sistema, así como por el número y la diversidad de las relaciones entre los objetos (Jose M. Framiñan, 2014).

El alcance del proyecto abordado en este Trabajo de Fin de Grado, está limitado por la complejidad de la metodología de resolución propuesta. Para la determinación de la complejidad, es necesario entender previamente cómo se aborda la complejidad en la planificación de la producción y las bases en las que se sustenta.

En la planificación/programación de la producción, la complejidad puede abordarse desde dos perspectivas: Restringida en el ámbito teórico (*complejidad computacional*) o bien, de manera más general y luego extenderse a la aplicación en el ámbito real (*complejidad de entornos reales*) (Jose M. Framiñan, 2014). Estos dos tipos de complejidad se desarrollan en las siguientes subsecciones.

3.1 Complejidad computacional

Es la complejidad del problema teórico y los algoritmos desarrollados para su resolución. De manera informal, podemos distinguir dos tipos de complejidad computacional (Jose M. Framiñan, 2014):

- *Complejidad del algoritmo*: Estimación computacional (tiempo de ejecución) sobre el desempeño de un algoritmo específico para la resolución de un modelo. Dependerá, entre otras cosas, de la instancia específica del modelo para la que se aplica el algoritmo. Aplicable a cualquier algoritmo, ya sea exacto o aproximado.
- *Complejidad del modelo*: Proporciona la complejidad del "mejor algoritmo" capaz de encontrar la solución óptima del modelo. Referida de forma exclusiva al mejor algoritmo del modelo.

A grandes rasgos, este concepto de complejidad sirve para separar los algoritmos cuyo esfuerzo computacional puede estar acotado por alguna función polinomial, de aquellos en los que dicho límite polinomial aún no se ha derivado y donde probablemente nunca se encontrará (Jose M. Framiñan, 2014). De forma más detallada:

- *Algoritmos polinomiales*: Se pueden lograr con un esfuerzo computacional razonable.
- *Algoritmos no polinomiales*: Realizan una enumeración explícita o implícita de cada solución factible. De manera teórica, no se pueden lograr con esfuerzo computacional razonable al no conocerse el límite polinomial. Por el contrario, desde el punto de vista del ámbito real, podría alcanzar una buena o incluso la solución óptima en un tiempo razonable. Sin embargo, la prueba de esta optimización requiere mucho tiempo.

Teniendo en cuenta que para el desarrollo de este Trabajo de Fin de Grado se ha elegido un método de programación lineal entera que realiza una enumeración de cada solución factible, la naturaleza del modelo matemático desarrollado es no polinómica.

3.2 Complejidad en entornos reales

Los problemas de decisión del mundo real a menudo incluyen la complejidad impuesta por un marco de decisión complejo. Al igual que en el entorno industrial, esto ocurre en el entorno sanitario, donde intervienen, entre otros (Jose M. Framiñan, 2014):

- Objetivos diferentes, en ocasiones no claramente especificados.
- Gran variedad de restricciones, no pre-establecidas en ocasiones.

- Gran variedad de posibles acciones, no especificadas de manera clara en ocasiones.
- Sistema jerárquico de planificación, toma de decisiones y encargado de toma de decisiones, en ocasiones, no especificado o claramente definido.
- Complejidad inducida por la dinámica y la incertidumbre.
- Iteración de todos los aspectos anteriores, dentro de su categoría y entre las categorías.

Los aspectos de la complejidad se clasifican en (Jose M. Framiñan, 2014):

- *Aspectos de masa*:
  - *Multiplicidad*: número de elementos e iteraciones en el sistema
  - *Variedad*: número de elementos e iteraciones diferentes en el sistema.
- *Aspectos del caos:*
  - *Ambigüedad:* Grado de incertidumbre de las características de los elementos e iteraciones en el sistema.
  - *Variabilidad:* Variación de las características de los elementos e iteraciones a lo largo del tiempo (relacionado con la dinámica).

Los aspectos de la complejidad en entornos reales se recogen de forma resumida en la tabla 2.

Tabla 3-1: Clasificación de la complejidad en entornos reales (Jose M. Framiñan, 2014)

Aspectos de la complejidad		**Factores de complejidad en programación de la producción**
Aspectos de masa	Multiplicidad	Número de trabajos, escenarios, etc.
	Variedad	Diversidad de tiempos de procesamiento, número/tipo de operaciones requeridas, etc.
Aspectos del caos	Ambigüedad	Incertidumbre sobre los tiempos de proceso, fechas de entrada, etc.
	Variabilidad	Llegada de nuevos trabajos, averías de maquinaria, etc.

Esta clasificación es aplicable a la complejidad existente en el entorno sanitario, en concreto en la planificación de intervenciones quirúrgicas y camas postoperatorias. En ella encontraremos aspectos de masa, como el número de intervenciones quirúrgicas, número de quirófanos y camas postoperatorias, número de cirujanos (*multiplicidad*) y diversidad de tiempos de intervención y recuperación en las distintas camas postoperatorias, tipo de camas postoperatorias requeridas (*variedad*). En cuanto a los aspectos del caos, la planificación realizada se caracteriza por la incertidumbre en la duración de las intervenciones y recuperación de los pacientes tras la intervención, así como en la fecha de entrada, entre otros (*ambigüedad*). Está sometida a una constante variación debida a la llegada de nuevas intervenciones quirúrgicas, en ocasiones inesperadas, imprevistos en la utilización de recursos perioperatorios, en la disponibilidad del cirujano y/o cama postoperatoria asignada, etc. (*variabilidad*).

4 ANTECEDENTES

Tradicionalmente, el problema de planificación de intervenciones quirúrgicas se ha abordado en dos etapas: una primera etapa donde se determina la fecha y quirófano donde se va a realizar la intervención quirúrgica, y una segunda etapa donde se obtiene la secuenciación de las cirugías a realizar en cada quirófano cada día. Sin embargo, solo en alguna de las contribuciones existentes se ha abordado el problema de planificación teniendo en cuenta de la disponibilidad de los recursos postoperatorios necesarios en cada intervención.

Hasta ahora, la no integración de recursos postoperatorios en la planificación de intervenciones quirúrgicas había desembocado en consecuencias negativas, como el aplazo de cirugías o rechazo de pacientes, aumento de los tiempos de espera de los cirujanos, etc., llevando al sobrecoste del proceso quirúrgico en el Hospital y a la insatisfacción de pacientes y personal sanitario. Por ello, surge la necesidad de integración de las áreas postoperatorias en la planificación de las intervenciones quirúrgicas, pues son otra parte de los quirófanos.

Esta situación ha tenido un impacto aún mayor en el contexto sanitario actual, donde la pandemia provocada por la COVID-19 ha generado un escenario de variabilidad e incertidumbre, donde los profesionales han debido hacer frente a una pandemia, con medios insuficientes, como por ejemplo las camas hospitalarias o camas ICU, quedando en segundo plano la prestación de otras especialidades médicas, entre ellas, las intervenciones quirúrgicas. La necesidad de elaborar un modelo de planificación que integre las camas postoperatorias es aún mayor.

4.1 Revisión de la literatura

La planificación/programación de intervenciones quirúrgicas en el nivel operativo *off-line* es un tema de investigación popular (ver las revisiones literarias propuestas por B. Cardoen et al. (2010), F. Guerreiro et al. (2011), J. H. May et al. (2011) y S. Gür et al. (2018)), centrándose la mayor parte de la literatura en la planificación/programación de los quirófanos.

José M. Molina-Pariente et al. (2015) propone un modelo de programación lineal entera para asignar la fecha de intervención y quirófano a un conjunto de cirugías en la lista de espera, minimizando el tiempo de acceso a los pacientes con distintas prioridades médicas. Además, define el número máximo de quirófanos donde cada cirujano puede realizar intervenciones teniendo en cuenta su especialidad y carga de trabajo, reduciendo el tiempo de inactividad y evitando la superposición de cirugías consecutivas. A. Jebali et al. (2006) utiliza la programación entera para asignar cirugías y pacientes a quirófanos y luego emplea la programación entera mixta para la secuenciación de los pacientes en los quirófanos. F. Dexter et al. (1999) y E. Hans et al. (2008) utilizan heurísticas constructivas para programar pacientes individuales con el objetivo de mejorar la utilización de quirófanos. Un tema común en estos y otros artículos, cuyo objetivo es programar pacientes individuales, es la decisión de ignorar los efectos posteriores de los programas resultantes.

Parte de la literatura que se centra en la planificación de quirófanos considera además de los quirófanos, las camas de recuperación. Sin embargo, la mayoría de ellas abordan de forma individual una de las Unidades de Gestión Clínica de recuperación, sin abordar las tres posibles Unidades de Gestión Clínica existentes en el proceso de recuperación y sus correspondientes camas postoperatorias (camas ICU, camas PACU y camas hospitalarias).

Algunas de las contribuciones, integran la planificación/programación de las intervenciones quirúrgicas con las camas de la Unidad de Recuperación Post-Anestésica, es decir, con las camas PACU. Así, H. Saadouli et al. (2015) emplea la programación matemática para decidir que intervenciones se realizarán y en qué quirófanos sin considerar los recursos de la Unidad Post-Anestésica y en una etapa posterior emplea otro modelo de simulación de eventos discretos para medir el impacto de la incertidumbre de los recursos PACU. Por otro lado, F. Dexter et al. (2006) estudia el impacto de distintas heurísticas de secuenciación de cirugías en la carga de trabajo de la Unidad de Recuperación Post-Anestésica. Sin embargo, algunos autores consideraron los recursos PACU en la fase de generación de la programación. Y. Wang et al. (2015) considera un algoritmo

de optimización por enjambre de partículas para el problema de programación de quirófanos con los recursos post-anestésicos. H. Fei et al. (2010) desarrolla una heurística de dos etapas, donde en la primera de ellas, se asigna fecha a las cirugías y en la segunda, se asignan las cirugías a los quirófanos y se secuencian mediante un algoritmo genético híbrido, modelo que respeta la asignación paciente-cirujano, considera los tiempos de recuperación post-anestésica y permite el abordaje de los quirófanos asumiendo duraciones deterministas de las cirugías y recuperaciones. M. Ban et al. (2017) realiza la programación de cirugías diaria de pacientes electivos, considerando cirujanos, quirófanos, y la Unidad Post-Anestesia.

Otras contribuciones integran en su modelo de planificación/programación de intervenciones quirúrgicas las camas de la Unidad de Cuidados Intensivos. Por ello, A. Jebali et al. (2006) propone un método de dos fases para la programación diaria de los quirófanos. En la primera fase, selecciona las cirugías a realizar de la lista de espera y las asigna a quirófanos considerando la disponibilidad de camas ICU. En la segunda fase, realiza la secuenciación de las cirugías asignadas con el objetivo de mejorar la utilización de los quirófanos.

Por otro lado, A. Fügener et al. (2014) desarrolla un modelo de programación de cirugías incorporando de manera conjunta la demanda de camas ICU y camas de hospitalización requerida por el paciente tras la intervención quirúrgica y A. Abedini et al. (2017) realiza un modelo de programación de quirófanos para reducir el bloqueo en el proceso perioperatorio considerando la disponibilidad de los recursos postoperatorios en la Unidad de Cuidados Intensivos y en la Unidad de Recuperación Post-Anestésica mediante un modelo de programación entera.

En la literatura, también podemos encontrar contribuciones que integran los recursos postoperatorios pero no hacen referencia a qué Unidad de Gestión Clínica pertenecen. Esto ocurre en V. Augusto et al. (2010), donde se considera la programación de cirugías considerando el número de camas de recuperación en el periodo t, pero no hace referencia a qué camas se considera camas de recuperación.

La mayoría de las contribuciones que abordan la programación e integran las camas postoperatorias, incorporan la disponibilidad de las camas como una constante, en lugar de cómo parte del objetivo, como por ejemplo la contribución de J. Blake et al. (2002), donde se emplea un modelo de programación lineal que determina cuánto tiempo asignar a cada especialidad, considerando la duración de estancia de los pacientes y la capacidad de las camas hospitalarias constantes, pero la ocupación del hospital no forma parte de su objetivo.

El número de artículos que incorporan la nivelación de la ocupación de camas en el objetivo de su modelo es limitado. J. Beliën et al. (2007) comienza con un modelo de programación entera no lineal que minimiza la escasez total de camas esperada en el transcurso del ciclo. Sin embargo, no distingue entre los tipos de camas de recuperación (PACU, ICU y hospitalarias). También, P. Santibáñez et al. (2007) busca minimizar la ocupación máxima de camas, y su modelo de programación entera incluyen restricciones que representan conflictos de equipos y objetivos de rendimiento. Por otro lado, C. Prince et al. (2011) tiene el objetivo de reducir el embarque en la Unidad de Recuperación Post-Anestésica, debido a la saturación de la camas ICU. Para ello, distingue entre los tres tipos de camas postoperatorias existentes (camas PACU, ICU y hospitalarias) con el objetivo de minimizar el exceso de capacidad en la Unidad de Cuidados Críticos, dado que sus camas son un recurso cuello de botella para el problema.

La contribución de este Trabajo de Fin de Grado es diseñar y desarrollar un modelo de programación lineal entera que aborde la resolución exacta del problema integrado de planificación de intervenciones quirúrgicas y camas postoperatorias, contemplando los tres posibles tipos de camas postoperatorias (camas PACU, camas ICU y camas hospitalarias). Además, el modelo propuesto integrará aspectos de gestión no considerados anteriormente por la literatura para el problema propuesto, como es el caso: asignación de cirujanos a quirófano y día, minimización de la tardanza total del paciente en el sistema (diferencia entre el tiempo de espera y el plazo máximo de respuesta impuesta por los sistemas sanitarios) e integración de las camas postoperatorias como parte de la función objetivo.

5 DESCRIPCIÓN DEL PROBLEMA

La Unidad de Cirugía Plástica y Grandes Quemados realiza cirugías de tres tipos: urgencias, emergencias y electivas. Este proyecto aborda la gestión de intervenciones quirúrgicas, exceptuando las de emergencia puesto que debido a su gravedad cuentan con recursos adicionales, y la gestión de las camas postoperatorias en el nivel de decisión operativo *off-line*.

Este Trabajo de Fin de Grado se desarrolla en un entorno de taller de flujo híbrido constituido por dos etapas: Quirófanos y camas postoperatorias. Todos los pacientes seguirán la misma ruta de fabricación (*Flow Shop*), primero la intervención quirúrgica y posteriormente la cama postoperatoria. Ambas etapas están constituidas por recursos idénticos en paralelo (*Parallel Machine*), quirófanos en la primera, y camas postoperatorias (PACU, ICU y hospitalarias) en la segunda. Cada paciente será atendido por uno de los recursos y el tiempo de acceso no dependerá de los recursos sino de la prioridad médica del paciente.

En este capítulo se presenta el problema, se describen las restricciones, las variables de decisión y el objetivo establecido que caracterizan al problema de estudio de forma detallada.

5.1 Limitaciones de los recursos operatorios y postoperatorios

La Unidad de Gestión Clínica objeto de análisis cuenta con 4 quirófanos multifuncionales para la realización de intervenciones quirúrgicas. Sin embargo, uno de ellos está reservado de forma permanente para la realización de intervenciones de urgencia, donde se usan recursos adicionales (recursos quirúrgicos urgentes).

Cada día, tres quirófanos están disponibles para realizar cirugías de urgencia y electivas de 8:30 a 14:30. Sin embargo, por las tardes, la disponibilidad de los quirófanos varía en función de la ocupación. De lunes a jueves, uno de los quirófanos está siempre disponible de 15:30 a 19:30. Otro quirófano, está disponible lunes martes y jueves de 15:30 a 19:30. El tercer quirófano en cuestión, está disponible de 15:30 a 19:30 de lunes a jueves, sin embargo, si la ocupación es baja, solo estará disponible lunes, martes y jueves.

En cuanto a los cirujanos, cada cirujano tiene un tiempo máximo disponible para realizar cirugías, siendo 0 (si ese día no está disponible) u 6,5 horas (si ese día trabaja). El número de quirófanos a los que se puede asignar un cirujano al día está limitado (u_s) para reducir el tiempo de inactividad y evitar el solapamiento de cirugías consecutivas (José M. Molina-Pariente, 2015).

Las camas postoperatorias, son recursos de capacidad limitada que deben gestionarse de forma integrada con la planificación de las intervenciones quirúrgicas, puesto que algunas de ellas, en concreto las camas ICU y las camas hospitalarias, representan un cuello de botella para este problema. Se corresponden con la capacidad de almacenaje de la producción en el entorno industrial.

Finalmente, el resto de recursos perioperatorios humanos e instrumentales, se consideran disponibles en cualquier momento, es por ello que no se contemplan restricciones correspondientes al transporte de pacientes en este problema.

5.2 Descripción del entorno

El proceso consta de dos etapas consecutivas; la primera etapa hace referencia a los quirófanos y la segunda a las camas postoperatorias. Dentro del conjunto de pacientes, se pueden distinguir cuatro tipos: En primer lugar, a aquellos que tras la intervención quirúrgica no necesitan cama postoperatoria. En segundo lugar, a aquellos que tras la intervención quirúrgica necesitan cama PACU. En tercer lugar, a aquellos que además tras la ocupación de la cama PACU, necesitan cama hospitalaria. Y, por último, aquellos que tras la intervención, necesitan cama ICU. Todos los pacientes que ingresen en la Unidad de Cuidados Intensivos, ocuparán posteriormente una cama hospitalaria. La duración de la intervención es estimada por el personal encargado, en base al historial médico y las características del paciente (José M. Molina-Pariente, 2015), al igual que el

tipo de cama necesaria tras la intervención. No obstante, situaciones como una complicación médica en la intervención o una recuperación postoperatoria distinta a la prevista podrían afectar a esta suposición, que por los demás, sería sólida.

Cada cirugía debe ser programada dentro de un periodo definido por su fecha más temprana y su fecha más tardía (José M. Molina-Pariente, 2015). La fecha más temprana (*release date*), es la fecha a partir de la cual el paciente puede ser intervenido una vez que las pruebas médicas previas necesarias se han completado. La fecha de entrega (*due date*), dependerá del tiempo máximo antes de tratamiento establecida según la urgencia médica del paciente. Los máximos tiempos considerados en la especialidad son 45, 180 y 365 días (José M. Molina-Pariente, 2015). La realización de la intervención *i* solo podrá llevarse a cabo por el cirujano asignado *s,* en función de su especialidad y carga de trabajo. Los mismo ocurre con las camas postoperatorias, un paciente no podrá ser asignado aleatoriamente a un tipo de cama u otra, si no que dependerá de la atención médica que precise tras su intervención (β_i).

Las camas PACU e ICU serán ocupadas por aquellos pacientes que las requieran el mismo día/semana de la intervención. En cuanto a las camas hospitalarias, serán ocupadas por los pacientes el día que reciban el alta en la Unidad de Cuidados Intensivos o Unidad de Recuperación Postanestésica.

5.3 Función objetivo

La función objetivo deriva de los indicadores de actuación empleados por el Sistema Regional de Salud de Andalucía (José M. Molina-Pariente, 2015). Dada una fecha límite de cirugía para cada paciente conocida como fecha de entrega o *due date*, el objetivo es maximizar el número de intervenciones quirúrgicas planificadas minimizando el retraso o *tardiness* en el tiempo de acceso de dichos pacientes, teniendo en cuenta su peso clínico, y maximizando a su vez la utilización de las camas postoperatorias disponibles.

6 METODOLOGÍA

Una vez comprendida la naturaleza y complejidad del problema y descrito el método elegido para su resolución, en esta sección se exponen los modelos de programación lineal desarrollados para la resolución del problema de planificación. Asumimos dos modelos deterministas donde todas las características del problema de decisión son conocidas y evaluadas individualmente.

6.1 Modelo de programación lineal entera 1: Por etapas

Este modelo aborda el modelo de planificación en dos etapas: una primera etapa donde se resuelve la planificación de intervenciones quirúrgicas, asignando fecha y quirófano para cada una de ellas en el horizonte temporal, y una segunda etapa, donde tras haberse realizado la planificación de las intervenciones quirúrgicas se asignan las camas postoperatorias en función de las necesidades de los pacientes intervenidos.

6.1.1 Datos

Tabla 6-1: Conjuntos, parámetros y variables usados en la **etapa 1** del modelo de decisión ILP (1)

Índices y conjuntos	
$i \in I$	Conjunto de pacientes
$h \in H$	Conjunto de días dentro del horizonte de planificación
$j \in J$	Conjunto de quirófanos
$s \in S$	Conjunto de cirujanos
Parámetros	
r_{jh}	Capacidad regular del quirófano j el día h (minutos)
a_{sh}	Capacidad regular del cirujano s el día h (minutos)
u_s	Número entero no negativo de quirófanos en los que el cirujano s puede realizar cirugías en el mismo día
y_i	Cirujano a cargo del paciente i
rd_i	Fecha más temprana para realizar la cirugía del paciente i
d_i	Fecha más tarde para realizar la cirugía del paciente i
δ_{ijh}	Parámetro binario que toma el valor 1 si la cirugía del paciente i puede ser realizada en el quirófano j el día h; 0 c.c
t_i	Duración esperada de la cirugía i (minutos)
w_i	Peso clínico de la cirugía i
a	Parámetro de ponderación del segundo término de la F.O. Etapa 1 (*Tardiness*)
Variables	
X_{ijh}	1 si el paciente i es operado en el quirófano j el día h; 0 c.c
Z_{sjh}	1 si el cirujano s es asignado al quirófano j el día h; 0 c.c

Tabla 6-2: Conjuntos, parámetros y variables usados en la **etapa 2** del modelo de decisión ILP (1)

Índices y conjuntos	
$i \in I$	Conjunto de pacientes
$h \in H$	Conjunto de días dentro del horizonte de planificación
$w \in W$	Conjunto de semanas dentro del horizonte de planificación
$l \in L$	Conjunto de camas PACU
$m \in M$	Conjunto de camas ICU
$n \in N$	Conjunto de camas HOSPITALARIAS
Parámetros	
$t_{i,pacu}$	Tiempo estimado del paciente i en cama PACU (horas)
$t_{i,icu}$	Tiempo estimado del paciente i en cama ICU (horas)

$t_{i,hospitalaria}$	Tiempo estimado del paciente *i* en cama HOSPITALARIA (horas)
a_{lh}	Capacidad de la cama PACU *l* el día *h* (horas)
b_{mw}	Capacidad de la cama ICU *m* en la semana *w* (horas)
c_{nw}	Capacidad de la cama HOSPITALARIA *n* en la semana *w* (horas)
X_{ijh}	1 si el paciente *i* es operado en el quirófano *j* el día *h*, 0 c.c
β_i	Parámetro que toma el valor: 0 si el paciente *i* no ocupa cama 1 si el paciente *i* ocupa cama PACU 2 si el paciente *i* ocupa cama PACU y cama HOSPITALARIA 3 si el paciente *i* ocupa cama ICU
b	Parámetro de ponderación del término de la F.O. Etapa 2
Variables	
A_{ilh}	1 si el paciente *i* ocupa la cama PACU *l* el día *h*, 0 c.c
B_{imw}	1 si el paciente *i* ocupa la cama ICU *m* la semana *w*, 0 c.c
C_{inw}	1 si el paciente *i* ocupa la *cama* HOSPITALARIA *n* la semana *w*, 0 c.c
T_i	Tardanza o *tardiness* en la fecha de operación del paciente *i*

6.1.2 Descripción del modelo

Consiste en un modelo de programación lineal entera que resuelve en una primera etapa la planificación de intervenciones quirúrgicas y en una segunda etapa, dependiente de la planificación de intervenciones quirúrgicas realizada en la primera etapa, la planificación de las camas postoperatorias.

6.1.2.1 Etapa 1 (Cirugías)

$$Max\ \frac{1}{W_{max}} \cdot \sum_{h \in H} \frac{1}{h} \left(\sum_{i \in I} \sum_{j \in J} w_i \cdot X_{ijh} \right) - \frac{a}{(H+1) \cdot W_{max}} \cdot \sum_{i \in I} w_i \cdot T_i$$

Sujeto a:

Restricciones Objetivo

$$T_i \geq \sum_{j \in J} \sum_{h \in H} X_{ijh} \cdot h - d_i \qquad (\forall i \in I)(\boldsymbol{R.1})$$

$$T_i \geq (H + 1 - d_i) \cdot \left(1 - \sum_{j \in J} \sum_{h \in H} X_{ijh} \right) \qquad (\forall i \in I)\ (\boldsymbol{R.2})$$

Restricciones de la etapa 1:

$$\sum_{j \in J} \sum_{h \in H} X_{ijh} \leq 1 \qquad (\forall i \in I)\ (\mathbf{2})$$

$$\sum_{j \in J} \sum_{h=0}^{rd_i - 1} X_{ij} = 0 \qquad (\forall i \in I)(\mathbf{3.1})$$

$$\sum_{i \in I} t_i \cdot X_{ijh} \leq r_{jh} \qquad (\forall j \in J, \forall h \in H)(\mathbf{4})$$

$$\sum_{j \in J} \sum_{\substack{i \in I \\ y_i = s}} t_i \cdot X_{ijh} \leq a_{sh} \qquad (\forall s \in S, \forall h \in H)(\mathbf{5})$$

$$\sum_{j\in J} Z_{sjh} \leq u_s \qquad (\forall s \in S, \forall h \in S)(\mathbf{6})$$

$$\sum_{\substack{i\in I \\ y_i=s}} t_i \cdot X_{ijh} \leq r_{jh} \cdot Z_{sjh} \qquad (\forall s \in S, \forall j \in J, \forall h \in H)\ (\mathbf{7.1})$$

$$\sum_{\substack{i\in I \\ y_i=s}} X_{ijh} \geq Z_{sjh} \qquad (\forall s \in S, \forall j \in J, \forall h \in H)\ (\mathbf{7.2})$$

$$X_{ijh} = 0 \qquad (\forall i \in I, \forall j \in J, \forall h \in H \mid \delta_{ij} = 0)\ (\mathbf{8})$$

$$X_{ijh} \in \{0,1\} \qquad (\forall i \in I, \forall j \in J, \forall h \in H)(\mathbf{9})$$

$$Z_{sjh} \in \{0,1\} \qquad (\forall s \in S, \forall j \in J, \forall h \in H)\ (\mathbf{10})$$

$$T_i \geq 0 \qquad (\forall i \in I)(\mathbf{11})$$

$$Donde\ W_{max} = \sum_{i\in I} w_{i,max} = I \cdot w_{max} \qquad (\forall i \in I | w_{i,max} = w_{max})$$

6.1.2.2 Etapa 2 (Camas postoperatorias)

$$Max\ \frac{b}{2} \cdot \sum_{i\in I}\left[\sum_{h\in H}\sum_{l\in L} A_{ilh} + \sum_{w\in W}\left(\sum_{m\in M} B_{imw} + \sum_{n\in N} C_{inw}\right)\right]$$

Sujeto a:

Restricciones camas PACU (Día)

$$\sum_{h\in H}\sum_{l\in L} A_{ilh} \leq 1 \qquad (\forall i \in I)\ (\mathbf{1})$$

$$\sum_{i\in I} t_{pacu} \cdot A_{ilh} \leq a_{lh} \qquad (\forall l \in L, \forall h \in H)(\mathbf{2})$$

$$\sum_{l\in L} A_{ilh} = \sum_{j\in J} X_{ijh} \qquad (\forall i \in I, \forall h \in H \mid \beta_i = 1 \text{ ó } \beta_i = 2)\ (\mathbf{3})$$

Restricciones camas ICU (Semana)

$$\sum_{w\in W}\sum_{m\in M} B_{imw} \leq 1 \qquad (\forall i \in I)(\mathbf{4})$$

$$\sum_{i\in I} t_{icu} \cdot B_{imw} \leq b_{mw} \qquad (\forall m \in M, \forall w \in W)(\mathbf{5})$$

$$\sum_{m\in M} B_{imw} = \sum_{h\in w}\sum_{j\in J} X_{ijh} \qquad (\forall i \in I, \forall w \in W \mid \beta_i = 3)\ (\mathbf{6})$$

Restricciones camas hospitalarias (Semana)

$$\sum_{n\in N} C_{inw} \leq 1 \qquad (\forall i \in I, \forall w \in W)\ (\mathbf{7})$$

$$\sum_{i \in I} t_{hospital} \cdot C_{inw} \leq c_{nw} \qquad (\forall n \in N, \forall w \in W)(\mathbf{8})$$

$$\sum_{n \in N} C_{inw} = \sum_{h \in w} \sum_{l \in L} A_{ilh} \qquad (\forall i \in I, \forall w \in W | \beta_i = 2)\ (\mathbf{9.1})$$

$$\sum_{n \in N} C_{inw} = \sum_{m \in M} B_{imw} \qquad (\forall i \in I, \forall w \in W | \beta_i = 3)\ (\mathbf{9.2})$$

$$A_{ilh} = B_{imw} = C_{inw} = 0 \qquad (i \in I, \forall l \in L, \forall m \in M, \forall n \in N, \forall h \in H, \forall w \in W \mid \beta_i = 0)\ (\mathbf{10})$$

$$A_{ilh} = 0 \qquad (\forall i \in I, \forall l \in L, \forall h \in H | \beta_i \neq 1, \beta_i \neq 2)\ (\mathbf{11})$$

$$B_{imw} = 0 \qquad (\forall i \in I, \forall m \in M, \forall w \in W \mid \beta_i \neq 3)\ (\mathbf{12})$$

$$C_{inw} = 0 \qquad (\forall i \in I, \forall n \in N, \forall w \in W \mid \beta_i \neq 2, \beta_i \neq 3\ )(\mathbf{13})$$

$$A_{il} \in \{0,1\} \qquad (\forall i \in I, \forall l \in L, \forall h \in H)(\mathbf{14})$$

$$B_{imw} \in \{0,1\} \qquad (\forall i \in I, \forall m \in M, \forall w \in W)\ (\mathbf{15})$$

$$C_{inw} \in \{0,1\} \qquad (\forall i \in I, \forall n \in N, \forall w \in W)(\mathbf{16})$$

6.1.3 Explicación del modelo

6.1.3.1 Etapa 1 (Cirugías)

La función objetivo está compuesta por la suma de dos términos: el primero de ellos maximiza el nivel de servicio (definido por el coeficiente entre el peso clínico (w_i) y la fecha de la cirugía programada) de una especialidad quirúrgica priorizando la intervención de pacientes con mayor peso clínico y el segundo minimiza el *tardiness* total del conjunto de pacientes considerando el peso clínico de cada paciente.

Las restricciones (R1) y (R2) calculan el valor del *tardiness* para cada paciente. Si el paciente ha sido intervenido en el horizonte temporal, (R1) calcula el valor del *tardiness* comparando la fecha de la intervención con la fecha de entrega o *due date*, de manera que si la intervención se realiza con posterioridad a la fecha de entrega, la diferencia entre ambas será el valor del *tardiness*. Por el contrario, si el paciente ha sido intervenido y la fecha de intervención es anterior a la fecha de entrega, tomará valor 0 (restricción 11). Si el paciente no ha sido intervenido en el horizonte temporal, la restricción (R2) obliga a que su *tardiness* sea igual a la diferencia entre el día posterior al último día del horizonte de planificación (H+1) y la fecha de entrega, siendo la máxima penalización posible teniendo en cuenta que si la cirugía no se ha programado dentro del horizonte temporal, no se hará al menos hasta el día posterior al último día del horizonte de planificación. La restricción (2) asegura que la cirugía del paciente *i* sea programada como máximo una vez en el horizonte de planificación *H*. Las restricciones (3.1) define la fecha más temprana para la realización de la cirugía del paciente *i*. De esta manera se garantiza que todas las pruebas médicas necesarias serán realizadas de manera previa a la intervención. La restricción (4) prohíbe que la duración total las intervenciones asignadas a los distintos cirujanos en un determinado quirófano *j* y día *h*, supere la capacidad regular del quirófano *j* en el día *h*. La restricción (5) prohíbe que la duración total de las cirugías asignadas al cirujano s sea mayor que su capacidad en el día *h* (a_{sh}). La restricción (6) limita el número de quirófanos diarios que pueden ser asignados al cirujano *s* en el día *h*. Las restricciones (7.1) y (7.2) definen a que quirófano *j* y día *h* se asigna el cirujano *s*. No se asignará un cirujano a un quirófano si no hay al menos una cirugía programada en dicho quirófano y día. El tiempo total de las cirugías realizadas por el cirujano *s* en el quirófano j y día *h* no debe superar la capacidad del quirófano en dicho día (r_{jh}). La restricción (8) garantiza que la intervención del paciente *i* sea realizada en un quirófano *j* y día *h* adecuado según el tipo de operación (δ_{ijh} =1). Por último, las restricciones (9) y (10) son restricciones binarias para las variables de decisión asociadas a la etapa 1.

6.1.3.2 Etapa 2 (Camas postoperatorias)

La función objetivo consiste en maximizar la ocupación de los tres tipos de camas existentes, es por ello por lo

que está formada por el sumatorio de los tres términos correspondientes: A_{ilh}, B_{imw} y C_{inw}.

La restricción (1) asegura que la ocupación de la cama PACU *l* por cada paciente *i* sea programada como máximo un día en el horizonte de planificación *H*. La restricción (4) garantiza que la ocupación de la cama ICU *m* por cada paciente *i* sea programada como máximo una semana en el horizonte de planificación. La restricción (7) asegura que la ocupación de la cama HOSPITALARIA *n* por cada paciente *i* sea programada como máximo una vez en el horizonte de planificación. Las restricciones (2), (5) y (8) prohíbe que la cantidad total de pacientes asignados a la cama aplicable en cada caso en el día *h*/semana *w* sea mayor que su capacidad regular correspondiente. La restricción (3) impide que la cama PACU *l* sea ocupada por el paciente *i* el día *h*, si el paciente no ha sido operado en cualquiera de los quirófanos en el mismo día. En caso contrario, además el paciente debe necesitar dicha cama ($\beta i=1$ ó $\beta i=2$). La restricción (6) garantiza que la cama ICU *m* no será ocupada por el paciente *i* en la semana *w*, si dicho paciente no ha sido operado en dicha semana y no necesita ingresar en la dicha unidad ($\beta_i \neq 3$). Las restricciones (9.1) y (9.2) garantiza que el paciente *i* ocupe la cama HOSPITALARIA *n* si ha ocupado cualquiera de los otros dos tipos de cama previamente y requiere ingreso hospitalario ($\beta_i=2$ ó $\beta_i=3$). La restricción (10) prohíbe que el paciente *i* ocupe cualquier cama si no necesita ninguna ($\beta_i=0$). Las restricciones (11), (12), (13) prohíben que el paciente *i* ocupe cualquier tipo de cama si no necesita dicho tipo. Finalmente, las restricciones (14), (15) y (16) son restricciones binarias para las variables de decisión asociadas a la etapa 2.

6.2 Modelo de programación lineal entera 2: Integrado

A diferencia del modelo anterior, el modelo propuesto en esta sección aborda la planificación de las intervenciones quirúrgicas y camas postoperatorias de manera integrada en un único problema de optimización que asigna fecha, quirófano y cama postoperatoria para cada intervención en el horizonte de planificación.

6.2.1 Datos

Tabla 6-3: Conjuntos, parámetros y variables usados en el modelo de decisión ILP (2)

Índices y conjuntos	
$i \in I$	Conjunto de pacientes
$h \in H$	Conjunto de días dentro del horizonte de planificación
$w \in W$	Conjunto de semanas dentro del horizonte de planificación
$j \in J$	Conjunto de quirófanos
$s \in S$	Conjunto de cirujanos
$l \in L$	Conjunto de camas PACU
$m \in M$	Conjunto de camas ICU
$n \in N$	Conjunto de camas HOSPITALARIAS
Parámetros	
r_{jh}	Capacidad regular del quirófano *j* el día *h* (minutos)
a_{sh}	Capacidad regular del cirujano *s* el día *h* (minutos)
u_s	Número entero no negativo de quirófanos en los que el cirujano *s* puede realizar cirugías en el mismo día
y_i	Cirujano a cargo del paciente *i*
rd_i	Fecha más temprana para realizar la cirugía del paciente *i*
d_i	Fecha más tarde para realizar la cirugía del paciente *i*
δ_{ijh}	Parámetro binario que toma el valor 1 si la cirugía del paciente *i* puede ser realizada en el quirófano *j* el día *h*; 0 c.c
t_i	Duración esperada de la cirugía *i* (minutos)
w_i	Peso clínico de la cirugía *i*
$t_{i,pacu}$	Tiempo estimado del paciente *i* en cama PACU (horas)
$t_{i,icu}$	Tiempo estimado del paciente *i* en cama ICU (horas)
$t_{i,hospitalaria}$	Tiempo estimado del paciente *i* en cama HOSPITALARIA (horas)

a_{lh}	Capacidad de la cama PACU *l* el día *h* (horas)
b_{mw}	Capacidad de la cama ICU *m* en la semana *w* (horas)
c_{nw}	Capacidad de la cama HOSPITALARIA *n* en la semana *w* (horas)
X_{ijh}	1 si el paciente *i* es operado en el quirófano *j* el día *h*, 0 c.c
β_i	Parámetro que toma el valor: 0 si el paciente *i* no ocupa cama 1 si el paciente *i* ocupa cama PACU 2 si el paciente *i* ocupa cama PACU y cama HOSPITALARIA 3 si el paciente *i* ocupa cama ICU
a	Parámetro de ponderación del segundo término de la F.O. Etapa 1 (*Tardiness*)
b	Parámetro de ponderación del término de la F.O. Etapa 2
Variables	
X_{ijh}	1 si el paciente *i* es operado en el quirófano *j* el día *h*; 0 c.c
Z_{sjh}	1 si el cirujano *s* es asignado al quirófano *j* el día *h*; 0 c.c
A_{ilh}	1 si el paciente *i* ocupa la cama PACU *l* el día *h*, 0 c.c
B_{imw}	1 si el paciente *i* ocupa la cama ICU *m* la semana *w*, 0 c.c
C_{inw}	1 si el paciente *i* ocupa la cama HOSPITALARIA *n* la semana *w*, 0 c.c
T_i	Tardanza o *tardiness* en la fecha de operación del paciente *i*

6.2.2 Descripción del modelo

$$Max\ \frac{1}{W_{max}} \cdot \sum_{h\in H} \frac{1}{h}\left(\sum_{i\in I}\sum_{j\in J} w_i \cdot X_{ijh}\right) - \frac{a}{(H+1)\cdot W_{max}} \cdot \sum_{i\in I} w_i \cdot T_i + \frac{b}{2} \cdot \sum_{i\in I}\left[\sum_{h\in H}\sum_{l\in L} A_{ilh} + \sum_{w\in W}\left(\sum_{m\in M} B_{imw} + \sum_{n\in N} C_{inw}\right)\right]$$

Sujeto a:

Restricciones Objetivo

$$T_i \geq \sum_{j\in J}\sum_{h\in H} X_{ij} \cdot h - d_i \qquad (\forall i \in I)(\boldsymbol{R.1})$$

$$T_i \geq (H+1-d_i)\cdot\left(1 - \sum_{j\in J}\sum_{h\in H} X_{ijh}\right) \qquad (\forall i \in I)\ (\boldsymbol{R.2})$$

Etapa 1:

$$\sum_{j\in J}\sum_{h\in H} X_{ijh} \leq 1 \qquad (\forall i \in I)\ (\mathbf{2})$$

$$\sum_{j\in J}\sum_{h=0}^{rd_i-1} X_{ijh} = 0 \qquad (\forall i \in I)(\mathbf{3.1})$$

$$\sum_{i\in I} t_i \cdot X_{ijh} \leq r_{jh} \qquad (\forall j \in J, \forall h \in H)(\mathbf{4})$$

$$\sum_{j\in J}\sum_{\substack{i\in I \\ y_i = s}} t_i \cdot X_{ijh} \leq a_{sh} \qquad (\forall s \in S, \forall h \in H)(\mathbf{5})$$

$$\sum_{j\in J} Z_{sjh} \leq u_s \qquad (\forall s \in S, \forall h \in S)(\mathbf{6})$$

$$\sum_{\substack{i\in I \\ y_i=s}} t_i \cdot X_{ijh} \leq r_{jh} \cdot Z_{sjh} \qquad (\forall s \in S, \forall j \in J, \forall h \in H)\ (\mathbf{7.1})$$

$$\sum_{\substack{i\in I \\ y_i=s}} X_{ijh} \geq Z_{sj} \qquad (\forall s \in S, \forall j \in J, \forall h \in H)\ (\mathbf{7.2})$$

$$X_{ijh} = 0 \qquad (\forall i \in I, \forall j \in J, \forall h \in H \mid \delta_{ijh} = 0)\ (\mathbf{8})$$

$$X_{ijh} \in \{0,1\} \qquad (\forall i \in I, \forall j \in J, \forall h \in H)(\mathbf{9})$$

$$Z_{sjh} \in \{0,1\} \qquad (\forall s \in S, \forall j \in J, \forall h \in H)\ (\mathbf{10})$$

$$T_i \geq 0 \qquad (\forall i \in I)(\mathbf{11})$$

$$Donde\ W_{max} = \sum_{i\in I} w_{i,max} = I \cdot w_{max} \qquad (\forall i \in I | w_{i,max} = w_{max})$$

Etapa 2:

Restricciones camas PACU (Día)

$$\sum_{h\in H} \sum_{l\in L} A_{ilh} \leq 1 \qquad (\forall i \in I)\ (\mathbf{1})$$

$$\sum_{i\in I} t_{pacu} \cdot A_{ilh} \leq a_{lh} \qquad (\forall l \in L, \forall h \in H)(\mathbf{2})$$

$$\sum_{l\in L} A_{ilh} = \sum_{j\in J} X_{ij} \qquad (\forall i \in I, \forall h \in H \mid \beta_i = 1 \text{ ó } \beta_i = 2)\ (\mathbf{3})$$

Restricciones camas ICU (Semana)

$$\sum_{w\in W} \sum_{m\in M} B_{imw} \leq 1 \qquad (\forall i \in I)(\mathbf{4})$$

$$\sum_{i\in I} t_{icu} \cdot B_{imw} \leq b_{mw} \qquad (\forall m \in M, \forall w \in W)(\mathbf{5})$$

$$\sum_{m\in M} B_{imw} = \sum_{h\in w} \sum_{j\in J} X_{ijh} \qquad (\forall i \in I, \forall w \in W \mid \beta_i = 3)\ (\mathbf{6})$$

Restricciones camas hospitalarias (Semana)

$$\sum_{n\in N} C_{inw} \leq 1 \qquad (\forall i \in I, \forall w \in W)\ (\mathbf{7})$$

$$\sum_{i\in I} t_{hospital} \cdot C_{inw} \leq c_{nw} \qquad (\forall n \in N, \forall w \in W)(\mathbf{8})$$

$$\sum_{n\in N} C_{inw} = \sum_{h\in w} \sum_{l\in L} A_{ilh} \qquad (\forall i \in I, \forall w \in W \mid \beta_i = 2)\ (\mathbf{9.1})$$

$$\sum_{n\in N} C_{inw} = \sum_{m\in M} B_{imw} \qquad (\forall i \in I, \forall w \in W \mid \beta_i = 3)\ (\mathbf{9.2})$$

$$A_{il} = B_{imw} = C_{inw} = 0 \qquad (i \in I, \forall l \in L, \forall m \in M, \forall n \in N, \forall h \in H, \forall w \in W \mid \beta_i = 0)\ (\mathbf{10})$$

$A_{ilh} = 0$ $(\forall i \in I, \forall l \in L, \forall h \in H | \beta_i \neq 1, \beta_i \neq 2)$ **(11)**

$B_{imw} = 0$ $(\forall i \in I, \forall m \in M, \forall w \in W \mid \beta_i \neq 3)$ **(12)**

$C_{inw} = 0$ $(\forall i \in I, \forall n \in N, \forall w \in W \mid \beta_i \neq 2, \beta_i \neq 3)$ **(13)**

$A_{il} \in \{0,1\}$ $(\forall i \in I, \forall l \in L, \forall h \in H)$ **(14)**

$B_{imw} \in \{0,1\}$ $(\forall i \in I, \forall m \in M, \forall w \in W)$ **(15)**

$C_{inw} \in \{0,1\}$ $(\forall i \in I, \forall n \in N, \forall w \in W)$ **(16)**

6.2.3 Explicación del modelo

La función objetivo está compuesta por la suma de tres términos: el primero de ellos maximiza el nivel de servicio de una especialidad quirúrgica priorizando la intervención de pacientes con mayor peso clínico, el segundo minimiza el *tardiness* total del conjunto de pacientes considerando el peso clínico de cada paciente y el tercero maximiza la ocupación de los tres tipos de camas postoperatorias.

Al igual que en el modelo anterior, las restricciones (R1) y (R2) calculan el valor del *tardiness* para cada paciente. Si el paciente ha sido intervenido en el horizonte temporal, el valor del *tardiness* será la diferencia entre la fecha de intervención y la fecha de entrega si la intervención se ha realizado con posterioridad a la fecha de entrega prevista (restricción R1), por el contrario, si se ha realizado con anterioridad, el *tardiness* tomará valor 0 (restricción 11). Si la intervención no se realiza en el horizonte temporal, el *tardiness* será la diferencia entre el día posterior al último día del horizonte de planificación (H+1) y la fecha de entrega (restricción R2).

6.2.3.1 Restricciones asociadas a las cirugías (Etapa 1)

La restricción (2) prohíbe que la cirugía del paciente *i* sea programada más de una vez en el horizonte de planificación *H*. Las restricciones (3.1) define la fecha más temprana o *release date* para la realización de la cirugía del paciente *i*, teniendo en cuenta el tiempo necesario para la realización de pruebas médicas previas. La restricción (4) garantiza que la duración total de las intervenciones asignadas al quirófano *j* sea menor o igual que su capacidad en el día *h* (r_{jh}). La restricción (5) prohíbe que la duración total de las cirugías asignadas al cirujano *s* sea mayor que su capacidad en el día *h* (a_{sh}). La restricción (6) limita el número de quirófanos diarios que pueden ser asignados al cirujano *s* en el día *h*. Las restricciones (7.1) y (7.2) definen a que quirófano *j* y día *h* se asigna el cirujano s. No se asignará un cirujano a un quirófano si no hay al menos una cirugía programada en dicho quirófano y día (restricción 7.2). La restricción (8) prohíbe que la intervención *i* sea realizada en un quirófano *j* y día *h* no adecuado para dicha intervención (δ_{ijh}=0). Finalmente, las restricciones (9) y (10) son restricciones binarias para las variables de decisión asociadas a las intervenciones quirúrgicas.

6.2.3.2 Restricciones asociadas a las camas postoperatorias (Etapa 2)

La restricción (1) prohíbe que el paciente *i* ocupe la cama PACU *l* más de un día en el horizonte de planificación H. La restricción (4) asegura que la ocupación de la cama ICU *m* por cada paciente *i* sea programada como máximo una semana en el horizonte de planificación. La restricción (7) limita la ocupación de la cama HOSPITALARIA *n* por cada paciente *i* a una semana en el horizonte de planificación. Las restricciones (2), (5) y (8) prohíbe que la duración total de los pacientes asignados a la cama aplicable en cada caso en el día *h*/semana *w* sea mayor que su capacidad correspondiente (a_{lh}, b_{mw}, c_{nw}). La restricción (3) garantiza que la cama PACU *l* será ocupada por el paciente *i* el día *h* si y solo sí el paciente ha sido intervenido ese día y requiere dicha cama (βi=1 ó βi =2). La restricción (6) garantiza que la cama ICU *m* será ocupada por el paciente *i* en la semana *w*, si dicho paciente ha sido operado en dicha semana y necesita ingresar en la dicha unidad (β_i=3). Las restricciones (9.1) y (9.2) prohíben que la cama HOSPITALARIA *n* sea ocupada por un paciente *i* que no haya ocupado previamente una cama PACU o ICU y no requiera ingreso en cama de hospitalización ($\beta \neq 2$ ó$\beta_i \neq 3$). La restricción (10) garantiza que el paciente *i* no ocupará ninguna cama tras la intervención si no la necesita (β_i=0). Las restricciones (11), (12), (13) aseguran que el paciente *i* solo ocupará

el/los tipo/s de cama que necesite en función de sus necesidades postoperatorias (determinado por el valor de β_i). Por último, las restricciones (14), (15) y (16) son restricciones binarias para las variables de decisión asociadas a las camas postoperatorias.

7 EXPERIMENTACIÓN

En esta sección se lleva a cabo la experimentación de los resultados de los modelos descritos para la resolución del problema. Además, se realiza un análisis de los resultados obtenidos para cada uno de ellos.

7.1 Implementación y optimización de los modelos

Para llevar a cabo la implementación y optimización de los modelos, se ha elegido el programa Microsoft Visual Studio, entorno integrado de desarrollo (IDE) que permite desarrollar aplicaciones, sitios web, aplicaciones webs, etc., para poder ser ejecutadas en multitud de plataformas (Honduras, 2020), en este caso, Microsoft Windows. Es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación (Microsoft, 2019). Más allá del editor y el depurador, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para el proceso de desarrollo de software (Microsoft, 2019). Es compatible con numerosos lenguajes de programación, como por ejemplo C++, C#, Java, Phyton, etc. En este Trabajo de Fin de Grado realizaremos la implementación de los modelos con el lenguaje de programación C#. Una vez implementados, se optimizarán empleando el software comercial de optimización *Gurobi*.

7.1.1 Lenguaje de programación elegido: *C#*

Es un lenguaje de programación caracterizado por una sintaxis básica derivada de los lenguajes de programación C/C++, basado en objetos y con seguridad de tipos. Esta última característica hace que sea imposible leer desde variables no inicializadas, inicializar matrices más allá de sus límites o realizar conversiones de tipo no comprobadas. Además de ser orientado a objetos, es orientado a componentes, permitiendo la definición de propiedades sin necesidad de crear métodos o usar eventos sin tratar con punteros a funciones. Presenta un sistema de tipos unificado, lo que significa que todos los tipos de datos sencillos de C#, derivan de una clase común llamada System.Object, que permite realizar operaciones comunes y que todos los valores de todos los tipos se puedan almacenar, transportar y utilizar de manera coherente. Mejora la gestión de la memoria gracias a la recolección de objetos no usados, que destruye los objetos no empleados en la memoria. Además, el control de excepciones proporciona un enfoque estructurado y extensible para la detección y recuperación de errores. Asimismo, posee otras muchas más características (ver https://lenguajesdeprogramacion.net/c-sharp/).

Si bien todas las características descritas anteriormente hacen C# un lenguaje atractivo para la programación, la elección de C# como lenguaje de programación de este Trabajo de Fin de Grado se debe principalmente al control de versiones ofrecido, que garantiza que los programas y bibliotecas puedan evolucionar con el tiempo de manera compatible con nuevas versiones, y a la similitud de su sintaxis con la del lenguaje C, lenguaje estudiado a lo largo del Grado.

7.1.2 Software de optimización: *Gurobi*

Un software o solucionador de optimización es una herramienta que ayuda a la toma de decisiones relativas a la planificación, asignación y programación de recursos. Integra potentes algoritmos capaces de resolver modelos de programación matemática, programación por restricciones y modelos de programación basados en restricciones (*https://www.ibm.com/ar-es/analytics/optimization-solver*). Algunos ejemplos de solucionadores capaces de optimizar modelos de programación lineal (ILP) son *Visual Math, Gurobi, CPLEX y Lingo*.

Para la optimización de los modelos propuestos, en Trabajo de Fin de Grado se ha empleado el software comercial *Gurobi (9.1.0),* uno de los solucionadores de optimización de problemas de programación lineal más versátiles. Su interfaz de programación de aplicaciones (APIs) es ligera, moderna e intuitiva, facilitando el manejo para usuarios sin experiencia previa. Además, admite una variedad de lenguaje de programación, entre ellos C#.

7.1.3 Elaboración del código de los modelos

En este apartado se describe brevemente el procedimiento seguido para la elaboración del código C# desarrollado para la resolución de los modelos descritos.

En primer lugar se ha elaborado el *Main*, función principal que contiene el cuerpo de la aplicación. En orden de elaboración, encontramos los siguientes elementos:

1. Creación de fichero donde se van a almacenar los resultados de los modelos propuestos.
2. Declaración e iniciación de los factores principales del *testbed* (sección 7.2.1) en forma vectorial.
3. Automatización del proceso de ejecución del código mediante la ejecución de iteraciones. Cada iteración es una combinación única de los factores principales del *testbed*. Dentro de cada iteración, se genera una instancia donde se determinan los datos relevantes a pacientes (sección 7.2.1.1), a cirugías (sección 7.2.1.2), cirujanos (sección 7.2.1.3) y camas postoperatorias (sección 7.2.1.4) y se ejecutan los modelos.
4. Cálculo del tiempo límite de ejecución de cada iteración según la siguiente fórmula:

$$Tiempo\ límite = I \cdot J \cdot H \cdot 0{,}05$$

   La fórmula del tiempo límite desarrollada deriva de la aplicada en (José M. Molina-Pariente, 2015), considerándose el valor del factor tiempo (η =0,05) el doble del propuesto en dicho artículo, debido a que cada uno de los modelos contiene 2 etapas (intervenciones quirúrgicas y camas postoperatorias).
5. Escritura de los resultados de los modelo en el fichero anteriormente creado para su posterior análisis.

En segundo lugar, se han desarrollado las funciones necesarias para la ejecución del *Main*: Función generadora de ficheros y función generadora de parámetros del *testbed*.

En tercer lugar, nos encontramos con los dos modelos desarrollados para la resolución del problema. Ambos tienen la siguiente estructura:

1. Creación del entorno *Gurobi* que recoja los datos asociados a las ejecuciones del modelo de optimización.
2. Creación del modelo donde se almacenarán el conjunto de variables, restricciones y atributos asociados al problema.
3. Declaración de las variables de *Gurobi*, que son las variables del modelo ILP.
4. Determinación de la función objetivo y su sentido (si es de maximizar o minimizar).
5. Determinación de las restricciones del modelo ILP.
6. Indicación del tiempo límite de ejecución para la instancia determinada.
7. Llamada a la función de optimización (*model.Optimize*).
8. Recogida de los resultados de la optimización de los modelos y transferencia al *Main*.
9. Liberación de los recursos asociados con el modelo y entorno.

Además, dada la naturaleza del problema donde las variables relacionadas con el tipo de cama PACU están en días y las relacionadas con las camas ICU y hospitalarias en semanas, se ha creado una función (función *correspondencia_día_semana*) que asocia los días con la semana a la que pertenecen, asociación necesaria para la ejecución de determinadas restricciones de los modelos.

7.2 Generación de las instancias del problema

Para la generación del *testbed* o banco de pruebas se ha tomado como referencia el artículo "*New heuristics for planning operating rooms*" (José M. Molina-Pariente, 2015), adaptando el método de generación de instancias propuesto e incorporando datos particulares que son de aplicación a este modelo. En este caso, las camas postoperatorias deben ser cuidadosamente determinadas e incorporadas a las instancias de prueba.

7.2.1 Factores principales

Los principales factores a considerar para la generación de un banco de pruebas según las contribuciones existentes en el ámbito de la planificación de quirófanos y las características del problema descrito son:

- |**H**|: número de días en el horizonte temporal
- |**J**|: número de quirófanos
- |**L**|: número de camas PACU. Tras la revisión de la literatura se ha determinado el número de camas PACU en función del número de quirófanos del problema, siendo un 50% y un 100% del mayor número de quirófanos establecidos.
- |**M**|: número de camas ICU. Al igual que en el caso anterior, se ha establecido en un 200% del número de quirófanos establecidos.
- |**N**|: número de camas hospitalarias. En las contribuciones literarias no se ha encontrado referencia acerca del número de camas de hospitalización, por ello, teniendo en cuenta las dimensiones de la Unidad de Cirugía Plástica y Grandes Quemados, se ha establecido de manera proporcional en un 500% del número de quirófanos.
- |**I**|: número de pacientes en la lista de espera. El número se puede establecer en función del tamaño de una lista de espera real, de forma arbitraria por el autor o generando el número de pacientes uno a uno hasta que la suma de sus tiempos esperados en el sistema supere un β% de la capacidad total disponible de los recursos en el horizonte temporal, siendo los recursos las distintas camas y los quirófanos (José M. Molina-Pariente, 2015).

  La duración estimada de las cirugías en la lista de espera suele exceder la capacidad disponible debido a los presupuestos restrictivos. Por ello, se ha optado por un 100% y un 125% de la capacidad para el factor β.
- |**mds**|: número de días a la semana en los que un cirujano está disponible.
- |**S**|: número de cirujanos. Al igual que el número de pacientes en la lista de espera, este valor puede ser establecido según datos reales, de forma arbitraria por el autor o siguiendo la fórmula del artículo citado anteriormente que se describe a continuación (José M. Molina-Pariente, 2015):

$$|S| = \alpha \cdot \frac{\sum_{j \in J} \sum_{h \in H} r_{jh}}{W \cdot a \cdot mds}$$

  La fórmula no presenta modificación alguna con respecto a la original porque en este modelo, la segunda etapa es referente a la asignación de los pacientes a las camas, en las cuales no es necesaria la presencia de cirujano, por lo cual la capacidad de las camas no es un factor a considerar para la determinación del número de cirujanos del modelo.

  El factor de control α se ha establecido en 1.5 y 2, de forma que como mínimo, cada quirófano esté ocupado siempre por un cirujano.
- $\mathbf{u_s}$: Número entero no negativo que determina el número de quirófanos distintos donde el cirujano s puede llevar a cabo cirugías en el mismo día.

En la tabla se recogen los valores asignados para cada uno de los factores basados en el artículo mencionado en la introducción de la sección.

Tabla 7-1: Factores considerados en la literatura para el diseño de un *testbed*

|H|	|J|	|L|	|M|	|N|	|mds|	$\mathbf{u_s}$	α	β%
5	3,4	2,4	6,8	24, 33	4	1, |J|	1.5, 2	100, 125

7.2.2 Datos referentes a pacientes

Los parámetros de los pacientes requeridos para el problema se recogen en la tabla 4.

Tabla 7-2: Parámetros de pacientes considerados para la generación de datos de entrada

t_i	w_i	rd_i	d_i	y_i	$t_{i,pacu}$	$t_{i,icu}$	$t_{i,hospitalaria}$
U[90,120]	$0{,}5 \cdot \frac{mp}{5} + 0{,}5 \cdot \frac{dwl}{MTBT}$	U[1,3]	$MTBT - dwl$	U[1,|S|]	U[1,3]	U[12,36]	U[1,168]

- $\mathbf{t_i}$: El tiempo estimado de la cirugía del paciente i se obtiene mediante una distribución uniforme discreta dados unos valores máximos y mínimos (en minutos) establecidos de forma arbitraria.
- $\mathbf{w_i}$: El peso clínico de la cirugía de cada paciente i se obtiene en el artículo "*New heuristics for planning operating rooms*" (José M. Molina-Pariente, 2015) como una combinación lineal de los valores normalizados de la prioridad clínica del paciente (*mp*) y el número de días del paciente en la lista de espera (dwl).

$$w = a \cdot \frac{mp}{mp_{max}} + (1 - a)\frac{dwl}{dwl_{max}}$$

  Donde la prioridad médica se genera de forma aleatoria en el intervalo [1, 5], siendo 5 la prioridad máxima (mp_{max}), el número de días del paciente en la lista de espera aleatoriamente en el intervalo [1, MTBT -1], siendo MTBT el máximo tiempo de tratamiento antes de la cirugía considerado por la especialidad (dwl_{max}) y *a* el peso de cada uno de los indicadores, que asumimos que es el mismo en ambos casos (a = 0,5) (José M. Molina-Pariente, 2015).

  La expresión del peso clínico de la cirugía de cada paciente *i* quedaría:

$$w = 0{,}5 \cdot \frac{mp}{5} + 0{,}5\ \frac{dwl}{MTBT}$$

- $\mathbf{rd_i}$: Se establece de forma aleatoria en el intervalo [1,3] el tiempo requerido para la realización de pruebas necesarias y previas a la cirugía (Ramírez Rojas, 2020).
- $\mathbf{d_i}$: En la literatura se ha calculado este valor como la diferencia entre el máximo tiempo antes de tratamiento (MTBT) y el número de días en la lista de espera (*dwl*). Los máximos tiempos antes de tratamiento considerados por la especialidad son 45, 180 y 360 días, definidos por el Servicio Nacional de Salud según la gravedad (José M. Molina-Pariente, 2015).
- $\mathbf{y_i}$: Toma un valor aleatorio entre 1 y S, asignando un cirujano arbitrario del conjunto S.
- $\boldsymbol{\delta_{ijh}}$: Para determinar el valor del parámetro para cada paciente, se debe considerar que en la lista de espera de la unidad el 10% de las cirugías requieren quirófanos especializados, mientras que el resto pueden ser programadas en cualquier quirófano, ya sea multifuncional o especializado (José M. Molina-Pariente, 2015). En la unidad, el 30% de los quirófanos están especializados.
- $\mathbf{t_{i,pacu}}$: El tiempo estimado del paciente i tras la cirugía en Unidad de Recuperación Post Anestésica oscila entre 1 y 3 en función de la anestesia requerida por la intervención.
- $\mathbf{t_{i,icu}}$: El tiempo estimado del paciente i tras la cirugía en la Unidad de Cuidados Intensivos está comprendido entre 12 y 36 horas.
- $\mathbf{t_{i,hospitalaria}}$: El tiempo estimado del paciente i en la cama de hospitalización está comprendido entre 1 y 168, siendo 168 el máximo tiempo posible de permanencia en la cama de hospitalización.
- $\boldsymbol{\beta_i}$: Puede valer 0, 1, 2, 3 en función de la necesidad de atención médica del paciente tras la cirugía. De forma más detallada, se define a continuación que valor tomará en cada caso:

- β_i=0, cuando el paciente no necesite ingresar en la Unidad de Recuperación Post Anestésica tras la cirugía.
- β_i=1, cuando el paciente ingrese en la Unidad de Recuperación Post Anestésica tras la cirugía.
- β_i=2, cuando el paciente ingrese en la Unidad de Recuperación Post Anestésica tras la cirugía, y posteriormente ingrese en una planta de hospitalización.
- β_i=3, cuando el paciente ingrese en la Unidad de Cuidados Intensivos tras la cirugía. Una vez que el paciente sea dado de alta en dicha unidad, ingresará en una planta de hospitalización.

Se determinará como sigue: Entre el 5- 10% de los pacientes intervenidos ingresarán en la Unidad de Cuidados Intensivos (β=3). Del resto de pacientes, entre el 50 y el 100% ingresarán en la Unidad de Cuidados Post Anestésica (β=1 y β=2), de los cuales solo el 10% ingresarán en cama hospitalaria tras la recuperación en cama PACU (β=2), y el porcentaje restante de pacientes intervenidos no necesitarán ingresar en ninguna de las unidades tras la intervención (β=0).

7.2.3 Datos referentes a cirujanos

En cuanto a los cirujanos, semanalmente se elabora una planificación semanal donde se especifica que cirujanos están disponibles cada día. Con el fin de evitar que un quirófano disponible esté inutilizado, se genera la programación semanal mediante el procedimiento de dos pasos (José M. Molina-Pariente, 2015):

1. Para cada día, asignar de forma aleatoria tantos cirujanos, pertenecientes al conjunto S, como quirófanos disponibles existan. De esta manera se garantiza la presencia de al menos un cirujano en cada quirófano.
2. Calcular la diferencia entre el número de días a la semana en los que el cirujano está disponible (mds) y el número de días asignados en la etapa anterior, añadiendo esta diferencia a cada cirujano de forma aleatoria de manera que realice cirugías todos los días en los que está disponible.

La capacidad de un cirujano *s* para realizar cirugías un día *h* (a_{sh}) dependerá de la disponibilidad del cirujano, que será igual a la capacidad del quirófano (r_{jh}), mientras que por el contrario si no trabaja se reduce a 0, o lo que es lo mismo, no está disponible. La capacidad regular de cada uno de los quirófanos para todos los días (r_{jh}) será de 8 horas.

7.2.4 Datos referentes a camas postoperatorias

En cuanto a cada uno de los tipos de cama, las capacidades se establecen como sigue:

Capacidad cama PACU: 24h/día

Capacidad cama ICU y Hospitalaria: 168h/semana

7.3 Evaluación de las alternativas

Establecidos los factores y parámetros del problema, en esta sección se evaluará la actuación de los dos modelos ILP propuestos en cada instancia generada. La generación de instancias se llevará a cabo conforme al procedimiento descrito en la sección 7.2.1, considerando las 128 combinaciones posibles de los factores β, α, |J|, |L|, |M|, |N| y u_s. Para cada combinación, se generarán 2 iteraciones, resultando en total 256 instancias objeto de análisis. El tamaño de la lista de espera dependerá de la combinación de los factores (|J|, β) (José M. Molina-Pariente, 2015), siendo 69, 92, 86 y 115 el número medio de intervenciones quirúrgicas para la elección de los factores (3, 1.00), (4, 1.00), (3, 1.25) y (4, 1.25) respectivamente. La experimentación se ha llevado a cabo en un ordenador con un procesador Intel Core i7-8550U y 8 GB de memoria RAM.

Es importante entender que cuando se resuelve un modelo de optimización, el algoritmo proporciona una cota inferior y superior (*Lower Bound y Upper Bound*) del valor óptimo. Para un modelo de maximización, el *lower bound* proporciona el valor de la solución factible obtenida, mientras que el *upper bound* proporciona un límite al valor óptimo de la función objetivo. El *GAP* es un indicador de la proximidad de la solución encontrada con respecto a la solución óptima que toma valores en el intervalo [0,1], siendo 0 en el mejor de los

casos donde la solución obtenida es la óptima, y aumentando su valor a medida que aumenta la diferencia entre la solución factible obtenida y la cota superior (*Upper Bound*).

Para cada una de las instancias, se ejecutarán los dos modelos de planificación planteados, obteniéndose para cada uno de ellos (en el caso del primer modelo para cada una de las etapas de planificación) si la solución ha sido hallada de manera óptima, factible o infactible, el tiempo de ejecución y los valores de las variables *óptimo*, *bound* y *gap* (detallado en la tabla 7-3). Si la solución hallada es la óptima, la variable *óptimo* tomará el valor de la función objetivo en el punto óptimo, y el *bound* o cota será también el valor óptimo. Si la solución encontrada es infactible, todas las variables tomarán valor 0. Por último, si la solución obtenida es una solución factible, la variable *óptimo* tomará el valor de la mejor solución encontrada durante la ejecución (*best lower bound*), y el *bound* será la cota superior o *upper bound*.

Tabla 7-3: Posibles valores de las variables tras la optimización

	Solución Óptima	**Solución Factible**	**Solución Infactible**
Óptimo	Valor Óptimo	Best (Lower) Bound	0
Bound	Valor Óptimo	Upper Bound	0
Gap (%)	0	$\frac{Upper\ Bound - Best\ Bound}{Best\ Bound} \cdot 100$	0

El número de cirujanos |S| (detallado en la tabla 7-4), se determina siguiendo la fórmula descrita en la sección 7.2 (generación de instancias), donde todos los factores empleados en la fórmula, exceptuando |J| y α, tomarán el mismo valor en cada una de las instancias.

Tabla 7-4: Número de cirujanos en función de α y |J|

α	**|J|**	**|S|**
1.5	3	**6**
	4	**8**
2	3	**8**
	4	**10**

La programación semanal de los cirujanos, descrita en la sección 7.2.3 (datos referentes a cirujanos), dependerá en cada instancia del número de quirófanos establecidos |J| y del factor de control α. Se realiza en dos pasos (descrito en la sección 7.2.3), un primer paso donde se realiza una programación inicial donde se asignan diariamente tantos cirujanos como quirófanos disponibles, y un segundo paso que adicionalmente garantiza que cada cirujano estará disponible *mds* días semanales.

Las tablas 7-5, 7-6, 7-7 y 7-8 recogen la programación inicial realizada en el primer paso.

Tabla 7-5: Programación semanal inicial para α=1.5 y |J| =3

Cirujano/día	**0**	**1**	**2**	**3**	**4**
0	1	0	1	1	0
1	1	0	0	1	1
2	1	1	1	0	0
3	0	1	0	1	0
4	0	1	0	0	1
5	0	0	1	0	1

Tabla 7-6: Programación semanal inicial para α=1.5 y |J| =4

Cirujano/día	0	1	2	3	4
0	1	0	1	0	1
1	1	0	1	0	1
2	0	1	1	1	0
3	1	1	0	0	0
4	0	0	0	1	0
5	0	1	1	1	1
6	1	0	0	0	0
7	0	1	0	1	1

Tabla 7-7: Programación semanal inicial para α=2 y |J| =3

Cirujano/día	0	1	2	3	4
0	1	0	1	1	0
1	1	0	0	1	0
2	0	0	1	0	1
3	1	1	0	0	0
4	0	0	0	0	0
5	0	1	0	1	1
6	0	1	0	0	0
7	0	0	1	0	1

Tabla 7-8: Programación semanal inicial para α=2 y |J| =4

Cirujano/día	0	1	2	3	4
0	0	0	1	0	1
1	1	0	1	0	0
2	1	0	1	1	0
3	0	1	0	1	0
4	1	1	0	0	0
5	0	0	0	0	0
6	0	1	1	1	1
7	1	0	0	0	1
8	0	0	0	0	0
9	0	1	0	1	1

Las tablas 7-9, 7-10, 7-11 y 7-12 recogen la programación semanal tras la realización del segundo paso.

Tabla 7-9: Programación semanal para α=1.5 y |J| =3

Cirujano/día	0	1	2	3	4
0	1	1	1	1	0
1	1	1	0	1	1
2	1	1	1	1	0
3	1	1	1	1	0
4	1	1	1	0	1
5	1	1	1	0	1

Tabla 7-10: Programación semanal para α=1.5 y |J| =4

Cirujano/día	0	1	2	3	4
0	1	1	1	0	1
1	1	1	1	0	1
2	1	1	1	1	0
3	1	1	1	1	0
4	1	1	1	1	0
5	0	1	1	1	1
6	1	1	1	1	0
7	1	1	0	1	1

Tabla 7-11: Programación semanal para α=2 y |J| =3

Cirujano/día	0	1	2	3	4
0	1	1	1	1	0
1	1	1	1	1	0
2	1	1	1	0	1
3	1	1	1	1	0
4	1	1	1	1	0
5	1	1	0	1	1
6	1	1	1	1	0
7	1	1	1	0	1

Tabla 7-12: Programación semanal para α=2 y |J| =4

Cirujano/día	0	1	2	3	4
0	1	1	1	0	1
1	1	1	1	1	0
2	1	1	1	1	0
3	1	1	1	1	0
4	1	1	1	1	0
5	1	1	1	1	0
6	0	1	1	1	1
7	1	1	1	0	1
8	1	1	1	1	0
9	1	1	0	1	1

7.3.1 Análisis de los resultados

Conocidos el número de cirujanos y la programación semanal para cada instancia en función de los parámetros α y |J|, los resultados de la experimentación para los dos modelos de planificación de intervenciones quirúrgicas y camas postoperatorias se muestran en esta sección. El valor medio de las variables *GAP* y *tiempo de ejecución* se obtiene haciendo la media aritmética de los valores obtenidos en las instancias con solución factible u óptima. El tiempo límite de ejecución de la **etapa 1** del modelo ILP (1) será el propuesto en la sección 7.1.3 dada la complejidad de resolución de la etapa. En cuanto a la **etapa 2** del modelo ILP (1) el tiempo límite de ejecución será la mitad del propuesto en la sección 7.1.3. El modelo ILP (2) tendrá el tiempo límite de ejecución total propuesto en la sección 7.1.3. A pesar de la aparente diferencia del tiempo de ejecución total de ambos modelos debido a los límites de tiempo de ejecución establecidos, el tiempo de ejecución total de ambos modelos será (prácticamente) el mismo debido al (casi) insignificante tiempo de ejecución de la **etapa 2** del modelo ILP (1) resuelto en aproximadamente el 0,13% del tiempo de ejecución total de dicho modelo (detallado en la tabla 7-13).

Tabla 7-13: Tiempo medio de ejecución de los dos modelos propuestos para el problema de decisión

Problema	Tiempo medio ejecución *Etapa 1* (s)	Tiempo medio ejecución *Etapa 2* (s)	Tiempo medio total de ejecución (s)
Modelo ILP(1): **Por etapas**	80,84	0,11	80,95
Modelo ILP(2): **Integración**	79,11		79,11

En la tabla 7-14 se calcula el valor del tiempo límite siguiendo la fórmula definida en la sección 7.1.3 función de los factores |H|, |I| y |J|.

Tabla 7-14: Cálculo del tiempo límite en función de H, J e I

|H|	|J|	|I|	Tiempo límite (s)
5	3	69	51,75
		86	64,5
		92	69
		115	86,25
	4	69	69
		86	86
		92	92
		115	115

Para evaluar la efectividad de los modelos propuestos se empleará la Desviación Porcentual Relativa (*RPD*), variable de respuesta empleada de manera común en la literatura (ver por ejemplo en (José M. Molina-Pariente, 2015) y (José. M. Molina-Pariente, 2015)), acorde con la expresión $RPD = (Best_{sol} \cdot Heu_{sol})/Best_{sol} \cdot 100$, donde $Best_{sol}$ es la mejor solución encontrada para la instancia dada teniendo en cuenta todos los algoritmos empleados y Heu_{sol} es la solución dada por un algoritmo determinado.

7.3.1.1 Influencia del número de quirófanos y camas postoperatorias

En esta sección se recogen los datos de interés para el análisis de los resultados experimentales del modelo para la combinación de quirófanos, por considerarse el recurso más limitante en la literatura (José M. Molina-Pariente, 2015), y camas postoperatorias, por considerarse los recursos postoperatorios limitantes en la planificación de intervenciones quirúrgicas, especialmente en la actualidad debido a la saturación que en ellos ocasiona la actual crisis sanitaria con motivo de la COVID-19.

Las tablas 7-15 y 7-16 recogen en ambos modelos, para la combinación de cada nivel del factor |J|, con cada nivel de cama postoperatoria, PACU (|L|), ICU (|M|) y hospitalaria (|N|) respectivamente, el GAP mínimo, máximo y medio, el porcentaje de soluciones óptimas, factibles e infactibles, así como el porcentaje de desviación porcentual relativa (*RPD*) y tiempo de ejecución para su posterior análisis.

Tabla 7-15: Actuación del modelo ILP (1) para la resolución del problema de decisión

Problema	|J|	|L|	|M|	|N|	Min. GAP (%)	Max. GAP (%)	GAP medio (%)	Soluciones óptimas (%)	Soluciones factibles (%)	Soluciones infactibles (%)	RPD (%)	Tiempo de ejecución (s)
Modelo ILP (1): **Por etapas**	3	2	6	15	0,0054	0,0277	0,0159	0	100	0	13,8787	58,2339
				20	0,0054	0,0277	0,0152	0	100	0	13,7477	58,2252
			8	15	0,0054	0,0277	0,0152	0	100	0	13,7512	58,2161
				20	0,0054	0,0277	0,0152	0	100	0	13,7488	58,2245
		4	6	15	0,0054	0,0280	0,0153	0	100	0	13,7529	58,2284
				20	0,0054	0,0277	0,0152	0	100	0	13,7494	58,2209
			8	15	0,0054	0,0277	0,0152	0	100	0	13,7500	58,2244
				20	0,0054	0,0277	0,0152	0	100	0	13,7514	58,2255
	4	2	6	15	0,0034	0,0106	0,0070	0	100	0	14,0627	103,6686
				20	0,0034	0,0106	0,0070	0	100	0	13,9675	103,6754
			8	15	0,0034	0,0106	0,0070	0	100	0	14,0619	103,6695
				20	0,0034	0,0106	0,0070	0	100	0	14,0611	103,6673
		4	6	15	0,0034	0,0106	0,0070	0	100	0	14,1567	103,6732
				20	0,0034	0,0106	0,0070	0	100	0	14,0628	103,6758
			8	15	0,0034	0,0106	0,0070	0	100	0	13,9653	103,6735
				20	0,0034	0,0106	0,0070	0	100	0	13,9675	103,6728
	Average				0,0044	0,0192	0,0112	0	100	0	13,9022	80,9484

Todas las soluciones obtenidas en el modelo ILP(1) son factibles, sin embargo, estas soluciones son muy próximas a la solución óptima en cada caso, lo cual podemos comprobar mediante la variable *GAP medio* (indicador de proximidad de la solución obtenida con respecto a la óptima), cuya media toma el valor de 0,0112%, siendo en el mayor el *GAP medio*, es decir, la mayor distancia posible entre la solución factible

obtenida y la posible solución óptima es de un 0,0192%. Luego, teniendo en cuenta el tiempo límite de ejecución establecido para este modelo (descrito en la sección 7.3.1) la no obtención de una solución óptima en todos los casos se debe al alcance del tiempo límite de ejecución del modelo ILP(1) en la **Etapa 1** en cada instancia.

Si bien procedemos al análisis de la influencia de los parámetros |J|, |L|, |M| y |N| podemos observar que el tiempo de ejecución aumenta conforme aumenta el valor del parámetro |J|, debido a que como ya se ha comentado en la sección 7.3.1, el tiempo de ejecución total se debe en un más de un 99% a la resolución de la **Etapa 1**, siendo la contribución de la resolución de la **Etapa 2** al tiempo total de ejecución del modelo ILP(1) prácticamente nula, y por tanto, la influencia de los parámetros relacionados con esta segunda etapa también lo es.

Tabla 7-16: Actuación del modelo ILP (2) para la resolución del problema de decisión

Problema	**|J|**	**|L|**	**|M|**	**|N|**	**Min. GAP (%)**	**Max. GAP (%)**	**GAP medio (%)**	**Soluciones óptimas (%)**	**Soluciones factibles (%)**	**Soluciones infactibles (%)**	**RPD (%)**	**Tiempo de ejecución (s)**
Modelo ILP (2): **Integrado**	3	2	6	15	0,0127	0,0655	0,0261	0	100	0	0	58,1471
				20	0,0126	0.0638	0,0317	0	100	0	0	58,1542
			8	15	0,0141	0,0504	0,0284	0	100	0	0	58,1512
				20	0	0,0552	0,0313	12,5	87,5	0	0	54,0065
		4	6	15	0,0103	0,0580	0,0256	0	100	0	0	58,1496
				20	0	0,0578	0,0307	25	75	0	0	53,1338
			8	15	0,0166	0,0546	0,0269	0	100	0	0	58,1492
				20	0	0,0555	0,0293	25	75	0	0	53,1582
	4	2	6	15	0	2,8231	0,4783	25	75	0	0	98,6552
				20	0,0106	2,8312	0,6008	0	100	0	0	103,5366
			8	15	0	2,8279	0,4809	12,5	87,5	0	0	103,0264
				20	0,0134	1,8798	0,4822	0	100	0	0	103,5433
		4	6	15	0	1,8692	0,3601	12,5	87,5	0	0	102,0291
				20	0	2,8165	0,4805	12,5	87,5	0	0	100,4130
			8	15	0	2,8331	0,6094	12,5	87,5	0	0	100,4701
				20	0	3,8039	0,6030	12,5	87,5	0	0	102,9778
	Average				0,0056	1,4721	0,2703	9,3750	90,6250	0	0	79,1063

Para el segundo modelo de resolución propuesto, el modelo ILP(2), el porcentaje de soluciones óptimas obtenidas en todas las instancias apenas alcanza el 9,375%, siendo en el resto de los casos soluciones factibles. Sin embargo, dichas soluciones factibles obtenidas son muy próximas a la posible solución óptima, esto lo podemos ver mediante el indicador *GAP medio*, cuya media es de 0,27%, siendo en el peor de los casos de 1,47%, lo que es también muy bajo. Al igual que en el modelo anterior, el no alcance de la solución óptima se debe en la mayoría de los casos al alcance del tiempo límite de ejecución establecido para el modelo (descrito en la sección 7.3.1).

En cuanto a la evaluación de la efectividad de los modelos propuestos, podemos concluir mediante el análisis de la desviación porcentual relativa (*RPD*) realizado considerando la influencia de los quirófanos y camas postoperatorias, que el modelo ILP (2) es el más efectivo entre los propuestos en todas las instancias.

Si bien analizamos aquellas instancias donde se han hallado soluciones óptimas y sus respectivos porcentajes, podemos observar la influencia de los parámetros |J|, |L|, |M|, |N| en dos partes como se detalla a continuación:

- Para |J|=3 se han encontrado soluciones óptimas en tres conjuntos de instancias, siendo el valor del parámetro |N| =20 en los tres casos. Luego podríamos afirmar en este caso que a mayor valor del parámetro |N|, mayor porcentaje de soluciones óptimas encontradas. En cuanto al parámetro |M|, no se aprecia relación de sus posibles valores con respecto al porcentaje de soluciones óptimas encontradas. Por último, en cuanto a la influencia del parámetro |L|, podríamos afirmar un mayor porcentaje de soluciones óptimas encontradas conforme su valor aumenta.
- Para |J|=4 se han encontrado soluciones óptimas en todas las instancias, siendo el porcentaje de soluciones óptimas en todas ellas idéntico. Sin embargo, teniendo en cuenta las influencias de los parámetros |L|, |M| y |N| encontradas en el apartado anterior donde |J|=3, no se evidencia que esto ocurra de la misma manera en estas instancias, ya que el valor porcentual de soluciones óptimas obtenidas es idéntico en todos los casos independientemente del valor que tomen estos parámetros.

Luego, la única hipótesis de la cual no existen evidencias para rechazarla en este modelo propuesto, es que conforme aumenta el valor de |J|, aumenta el número de conjuntos de instancias (clasificadas en función de |J|, |L|, |M| y |N| tal y como se detalla en la tabla 7-16) en las que se encuentran soluciones óptimas.

7.3.2 Influencia del parámetro u_s

En esta sección se pretende analizar la influencia del parámetro u_s que determina la movilidad del cirujano. Se realizará una programación avanzada (José M. Molina-Pariente, 2015) en aquellas iteraciones donde el parámetro permita la movilidad diaria del cirujano por todos los quirófanos (u_s= |J|). Por el contrario, en el resto de iteraciones donde la movilidad del cirujano quedará reducida a un quirófano diario (u_s=1) se realizará la programación mediante un enfoque integrado.

Tabla 7-17: Influencia del parámetro u_s en los modelos propuestos para el problema de decisión

Problema	|J|	u_s	Min. GAP (%)	Max. GAP (%)	GAP medio (%)	Soluciones óptimas (%)	Soluciones factibles (%)	Soluciones infactibles (%)	RPD (%)	Tiempo de ejecución (s)
Modelo ILP (1): **Por etapas**	**3**	**1**	0,0054	0,0273	0,0155	0	100	0	14,3487	58,2266
		|J|	0,0084	0,0280	0,0151	0	100	0	13,1838	58,2231
	4	**1**	0,0080	0,0106	0,0091	0	100	0	14,8412	103,6693
		|J|	0,0034	0,0066	0,0048	0	100	0	13,2351	103,6747
	Average		0,0063	0,0181	0,0111	0	100	0	13,9022	80,9484
Modelo ILP (2): **Integrado**	**3**	**1**	0	0,0655	0,0310	6,25	93,75	0	0	58,1140
		|J|	0	0,0565	0,0265	9,375	90,625	0	0	54,6485
	4	**1**	0,0138	3,8039	0,7261	0	100	0	0	103,5390
		|J|	0	1,8655	0,2977	21,875	78,125	0	0	100,1239
	Average		0,0035	1,4479	0,2703	9,3750	90,6250	0	0	79,1064

La efectividad del modelo ILP(2) propuesto es mayor en todas las instancias que en el modelo ILP(1) independientemente de los valores que tomen los parámetros de influencia estudiados, en este caso |J| y u_s, logrando un valor nulo en la desviación relativa porcentual (*RPD*) en cada una de las instancias ejecutadas.

Al igual que en el apartado anterior, todas las soluciones obtenidas por el modelo ILP(1) son factibles, esto se debe al alcance del tiempo límite especificado para en la **Etapa 1** en la sección 7.3.1, que impide alcanzar soluciones óptimas pero que, sin embargo, encuentra soluciones factibles muy próximas al óptimo en todas las instancias (El valor medio de la variable *GAP medio* es de 0,0111%). Por otro lado, en el modelo ILP(2) encontramos de media 9,375% de soluciones óptimas, un valor bajo con respecto al total. No obstante, el resto de soluciones encontradas son factibles y muy cercanas a la solución óptima, detallado en la variable *GAP medio* que toma un valor medio de 0,27% para todas las instancias ejecutadas en este modelo.

Por otro lado, podemos observar en la tabla 7-17 que las soluciones óptimas en el modelo ILP(2) se encuentran en mayor medida en aquellas instancias que permiten la movilidad diaria del cirujano por todos los quirófanos (u_s =|J|), mejorando el porcentaje de soluciones óptimas cuanto mayor sea el número de quirófanos (|J|) establecidos en la instancia.

7.3.3 Influencia de la lista de espera y número de cirujanos

El tamaño de la lista de espera, que excede en un β% la capacidad total de los recursos postoperatorios debido a los presupuestos restrictivos y el número de cirujanos, determinado de manera que en cada quirófano (recurso más costoso) haya siempre al menos un cirujano (α) ejercen una influencia en el valor de la función objetivo y en la efectividad (si la solución es óptima, factible o infactible) de los modelos propuestos. Por ello, en la tabla 7-18 se recogen datos de interés para el análisis de la influencia de estos parámetros en los dos modelos propuestos.

Tabla 7-18: Influencia de α y β en los modelos propuestos para el problema de decisión

Problema	α	β(%)	Min. GAP (%)	Max. GAP (%)	GAP medio (%)	Soluciones óptimas (%)	Soluciones factibles (%)	Soluciones infactibles (%)	RPD (%)	Tiempo de ejecución (s)
Modelo ILP (1): **Por etapas**	**1.5**	**100**	0,0034	0,0273	0,0108	0	100	0	7,4468	71,9924
		125	0,0066	0,0253	0,0124	0	100	0	20,7166	89,8056
	2	**100**	0,0044	0,0280	0,0114	0	100	0	8,8958	71,9940
		125	0,0049	0,0125	0,0099	0	100	0	18,5497	89,9018
	Average		0,0048	0,0233	0,0111	0	100	0	13,9022	80,9235
Modelo ILP (2): **Integrado**	**1.5**	**100**	0	0,0655	0,0245	6,25	93,75	0	0	71,4874
		125	0	1,0569	0,0608	3,125	96,875	0	0	89,6577
	2	**100**	0	0,0565	0,0162	18,75	81,25	0	0	69,0003
		125	0	3,8039	0.9798	9,375	90,625	0	0	86,2801
	Average		0	1,2457	0,0338	9,3750	90,6250	0	0	79,1064

Al igual que en los apartados anteriores, el modelo ILP(2) es el modelo más efectivo entre los propuestos debido a su nula desviación porcentual relativa (RPD) en todos las instancias, siendo el modelo que proporciona la mejor solución para cada una de las instancias entre los modelos propuestos. Tal y como ocurre en los apartados anteriores, la mayoría de soluciones encontradas en el modelo ILP(2) y todas las soluciones encontradas en el modelo ILP(1) son soluciones factibles muy próximas al óptimo (tal y como indica la variable *GAP medio*) debido al alcance del tiempo límite de ejecución en ambos casos.

En cuanto a la influencia de los parámetros α y β, podemos afirmar que en el modelo ILP(2) las soluciones óptimas se encuentran en mayor medida en aquellas instancias con mayor valor de α (α =2), mejorando el porcentaje de soluciones óptimas encontradas cuanto menor sea el valor de β (β=100).

Si bien recordamos la definición del tiempo límite de ejecución detallada en la sección 7.1.3, podemos ver que el tiempo límite de ejecución de cada modelo depende del tamaño de la lista de espera (|I|) y que esta, a su vez, está determinada en función del parámetro β. Luego podemos concluir que para ambos modelos, el tiempo de ejecución será menor en aquellas instancias cuyo valor de β sea menor (β=100).

7.4 Ejemplo de aplicación: Unidad de Cirugía Plástica y Grandes Quemados (HUVR)

Una vez estudiada la experimentación en el banco de pruebas propuesto y comprobada la eficacia de los modelos propuestos, en esta sección se procede a estudiar la solución ofrecida por cada uno de los modelos propuestos para el caso particular de la Unidad de Cirugía Plástica y Grandes Quemados del Hospital Universitario Virgen del Rocío. Para ello, es necesario considerar las limitaciones de los recursos operatorios y postoperatorios de la Unidad de Gestión Clínica, descritos en la sección 5.1, y el entorno del problema de decisión, descrito en la sección 5.2.

7.4.1 Datos principales

La unidad de Cirugía Plástica y Grandes Quemados cuenta con 3 quirófanos, 3 camas PACU, 12 camas ICU y 19 camas hospitalarias. El resto de los recursos perioperatorios se consideran disponibles. La movilidad de cada cirujano se determina de forma aleatoria en el intervalo (0, |J|). Tal y como se describe en la sección 5.1, todos los quirófanos estarán disponibles en el turno de mañana 6 horas (de 8:30 a 14:30). En el turno de tarde, dos quirófanos estarán disponibles de lunes a jueves 4 horas (de 15:30 a 19:30) y el otro quirófano lo estará en el mismo horario lunes, martes y jueves. En cuanto al número de cirujanos, teniendo en cuenta la capacidad de los quirófanos del problema de decisión abordado, se establecen 4 cirujanos. Por último, con el objetivo de analizar la solución ofrecida por los dos modelos, se establece el tamaño de la lista de espera en 20 pacientes.

Tabla 7-19: Datos principales del ejemplo de aplicación

|I|	|S|	|H|	|J|	|L|	|M|	|N|	|mds|
15	4	5	3	4	12	19	4

7.4.2 Datos referentes a pacientes

En esta sección se generan los datos referentes a los pacientes de la Unidad de Cirugía Plástica y Grandes Quemados del Hospital Universitario Virgen del Rocío tal y como se describe en la sección 7.2.2 y considerándose la naturaleza del entorno del ejemplo de aplicación.

Tabla 7-20: Datos referentes a pacientes del ejemplo de aplicación

PACIENTE	t_i	w_i	y_i	$especializada_i$	rd_i	d_i	β_i	$t_{i,pacu}$	$t_{i,icu}$	$t_{i,hospitalaria}$
1	101	0,56	1	-	2	31	2	3	0	38
2	117	0,78	2	-	2	160	1	1	0	0
3	92	0,41	2	-	1	355	0	0	0	0
4	99	0,93	3	-	3	51	1	1	0	0
5	90	0,57	1	-	1	21	1	3	0	0
6	93	0,84	3	-	3	14	0	0	0	0
7	92	0,64	3	-	3	59	1	2	0	0
8	95	0,42	3	-	2	272	0	0	0	0
9	91	0,39	4	-	2	149	1	3	0	0
10	114	0,6	3	-	2	109	1	1	0	0
11	112	0,54	4	-	3	41	1	2	0	0
12	101	0,62	2	-	3	130	0	0	0	0
13	101	0,21	1	-	1	35	1	1	0	0
14	97	0,48	1	-	1	20	1	3	0	0
15	110	0,93	2	-	2	6	1	1	0	0
16	111	0,25	2	SÍ	3	323	1	2	0	0
17	110	0,78	3	-	1	2	1	1	0	0
18	108	0,87	3	-	1	10	2	1	0	152
19	95	0,47	3	-	1	30	2	2	0	15
20	93	0,48	1	-	1	29	0	0	0	0

Aquellos pacientes cuya intervención quirúrgica haya sido catalogada como "especializada" solo podrán ser intervenidos en el quirófano especializado, que es el quirófano 1, como es el caso del paciente 16. El resto, podrán ser intervenido en cualquiera de ellos.

7.4.3 Datos referentes a cirujanos

A diferencia del banco de pruebas generado en la sección 7.2 donde la movilidad de los cirujanos (u_s) era 0 o |J| para todos los cirujanos en función de la instancia generada, en el ejemplo de aplicación abordado la movilidad de cada cirujano se determina de forma aleatoria el intervalo (1,|J|).

Tabla 7-21: Movilidad cirujanos del ejemplo de aplicación

Cirujano (*s*)	u_s
1	1
2	1
3	2
4	3

7.4.4 Evaluación de las alternativas

En esta sección se analizan y las soluciones ofrecidas por los modelos propuestos para el ejemplo de aplicación de la Unidad de Cirugía Plástica y Grandes Quemados del Hospital Universitario Virgen del Rocío. Las variables objetos de análisis son:

1. X_{ijh}: Variable que tomará el valor 1 si el paciente *i* es intervenido en el quirófano *j* el día *h* y 0 en caso contrario.
2. T_i: Variable que mide el valor del *tardiness* para la intervención de cada paciente *i*.
3. Z_{sjh}: Variable que tomará el valor 1 si el cirujano *s* está disponible para realizar intervenciones quirúrgicas en el quirófano *j* el día *h*.
4. A_{ih}: Variable que tomará el valor 1 si el paciente *i* ocupa cama PACU tras la intervención el día *h*.
5. B_{iw}: Variable que tomará el valor 1 si el paciente *i* ocupa cama ICU tras la intervención la semana *w*.
6. C_{iw}: Variable que tomará el valor 1 si el paciente *i* ocupa cama hospitalaria tras la intervención la semana *w*.

7.4.4.1 Modelo de programación lineal entera (1): Por etapas

Para poder realizarse la intervención quirúrgica del paciente *i* en el quirófano *j* el día *h* es necesario que el cirujano asignado a dicho paciente esté disponible en el quirófano y día asignado. La disponibilidad de los cirujanos se muestra en la tabla 7-22.

Tabla 7-22: Valor de Z_{sjh} para el modelo ILP (1)

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5		
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3
CIRUJANO 1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
CIRUJANO 2	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
CIRUJANO 3	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
CIRUJANO 4	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0

Tabla 7-23: Valor de X_{ijh} y T_i para el modelo ILP (1)

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5			**Tardiness**
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	
CIRUGÍA 1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 13	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 15	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

En el modelo propuesto la capacidad de los recursos perioperatorios es ociosa dado el tamaño reducido de la lista de espera (I=20) Por ello, todas las cirugías se programan en la **etapa 1** lo antes posible tras su fecha más temprana (ver en tabla 7-20, datos referentes a pacientes), logrando la maximización del nivel de servicio, la minimización del *tardiness*, que es cero en todos los casos, y la optimalidad de la solución. Esta programación se realiza sin consideración de la disponibilidad de las camas postoperatorias. Sin embargo, al existir camas postoperatorias suficientes, la planificación de las camas postoperatorias realizadas en la **etapa 2** será óptima tal y como se muestra en la tabla 7-24, realizada en función de la necesidad de los pacientes tras la intervención (ver β_i en la tabla 7-20).

Tabla 7-24: Valor de las variables A_{ih}, B_{iw} y C_{iw} para el modelo ILP (1)

	Cama PACU (β_i=1 o β_i=2)					Cama ICU (β_i=3)	Cama hospitalaria (β_i=2 o β_i=3)
	h=1	h=2	h=3	h=4	h=5	w=0	w=0
CIRUGÍA 1	0	1	0	0	0	0	1
CIRUGÍA 2	0	1	0	0	0	0	0
CIRUGÍA 3	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	1	0	0	0	0
CIRUGÍA 5	1	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	1	0	0	0	0
CIRUGÍA 8	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	1	0	0	0	0
CIRUGÍA 10	0	1	0	0	0	0	0
CIRUGÍA 11	0	0	1	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	0
CIRUGÍA 13	1	0	0	0	0	0	0
CIRUGÍA 14	1	0	0	0	0	0	0
CIRUGÍA 15	0	1	0	0	0	0	0
CIRUGÍA 16	0	0	1	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0
CIRUGÍA 18	1	0	0	0	0	0	1
CIRUGÍA 19	1	0	0	0	0	0	1
CIRUGÍA 20	0	0	0	0	0	0	0

En la **etapa 2** del modelo ILP (1) se asigna a cada paciente la/las cama/s postoperatoria/s necesaria/s en función de la necesidad tras haberse planificado la intervención en la **etapa anterior**. La función objetivo del modelo ILP (1) es finalmente la suma del valor de la solución obtenida en **la etapa 1** y en la **etapa 2**, resultando un total de 9,3657.

7.4.4.2 Modelo de programación entera (2): Integrado

A diferencia del modelo anterior, en este modelo se planifican las intervenciones quirúrgicas considerando la disponibilidad de las camas postoperatorias necesarias. En las tablas 7-25, 7-26 y 7-27 se recogen para el modelo la disponibilidad de los cirujanos, la planificación de las intervenciones quirúrgicas y la planificación de las camas postoperatorias respectivamente.

Tabla 7-25: Valor de Z_{sjh} para el modelo ILP (2)

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5		
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3
CIRUJANO 1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
CIRUJANO 2	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
CIRUJANO 3	1	0	0	1	1	0	1	0	1	0	0	0	0	0	0
CIRUJANO 4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Tabla 7-26: Valor de X_{ijh} y T_i para el modelo ILP (2)

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5			**Tardiness**
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	
CIRUGÍA 1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
CIRUGÍA 5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 13	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 15	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 7-27: Valor de las variables A_{ih}, B_{iw} y C_{iw} para el modelo ILP (2)

	Cama PACU (β_i=1 o β_i=2)					Cama ICU (β_i=3)	Cama hospitalaria (β_i=2 o β_i=3)
	h=1	h=2	h=3	h=4	h=5	w=0	w=0
CIRUGÍA 1	0	1	0	0	0	0	1
CIRUGÍA 2	0	1	0	0	0	0	0
CIRUGÍA 3	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	1	0	0	0	0
CIRUGÍA 5	1	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	1	0	0	0	0
CIRUGÍA 8	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	1	0	0	0	0
CIRUGÍA 10	0	1	0	0	0	0	0
CIRUGÍA 11	0	0	1	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	0
CIRUGÍA 13	1	0	0	0	0	0	0
CIRUGÍA 14	1	0	0	0	0	0	0
CIRUGÍA 15	0	1	0	0	0	0	0
CIRUGÍA 16	0	0	1	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0
CIRUGÍA 18	1	0	0	0	0	0	1
CIRUGÍA 19	1	0	0	0	0	0	1
CIRUGÍA 20	0	0	0	0	0	0	0

La solución obtenida para el problema de decisión por el modelo ILP (2) es óptima, con un valor de 9,3657.

7.4.4.3 Comparación entre ambos modelos

Ambos modelos ofrecen una solución óptima para el problema de decisión abordado con un valor de 9,3657. Esto se debe a que al existir camas postoperatorias suficientes, estas no representan ninguna limitación en la planificación de intervenciones quirúrgicas. Por lo tanto podríamos concluir que para problemas de decisión que no presenten limitación de los recursos postoperatorios, ambos modelos propuestos, independientemente de si considera o no la disponibilidad de las camas postoperatorias en la etapa de planificación de intervenciones quirúrgicas (**Etapa 1**), ofrecen una solución del mismo valor.

7.4.5 Saturación de camas postoperatorias

La crisis provocada por la COVID-19 ha llevado al sistema sanitario a un entorno caracterizado por la incertidumbre y variabilidad que ha ocasionado la saturación de las camas postoperatorias tales como camas UCI y camas hospitalarias. Por ello, en esta sección se pretende analizar qué ocurre en el ejemplo de aplicación propuesto si el número de las camas hospitalarias (dado que ningún paciente requiere camas UCI) se reduce significativamente a una sola unidad, debido a que solo tres pacientes requieren cama hospitalaria (ver β_i =2 o β_i =3 en la tabla 7-20) y a la duración de la estancia de cada paciente en dicha cama.

7.4.5.1 Modelo de programación lineal entera (1)

Tabla 7-28: Valor de Z_{sjh} para el modelo ILP (1) con limitación de camas hospitalarias

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5		
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3
CIRUJANO 1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
CIRUJANO 2	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
CIRUJANO 3	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
CIRUJANO 4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Tabla 7-29: Valor de X_{ijh} y T_i para el modelo ILP (1) con limitación de camas hospitalarias

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5			**Tardiness**
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	
CIRUGÍA 1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 13	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 15	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

El modelo ILP (1) realiza en la **etapa 1** una planificación de las intervenciones quirúrgicas sin la consideración de la disponibilidad de las camas postoperatorias. Sin embargo, la resolución de la **etapa 2** es infactible debido a la no existencia de camas hospitalarias suficientes. Este es sin duda el mayor inconveniente del modelo ILP (1), ya que al planificar intervenciones sin tener en cuenta la disponibilidad de las camas postoperatorias, planifica intervenciones que sí pueden ser realizadas debido a la disponibilidad de los recursos operatorios, pero que sin embargo, posteriormente no pueden ser enviadas al postoperatorio debido a la indisponibilidad de los recursos postoperatorios, en este caso camas hospitalarias. Esto ocasiona un bloqueo en las camas PACU o camas ICU, puesto que los pacientes que tras la estancia en dicha cama deben pasar a la cama de hospitalización, deberán permanecer en ella hasta que una cama hospitalaria queda liberada, resultando en la utilización innecesaria de un recurso. Esta situación puede desembocar a su vez en un bloqueo de quirófano, ya que la ocupación innecesaria de camas ICU (recurso también saturado) debido a la no disponibilidad de cama hospitalaria, impide que lo pacientes que sí la requieren tras la intervención la ocupen, debiendo permanecer en quirófano hasta que haya disponibilidad.

7.4.5.2 Modelo de programación lineal entera (2): Integrado

Tabla 7-30: Valor de Z_{sjh} para el modelo ILP (2) con limitación de camas hospitalarias

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5		
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3
CIRUJANO 1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUJANO 2	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
CIRUJANO 3	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0
CIRUJANO 4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Tabla 7-31: Valor de X_{ijh} y T_i para el modelo ILP (2) con limitación de camas hospitalarias

	DÍA 1			DÍA 2			DÍA 3			DÍA 4			DÍA 5			**Tardiness**
	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	OP 1	OP 2	OP 3	
CIRUGÍA 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
CIRUGÍA 5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 15	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 18	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIRUGÍA 20	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla 7-32: Valor de A_{ih}, B_{iw} y C_{iw} para el modelo ILP (2) con limitación de camas hospitalarias

	Cama PACU (β_i=1 o β_i=2)					Cama ICU (β_i=3)	Cama hospitalaria (β_i=2 o β_i=3)
	h=1	*h*=2	*h*=3	*h*=4	*h*=5	*w*=0	*w*=0
CIRUGÍA 1	0	0	0	0	0	0	0
CIRUGÍA 2	0	1	0	0	0	0	0
CIRUGÍA 3	0	0	0	0	0	0	0
CIRUGÍA 4	0	0	1	0	0	0	0
CIRUGÍA 5	1	0	0	0	0	0	0
CIRUGÍA 6	0	0	0	0	0	0	0
CIRUGÍA 7	0	0	1	0	0	0	0
CIRUGÍA 8	0	0	0	0	0	0	0
CIRUGÍA 9	0	0	1	0	0	0	0
CIRUGÍA 10	0	1	0	0	0	0	0
CIRUGÍA 11	0	0	1	0	0	0	0
CIRUGÍA 12	0	0	0	0	0	0	0
CIRUGÍA 13	1	0	0	0	0	0	0
CIRUGÍA 14	1	0	0	0	0	0	0
CIRUGÍA 15	0	1	0	0	0	0	0
CIRUGÍA 16	0	0	1	0	0	0	0
CIRUGÍA 17	1	0	0	0	0	0	0
CIRUGÍA 18	1	0	0	0	0	0	1
CIRUGÍA 19	1	0	0	0	0	0	1
CIRUGÍA 20	0	0	0	0	0	0	0

A diferencia del modelo anterior, este modelo sí considera la disponibilidad de las camas postoperatorias. En este caso, al no existir camas hospitalarias suficientes en la unidad teniendo en cuenta el número de pacientes que la requieren (ver β_i en la tabla 7-20), el modelo planificará entre las intervenciones que requieren camas hospitalarias, aquellas que maximizan la función objetivo y respetan las restricciones de capacidad. Por ello, el modelo planifica las cirugías 18 y 19, cirugías que requieren camas hospitalarias y pueden ser albergadas en la misma semana en una única cama de forma consecutiva, ya que la suma de sus tiempos no supera la capacidad semanal de la cama hospitalaria.

Sin embargo, si analizamos en profundidad, podemos ver que aunque los pacientes de las dos intervenciones puedan ocupar la única cama hospitalaria existente en la misma semana, las intervenciones se realizan el mismo día. Teniendo en cuenta que la cama hospitalaria es ocupada por los pacientes de forma sucesiva y la fecha de intervención prevista en la planificación, uno de los pacientes no podrá ocupar la cama hospitalaria de

forma directa tras la recuperación postanestésica ($ti_{,pacu}$) hasta que el otro paciente la libere. Por ello, se debe realizar un ajuste de la planificación realizada en un horizonte temporal menor, lo que queda fuera del alcance de este proyecto.

7.4.5.3 Comparación de los modelos propuestos

Teniendo en cuenta la actuación de cada uno de los modelos propuestos ante el problema de decisión con limitación de recursos postoperatorios estudiada en las secciones 7.4.5.1 y 7.4.5.2, podemos afirmar que el modelo que proporciona solución (factible u óptima) para el problema de planificación de intervenciones quirúrgicas y camas postoperatorias con limitación de recursos postoperatorios es el modelo ILP (2).

8 CONCLUSIONES

Tras revisar las contribuciones literarias existentes hasta el momento referente a la planificación de intervenciones quirúrgicas, se ha identificado que en ninguna de ellas se aborda la planificación de intervenciones quirúrgicas integrando todos los recursos postoperatorios existentes (camas PACU, camas ICU y camas hospitalarias) de manera diferenciada en función de sus características (servicio que ofrece al paciente, capacidad, coste, etc.). Por ello, se ha identificado la necesidad de abordar el problema de planificación de intervenciones quirúrgicas y camas postoperatorias de manera conjunta en este Trabajo de Fin de Grado con el objetivo de reducir el tiempo de acceso de los pacientes al sistema sanitario.

Si bien el estudio de la planificación de intervenciones quirúrgicas considerando la disponibilidad de las camas postoperatorias era necesario para evitar la interrupción del flujo del paciente a lo largo de las distintas etapas del proceso perioperatorio y el bloqueo de quirófanos, esta necesidad se ha hecho aún mayor con la crisis sanitaria provocada por la COVID-19 donde la planificación de las intervenciones quirúrgicas se ha visto afectada por la saturación de camas postoperatorias, recurso necesario tras la intervención quirúrgica que hasta el momento no había sido considerado en su totalidad en la planificación de intervenciones quirúrgicas pues no suponía una limitación en el proceso perioperatorio.

En este Trabajo de Fin de Grado se ha enmarcado en la planificación de intervenciones quirúrgicas y camas postoperatorias dentro de la gestión de la producción mediante el desarrollo de dos modelos de programación lineal entera (ILP). En ambos modelos, el proceso consta de dos etapas, una primera etapa donde se realiza la intervención quirúrgica y una segunda etapa donde tiene lugar la recuperación del paciente en las camas postoperatorias. Sin embargo, cada uno de los modelos planteados ha abordado la resolución del problema de decisión de manera diferente.

El primer modelo de programación lineal entera aborda el problema de planificación en dos etapas: Una primera etapa donde se realiza la planificación de las intervenciones quirúrgicas, y una segunda etapa donde tras la planificación de las intervenciones quirúrgicas, se planifican las camas postoperatorias. El principal inconveniente encontrado en este modelo es que al abordarse el problema de decisión en dos etapas consecutivas, la resolución de la segunda etapa se verá condicionada por los resultados obtenidos previamente en la primera etapa. El resultado de este modelo maximiza la utilización de los recursos operatorios, sin embargo, esto no ocurre con los recursos postoperatorios, puesto que ante la planificación de dos cirugías con el mismo peso clínico, planificará indiferenciadamente una cirugía u otra y no priorizará al paciente que maximice el objetivo de la segunda etapa, es decir, que ocupe cama postoperatoria.

A diferencia del modelo anterior, el segundo modelo de programación lineal entera propuesto aborda el problema de planificación de manera conjunta, es decir, realiza la planificación de las intervenciones quirúrgicas y camas postoperatorias simultáneamente de manera que maximizará la ocupación de los recursos operatorios y postoperatorios.

En ambos modelos, la planificación de las intervenciones quirúrgicas se aborda de la misma manera que se ha realizado en el artículo "*New heuristics for planning operating rooms*" (José M. Molina-Pariente, 2015).

Para evaluar la eficiencia de los modelos propuestos, se ha generado un banco de pruebas adaptando el generado en (José M. Molina-Pariente, 2015) al problema de decisión abordado, es decir, integrando las camas postoperatorias. El banco de pruebas se generó a partir de la combinación de los factores del problema β, α, $|J|$, $|L|$, $|M|$, $|N|$ y u_s, generándose dos iteraciones para cada combinación posible resultando en 256 instancias.

Tras analizar la experimentación de los dos modelos podemos concluir que el segundo modelo de programación lineal entera propuesto es el modelo más efectivo en la resolución del problema de decisión en todas las instancias tras el análisis de la desviación porcentual relativa (*RPD*) y el que logra al menos alguna solución óptima. No obstante, a pesar de que el primer modelo de programación lineal entera propuesto no halle ninguna solución óptima, todas las soluciones obtenidas, que son factibles, son muy próximas al óptimo debido al bajo valor del GAP, indicador de la distancia entre la solución factible obtenida y la óptima, que en ninguno de los casos supera el 0,02%. Por otro lado, ambos modelo propuestos proporcionan solución, ya sea factible u óptima para todas las instancias del problema abordado. A pesar de que la mayoría de las soluciones

proporcionadas por ambos modelos sean factibles y no óptimas, esto se debe al alcance del tiempo límite de ejecución de los dos modelos, lográndose de igual manera soluciones factibles muy próximas al óptimo (según indicador *GAP* en ambos modelos).

En ambos modelos encontramos determinados parámetros que influyen en mayor o menor medida en la obtención de los resultados, que son: |J|, u_s, α y β. Así, |J| y β influyen en el tiempo límite de ejecución de los modelos. La movilidad del cirujano determinada por el parámetro u_s interfiere en el número de soluciones óptimas encontradas, siendo mayor conforme aumenta el valor de este parámetro, al igual que ocurre con el factor de control α y el número quirófanos de la instancia (|J|).

Por último, se ha querido probar la efectividad de los modelos propuestos en un entorno real, en este caso la Unidad de Cirugía Plástica y Grandes Quemados del Hospital Universitario Virgen del Rocío. Tras la experimentación se ha llegado a la siguiente conclusión: Si no existe limitación de recursos postoperatorios, ambos modelos propuestos proporcionan la misma solución puesto que todos los pacientes intervenidos pueden pasar al postoperatorio independientemente de si se ha considerado o no la disponibilidad de los recursos postoperatorios en la planificación de intervenciones, ya que siempre habrá recursos postoperatorios disponibles. Si por lo contrario, nos encontramos ante un entorno variable, como ocurre con la actual crisis sanitaria provocada por la COVID-19, donde la disponibilidad de los recursos postoperatorios es limitada, el único modelo capaz de encontrar una solución que aborde las dos etapas del problema (intervenciones quirúrgicas y camas postoperatorias) es el modelo de ILP (2).

En conclusión, el modelo de planificación lineal entera que aborda de manera más efectiva la planificación de intervenciones quirúrgicas y camas postoperatorias es el modelo ILP(2).

8.1 Líneas de futuro

Con el objetivo de buscar un mayor número de soluciones óptimas para ambos modelos, se propone la realización de la experimentación en un horizonte temporal mayor donde la búsqueda de la solución en cada instancia no se vea restringida por el tiempo límite de ejecución establecido. Además, con el objetivo de afirmar la eficacia y validez de los métodos de resolución propuestos bajo otras condiciones sanitarias, se propone la generación de otro banco de pruebas (basado en datos reales) y la realización de la experimentación en el mismo, analizando el alcance de los métodos de resolución propuestos.

Además, para que los modelos de programación lineal entera desarrollados, especialmente el modelo con mayor efectividad demostrada a lo largo de este Trabajo de Fin de Grado pueda servir de aplicación al entorno real, sería conveniente estudiar la etapa del proceso perioperatorio no abordada en este proyecto, que es la fase médica previa a la intervención, y analizar si tiene influencia alguna en la planificación de intervenciones quirúrgicas y camas postoperatorias.

Finalmente, dado que los algoritmos empleados en los métodos de resolución propuestos son algoritmos exactos, se propone el desarrollo de un método de resolución basado algoritmos aproximados para el problema de decisión abordado y la comparación de la solución ofrecida con la de los métodos propuestos en este proyecto.

REFERENCIAS

A. Abedini, W. L. (2017). An optimization model for operating room scheduling to reduce blocking across the perioperative process. *45th SME North American Manufacturing Research* . Los Ángeles, Estados Unidos.

A. Fügener, E. H. (2014). Mater surtgery scheduling with consideration of multiple downstream units. *European Journal of Operational Research* , 227-236.

A. Jebali, A. H. (2006). Operating rooms scheduling. *International Journal of Production Economics* , 52-62.

A. Macario, T. S. (Diciembre de 1995). Where Are the Costs in Perioperative Care?: Analysis of Hospital Costs and Charges for Inpatien Surgical Care. *Anesthesiology* , págs. 1138-1144.

A. Testi, E. T. (2007). A three-phase approach for operating theatre schedule. *Health Care Management* , 163-172.

B. Cardoen, E. D. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research , 201* (3), 921-932.

B. Cardoen, E. D. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research* , 921-932.

B. González-Nusto, R. G. (1999). Waiting lists in Spanish public hospitals: A system dynamics approach. *System Dynamics Review , 15* (3), 201-224.

B. Roland, C. D. (2010). Scheduling an operating theatre under human resource constraints. *Computers and Industrial Engineering , 58* (2), 212-220.

C. Price, B. G. (2011). Reducing Boarding in a Post-Anesthesia Care Unit. *Production and Operations Management , 20* (3), 431-441.

D. Min, Y. Y. (2010). Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operation Research , 206*, 642-652.

E. Hans, G. W. (2008). Robust surgery loading. *European Journal of Operational Research* , 1038-1050.

E. Hans, M. v. (2012). A framework for healthcare planning and control. En *Handbook of healthcare system scheduling* (Vol. 168, págs. 303-320). Springer.

F. Dexter, A. M. (1999). An Operating Room Scheduling Strategy to Maximize the Use of Operating Room Block Time. *Anesthesia and Analgesia* , 7-20.

F. Dexter, E. M. (2006). Impact of surgical sequencing on postanesthesia care unit staffing. *Health Care Management Science* , 87-98.

F. Guerreiro, R. G. (2011). Operational research in the management of the operating theatre: A survey. . *Health Care Management Science* , 89-114.

H. Fei, N. M. (2010). A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers and Industrial Engineering , 58* (2), 221-230.

H. Saadouli, B. J. (2015). A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery departament. *Computers and Industrial Engineering* , 72-79.

Honduras, E. (7 de diciembre de 2020). *Espacio Honduras*. Recuperado el 6 de junio de 2021, de https://www.espaciohonduras.net/microsoft-visual-studio-concepto-y-que-es-y-para-que-sirve-microsoft-visual-studio

I. Marques, M. E. (2012). An integer programming approach to elective surgery scheduling . *OR Spectrum , 34*, 407-427.

IBM. (s.f.). Recuperado el 10 de junio de 2021, de https://www.ibm.com/es-es/analytics/optimization-solver

J. Beliën, E. D. (2007). Building cyclic master surgery schedules with leveled resulting bed occupancy.

European Journal of Operational Research , 1185-1204.

J. Blake, M. C. (2002). A goal programming approach to strategic resource allocation in acute care hospitals. *European Journal of Operational Research* , 541-561.

J. Bowers, G. M. (2005). Ambulatory care and orthopaedic capacity planning. *Health Care Management Science* , 41-47.

J. Vissers, R. B. (2005). Introduction. En *Health Operations Management: Patient Flow Logistics in Health Care.* Routledge.

J.H. May, W. E. (2011). The surgical problem: Current research and future oppotunities. *Production and Operations Management* , 392-405.

Jose M. Framiñan, R. L. (2014). *Manufacturing Scheduling Systems.* Londres: Springer.

José M. Molina-Pariente, E. W.-C. (2015). New heuristics for planning operating rooms. *Computers and Industrial Engineering , 90*, 429-443.

José. M. Molina-Pariente, V. F.-V. (2015). Integrated operating room planning and scheduling problem with assistant surgeon dependent surgery durations. *Computers and Industrial Engineering , 82*, 8-20.

L. de Pablos Escobar, M. C. (2021). Impacto de la COVID-19 sobre las listas de espera quirúrgicas. *Revista Española de Salud Pública , 95*.

M.Ban, B. D. (2017). Surgery scheduling with recovery resources. *IISE Transactions* , 942-955.

McClain, L. J. (1993). Chapter 7 An Overview of Production Planning. En *Logistics of Production and Inventory* (págs. 333-370). Nueva York: Elsevier Science Publisher.

Microsoft. (19 de marzo de 2019). Recuperado el 9 de junio de 2021, de https://docs.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2019

Miranda, P. (31 de Enero de 2018). *iberochile.com*. Recuperado el 24 de 05 de 2021, de http://www.iberochile.com/los-distintos-niveles-gestion-las-empresas/

P. Santibáñez, M. B. (2007). Surgical block scheduling in a system of hospitals: an aplication to resource and wait list management in a British Columbia health autority. *Health Care Management Science* , 269-282.

Ramírez Rojas, M. d. (2020). *Trabajo de Fin de Grado: Integración de planificación de consultas y quirófanos en el sector sanitario.* Sevilla: Escuela Técnica Superior de Ingeniería (Universidad de Sevilla).

S. Gür, T. E. (2018). Application of Operational Research Techniques in Operating. *Journal of Healthcare Engineering* , 1-16.

S.A. Kumar, N. S. (2006). Introduction to production and operation management. En *Production and Operations Managements* (pág. 1). New Age International Publishers.

Tutoriales, G. (2015 de 01 de 12). *Gestión de Operaciones*. Recuperado el 24 de 05 de 2021, de https://www.gestiondeoperaciones.net/procesos/que-es-la-gestion-de-operaciones/

V. Augusto, X. X. (2010). Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers and Industrial Engineering* , 231-238.

Y. Wang, J. T. (2015). Particle swar optimization-based planning and scheduling for a laminar-flow operating room with downstream resources. *Soft Computing* , 2913-2926.

ANEXO

```
using System;
using Gurobi;
using System.IO;

namespace Testbed_V._4
{
  class Program
  {
    static void Main(string[] args)
    {
      //Creación fichero
      System.IO.StreamWriter ResultadosUnificación, ResultadosIntegración;
      creacionFicheroCSV_nuevo(out ResultadosUnificación, out ResultadosIntegración);
      ResultadosUnificación.Write("Beta" + ";" + "Alfa" + ";" + "L" + ";" + "M" + ";" + "N" + ";" + "J" + ";" + "u_s" + ";" + "I" + ";" + "Iteracion" + ";" + " " + ";" + "Etapa 1" + ";" + "% GAP" + ";" + "Bound" + ";" + "Optimo" + ";" + "Solucion Optima" + ";" + "Solucion Factible" + "; " + "Solucion Infactible" + "; " + "Tiempo Ejecucion" + ";" +" " + ";" + "Etapa 2" + ";" + "% GAP" + ";" + "Bound" + ";" + "Optimo" + ";" + "Solucion Optima" + ";" + "Solucion Factible" + "; " + "Solucion Infactible" + "; " + "Tiempo Ejecucion" + ";" + "" + ";" + "Solución Unificacion" + ";" + "% GAP total"+ ";" + "RPD Unificacion" + "\n");
      ResultadosIntegración.Write("Beta" + ";" + "Alfa" + ";" + "L" + ";" + "M" + ";" + "N" + ";" + "J" + ";" + "u_s" + ";" + "I" + ";" + "Iteracion" + ";" + " " + ";" + "% GAP" + ";" + "Bound" + ";" + "Optimo" + ";" + "Solucion Optima" + ";" + "Solucion Factible" + "; " + "Solucion Infactible" + "; " + "Tiempo Ejecucion" + ";"+"RPD Integracion"+ "\n");
      ResultadosUnificación.Close();
      ResultadosIntegración.Close();

      //DECLARACIÓN FACTORES Y LE DAMOS LOS VALORES POSIBLES (EN FORMA DE VECTOR PARA QUE LUEGO GENERE LAS INSTANCIAS POSIBLES CON LOS DISTINTOS BUCLES)

      double[] Beta = new double[] { 1, 1.25 }; //El 100% y el 125% de la capacidad para generar el número de pacientes
      double[] Alfa = new double[] { 1.5, 2 }; // Factor de control para generar el número de cirujanos
      int[] J = new int[] {3,4}; //Número de ORS.
      int H = 5; //Horizonte temporal (días)
      int W = 1; //Horizonte temporal (semanas)
      int mds = 4; //Máximo número de días a la semana en los que trabajan los cirujanos
```

```
int[] t_i = new int[] { 90, 120 };
int t_i_medio = (int)((t_i[0] + t_i[1]) / t_i.Length);
int[] L = new int[] {2, 4}; //Número de cama PACU
int[] M = new int[] {6, 8}; //Número de cama ICU
int[] N = new int[] {15, 20}; //Número de cama hospitalaria
int numIteraciones = 2; //256 instancias

//Variables para recoger el análisis de resultados UNIFICACIÓN
//ETAPA1
int S_Optima1_Etapa1=0, S_Factible1_Etapa1=0, S_Infactible1_Etapa1 = 0;
double Optimo1_Etapa1=0, Bound1_Etapa1=0, Gap1_Etapa1=0, tiempo_ejecución1_Etapa1 = 0;

//ETAPA2
int S_Optima1_Etapa2 = 0, S_Factible1_Etapa2 = 0, S_Infactible1_Etapa2 = 0;
double Optimo1_Etapa2 = 0, Bound1_Etapa2 = 0, Gap1_Etapa2 = 0, tiempo_ejecución1_Etapa2 = 0;

//ETAPA1 + ETAPA 2 (Unificación)
double Gap1 = 0, Optimo1 = 0;



//Variables para recoger el análisis de resultados UNIFICACIÓN
int S_Optima2=0, S_Factible2=0, S_Infactible2 = 0;
double Optimo2=0, Bound2=0, Gap2=0, tiempo_ejecución2 = 0;

//Variable RPD para comparar los dos modelos
double RPD1=0, RPD2=0;

//Bucles para obtener cada uno de los escenarios de las instancias
for (int beta = 0; beta < Beta.Length; beta++)
{
   for (int alfa = 0; alfa < Alfa.Length; alfa++)
   {
      for (int l = 0; l < L.Length; l++)
      {
         for (int m = 0; m < M.Length; m++)
         {
            for (int n = 0; n < N.Length; n++)
            {
```

```
for (int j = 0; j < J.Length; j++)
{
   Console.WriteLine($"ALFA = {Alfa[alfa]} y J= {J[j]}");

   //Para determinar el número de cirujanos debemos seguir la formula S= capacidad de todos los quirófanos en HP/(nº de semanas * maximo tiempo disponible de un cirujano al día * mds)
   int S = (int)Math.Ceiling((Alfa[alfa] * 480 * H * J[j]) / (W * mds * 480)); //Redondeo al mas alto para que no sobre capacidad ORs

   //No se puede definir u_s hasta que no esté S definida porque es su dimensión.
   int[] datos_u_s = new int[S];
   int[] valor_u_s = new int[] { 1, J[j] };

   //Recorremos los posibles valores que puedde tomar u_s en función de los parámetros de los que depende, que es S.
   for (int v = 0; v < valor_u_s.Length; v++)
   {
      //Asignacion a los cirujanos nº quirofanos distintos/distintos.
      for (int s = 0; s < S; s++)
      {
         datos_u_s[s] = valor_u_s[v];
      }

      //Calcular el tamaño de la lista de cirugías. (I) (Generados 1 a 1)

      double CapacidadORs = J[j] * 480 * H * Beta[beta]; //Se multiplica por beta para generar el exceso de capacidad

      int I = 0;
      while (CapacidadORs > 0)
      {
         I++;
         CapacidadORs -= t_i_medio;
      }

      //Cálculo del tiempo límite
      double t_limite = I*H*J[j]*0.05;
```

```
// Factores principales declarados

/*//////////////////////
 //////////////////////
 C O M I E N Z O  D E  I T E R A C I O N E S
//////////////////////
//////////////////////*/

//Limito las iteraciones a una iteración para verificar el código C#
for (int iteracion = 0; iteracion < numIteraciones; iteracion++)
{
   //Abro fichero
   creacionFicherosCSV_ARPD_sinN(out ResultadosUnificación, out
ResultadosIntegración);

   //Genero los parámetros y EJECUTO LOS DOS MODELOS AQUÍ DENTRO
(PUESTO QUE AQUÍ ESTAN TODOS SUS PARÁMETROS GENERADOS)

   //Declaración parámetros de salida generador_instancias
   int[,] datos_r_jh;
   int[,] datos_a_sh;
   int[] y_i;
   int[] datos_rd_i;
   int[] datos_d_i;
   int[] datos_t_i;
   double[] datos_w_i;
   int[,,] delta_ijh;
   int[] t_i_pacu;
   int[] t_i_icu;
   int[] t_i_hospitalaria;
   int[,] a_lh;
   int[,] b_mw;
   int[,] c_nw;
   int[] beta_i;
   int a;
   int b;
   int[] VS;
   double Wmax;
```

```
//LLAMADA A LA FUNCIÓN GENERACIÓN INSTANCIA
Generador_instancias(H, W, S, I, J, j, L, l, M, m, N, n, mds, datos_u_s, out datos_r_jh, out datos_a_sh, out y_i, out datos_rd_i, out datos_d_i, out datos_t_i, out datos_w_i, out delta_ijh, out t_i_pacu, out t_i_icu, out t_i_hospitalaria, out a_lh, out b_mw, out c_nw, out beta_i, out a, out b, out VS, out Wmax);

/*//////////////////////////////
   LLAMADA A LOS MODELOS
//////////////////////////////*/
Unificación(H, I, J[j], S, datos_r_jh, datos_a_sh, datos_u_s, y_i, datos_rd_i, datos_d_i, datos_t_i, datos_w_i, delta_ijh, W, L[l], M[m], N[n], t_i_pacu, t_i_icu, t_i_hospitalaria, beta_i, a_lh, b_mw, c_nw, Wmax, a, b, VS, t_limite, ref S_Optima1_Etapa1, ref S_Factible1_Etapa1, ref S_Infactible1_Etapa1, ref Optimo1_Etapa1, ref Gap1_Etapa1, ref Bound1_Etapa1, ref tiempo_ejecución1_Etapa1, ref S_Optima1_Etapa2, ref S_Factible1_Etapa2, ref S_Infactible1_Etapa2, ref Optimo1_Etapa2, ref Gap1_Etapa2, ref Bound1_Etapa2, ref tiempo_ejecución1_Etapa2);

//Cálculo del GAP conjunto de las dos etapas
Gap1 = (((Bound1_Etapa1 + Bound1_Etapa2) - (Optimo1_Etapa1 + Optimo1_Etapa2)) / (Optimo1_Etapa1 + Optimo1_Etapa2))*100;
Optimo1 = Optimo1_Etapa1 + Optimo1_Etapa2;

Integración(H, I, J[j], S, datos_r_jh, datos_a_sh, datos_u_s, y_i, datos_rd_i, datos_d_i, datos_t_i, datos_w_i, delta_ijh, W, L[l], M[m], N[n], t_i_pacu, t_i_icu, t_i_hospitalaria, beta_i, a_lh, b_mw, c_nw, Wmax, a, b, VS, t_limite, ref S_Optima2, ref S_Factible2, ref S_Infactible2, ref Optimo2, ref Gap2, ref Bound2, ref tiempo_ejecución2);

/*//////////////////
CALCULO RPD
//////////////////*/

//Ver cual de las soluciones es mejor (Best sol - Heu sol)/Best sol * 100 (Al ser de maximizar, la mayor solución es la mejor)
if (Optimo1 > Optimo2)
{
   RPD1 = ((Optimo1 - Optimo1) / Optimo1)*100;
   RPD2 = ((Optimo1 - Optimo2) / Optimo1) * 100;
}
else
{
   RPD1 = ((Optimo2 - Optimo1) / Optimo2) * 100;
   RPD2 = ((Optimo2 - Optimo2) / Optimo2) * 100;
}
```

```
                        //Escribo en fichero la iteración que estoy realizando y vuelco los datos de interés

                        ResultadosUnificación.Write(Beta[beta] + ";" + Alfa[alfa] + ";" + L[l] + ";" + M[m] + ";" + N[n] + ";" + J[j] + ";" +  datos_u_s[v] + ";" + I + ";" + iteracion + ";" + " " + ";" + "" + ";" + Gap1_Etapa1 + ";" + Bound1_Etapa1 + ";" + Optimo1_Etapa1 + ";" + S_Optima1_Etapa1 + ";" + S_Factible1_Etapa1 + "; " + S_Infactible1_Etapa1 + "; " + tiempo_ejecución1_Etapa1+ ";" + " " + ";" + "" + ";" + Gap1_Etapa2 + ";" + Bound1_Etapa2 + ";" + Optimo1_Etapa2 + ";" +  S_Optima1_Etapa2 + ";" + S_Infactible1_Etapa2 + "; " + S_Infactible1_Etapa2+ "; " + tiempo_ejecución1_Etapa2 + ";" +"" + ";" + Optimo1 + ";" + Gap1 + ";" + RPD1 + "\n");
                        ResultadosIntegración.Write(Beta[beta] + ";" + Alfa[alfa] + ";" + L[l] + ";" + M[m] + ";" + N[n] + ";" + J[j] + ";" + datos_u_s[v]  + ";" + I + ";" + iteracion + ";" + " " + ";" + Gap2 + ";" + Bound2 + ";" + Optimo2 + ";" + S_Optima2 + ";" + S_Factible2 + ";" + S_Infactible2 + ";" + tiempo_ejecución2 + ";" + RPD2 + "\n");


                        // Cierro ficheros
                        ResultadosUnificación.Close();
                        ResultadosIntegración.Close();
                    }
                }
            }
        }
    }
  }
} //LLAVE CIERRE MAIN


/*////////////////////////////////
FUNCIÓN CREACIÓN DE FICHERO NUEVO
////////////////////////////////*/
private static void creacionFicheroCSV_nuevo(out System.IO.StreamWriter ResultadosUnificación, out System.IO.StreamWriter ResultadosIntegración)
{
  //creación fichero 1
  String ruta = AppDomain.CurrentDomain.BaseDirectory;
  ruta = ruta.Replace("\\", "/").ToString();
  DirectoryInfo DIRESC = new DirectoryInfo(ruta + "/temp");
  if (!DIRESC.Exists)
  {
    DIRESC.Create();
```

```
        }
        String ficBatBorrar = ruta + "temp" + "/ResultadosUnificación.csv";
        FileInfo FicheroEliminar = new FileInfo(ficBatBorrar);
        if (FicheroEliminar.Exists)
        {
           File.Delete(ficBatBorrar);
        }
        String ficCSV = ruta + "temp" + "/ResultadosUnificación.csv";
        ResultadosUnificación = new System.IO.StreamWriter(ficCSV, true);
        ResultadosUnificación.Write("\n\n");

        //creación fichero 2
        ruta = AppDomain.CurrentDomain.BaseDirectory;
        ruta = ruta.Replace("\\", "/").ToString();
        DIRESC = new DirectoryInfo(ruta + "/temp");
        if (!DIRESC.Exists)
        {
           DIRESC.Create();
        }
        ficBatBorrar = ruta + "temp" + "/ResultadosIntegración.csv";
        FicheroEliminar = new FileInfo(ficBatBorrar);
        if (FicheroEliminar.Exists)
        {
           File.Delete(ficBatBorrar);
        }
        ficCSV = ruta + "temp" + "/ResultadosIntegración.csv";
        ResultadosIntegración = new System.IO.StreamWriter(ficCSV, true);
        ResultadosIntegración.Write("\n\n");
    }

    private static void creacionFicherosCSV_ARPD_sinN(out System.IO.StreamWriter
ResultadosUnificación, out System.IO.StreamWriter ResultadosIntegración)
    {
        String ruta = AppDomain.CurrentDomain.BaseDirectory;
        ruta = ruta.Replace("\\", "/").ToString();
        String ficCSV = ruta + "temp" + "/ResultadosUnificación.csv";
        ResultadosUnificación = new System.IO.StreamWriter(ficCSV, true);
```

```
    ruta = AppDomain.CurrentDomain.BaseDirectory;
    ruta = ruta.Replace("\\", "/").ToString();
    ficCSV = ruta + "temp" + "/ResultadosIntegración.csv";
    ResultadosIntegración = new System.IO.StreamWriter(ficCSV, true);
}

/*////////////////////////////////
FUNCIÓN CORRESPONDENCIA DÍA-SEMANA
 ////////////////////////////////*/
static int[] Correspondencia(int H, int W, int díasemana)
//Correspondencia entre días y semanas
{
    int[] VS = new int[H];
    int x = 0;
    for (int w = 0; w < W; w++)
    {
        int contador = 0;
        while (x < H && contador < díasemana)
        {
            VS[x] = w;
            contador++;
            //Console.WriteLine($"El día {x} pertenece a la semana {w}");
            x++;

        }
    }

    return VS;
}

/*////////////////////////////////
FUNCIÓN GENERADORA DE PARÁMETROS
////////////////////////////////*/
    static void Generador_instancias(int H, int W, int S, int I, int[] J, int j, int[] L, int l, int[] M, int m, int[] N, int n, int mds, int[] datos_u_s, out int[,] datos_r_jh, out int[,] datos_a_sh, out int[] y_i, out int[] datos_rd_i, out int[] datos_d_i, out int[] datos_t_i, out double[] datos_w_i, out int[,,] delta_ijh, out int[] t_i_pacu, out int[] t_i_icu, out int[] t_i_hospitalaria, out int[,] a_lh, out int[,] b_mw, out int[,] c_nw,out int[] beta_i, out int a, out int b, out int[] VS, out double Wmax)
    {
```

```
//Parámetros necesarios para el cálculo de parámetros del modelo
Random aleatorio = new Random(1);
int[] valores_MTBT = new int[] { 45, 180, 360 };
int díasemana = 5; //nº de días laborables que tiene 1 semana //La poongo aquí para que esté en la bateria genérica, pero me lanza problema de advertencia.
VS = new int[H];

//Llamada a la función CORRESPONDENCIA
VS = Correspondencia(H, W, díasemana);

//Parámetros de las F.O.
a = 1; //Parámetro de ponderación del tardiness
b = 1;//Parámetro de ponderación de la F.O etapa 2 (Asignación de camas)

  /*////////////////////////////////////
   PARÁMETROS RELEVANTES QUIRÓFANOS
  ////////////////////////////////////*/
  datos_r_jh = new int[J[j], H];
  for (int aux_j = 0; aux_j < J[j]; aux_j++)
  {
    for (int h = 0; h < H; h++)
    {
      datos_r_jh[aux_j, h] = 8 * 60; //Todos los OP tienen 8 horas/dia de capacidad
    }
  }

  /*////////////////////////////////////
   PARÁMETROS RELEVANTES CIRUJANOS
  ////////////////////////////////////*/

  //Construcción de la programación semanal especificando que cirujanos estarán disponibles cada día.
  //PASO 1: Asignar para cada día número de cirujanos=número de quirófanos disponibles de manera aleatoria.

  int[,] programacion_s = new int[H, J[j]];
  for (int h = 0; h < H; h++)
  {
    programacion_s[h, 0] = aleatorio.Next(0, S);
    //Console.WriteLine($" El cirujano asignado al OR 0 el día {h} es {programacion_s[h,0]}");
```

```
            for (int or = 1; or < J[j]; or++)
            {
              //Condición para que asigne cirujano a un quirófano
              int stop = 1;

              while (stop == 1)
              {
                programacion_s[h, or] = aleatorio.Next(0, S);

                int auxOR = 0;
                stop = 0; //Condición de parada para que pase al siguiente quirófano

                //Comprobación de que el cirujano no ha sido asignado previamente en cualquiera de los OR
                while (auxOR < or && stop == 0) //SOLO HACE FALTA MIRARLO EN LOS QUIRÓFANOS ANTERIORES AL QUE SE HA ASIGNADO PORQUE EL RESTO ESTÁN SIN ASIGNAR AÚN, POR LO CUAL LA CONDICIÓN ANTERIOR DONDE MIRABA EN TODOD & QUE FUERA DISTINTO AL Q ESTABA MIRANDO, ERA REDUNDANTE)
                {
                  if (programacion_s[h, or] == programacion_s[h, auxOR])

                  {
                    stop = 1; //En el caso de que haya asignado un cirujano ya asignado previamente a otro OR para el mismo día, ponemos 1 para que vuelva a realizar la asignación
                  }
                  auxOR++;
                }
              }
              //Console.WriteLine($" El cirujano asignado al OR {or} el día {h} es {programacion_s[h,or]}");
            }
          }

          //PASO 2:

          /*//////////
           Declaro las variables fuera del bucle para que sean de ámbito de aplicación para la segunda parte del paso 2
          //////////*/

          //Matriz Cirujanos/días
```

```
int[,] disponibilidad_s = new int[S, H];
//0= sin quirófano asignado
//1= disponible en quirófano asignado

//Variable para contar el número de días que el cirujano ha sido asignado en el HP
int[] contador = new int[S]; //1 contador para cada cirujano

//Parte 1 paso 2: Contar veces asignados semanalmente en paso 1 y asignar variable disponibilidad

for (int s = 0; s < S; s++)
{

   for (int h = 0; h < H; h++)
   {
      for (int or = 0; or < J[j]; or++)
      {
         if (s == programacion_s[h, or])
         {
            contador[s]++;
            disponibilidad_s[s, h] = 1;
         }
      }
         //Console.WriteLine($"La disponibilidad del cirujano {s} para realizar cirugías el día {h} es {disponibilidad_s[s, h]}");
   }

   //Console.WriteLine( $"El cirujano {s} ha sido asignado {contador[s]} veces en el PASO 1");

}

//Parte 2 paso 2: Asignar cirujanos disponibles teniendo en cuenta la diferencia entre mds y veces asignados en paso 1.

//Matriz cirujanos tras la reasignación del paso 2
int[,] disponibilidad_s_p2 = new int[S, H];
//CREO ESTA MATRIZ PORQUE NO QUIERO MODIFICAR LA MATRIZ DE DISPONIBILIDAD INICIAL, Y ASÍ PUEDO COMPARAR LO QUE SE HA HECHO EN P2 CON RESPECTO A P1.
```

```
for (int s = 0; s < S; s++)
{
   int comparador = 0;
   comparador = mds - contador[s];

   for (int h = 0; h < H; h++)
   {
      //Asignación a la matriz de reasignación los valores iniciales

      disponibilidad_s_p2[s, h] = disponibilidad_s[s, h];
      if (comparador > 0 && disponibilidad_s[s, h] == 0)
      {
         //Asignamos en la matriz de reasignación el cambio en la disponibilidad del cirujano debido el P.2
         disponibilidad_s_p2[s, h] = 1; //Asignamos al cirujano a dicho día.
         comparador--; //Garantiza que una vez asignada la diferencia deje de asignar (no entra más en el bucle)
      }
   }
}

//Calculo capacidad cirujano. Matriz a_sh . Si esta disponible, capacidad =8h * 60 min/h
 datos_a_sh = new int[S, H];

for (int s = 0; s < S; s++)
{
   for (int h = 0; h < H; h++)
   {
      datos_a_sh[s, h] = 0; //Asigno el valor 0 inicialmente a toda la matriz

      if (disponibilidad_s_p2[s, h] == 1)
      {
         datos_a_sh[s, h] = 480; //Si el cirujano está disponible, cambio el valor por la capacidad de OR.
      }
   }
}
```

```
/*//////////////////////////
 PARÁMETROS RELEVANTES PACIENTE CIRUGÍA I
//////////////////////////*/


//ETAPA 1
datos_t_i = new int[I];
datos_w_i = new double[I]; //Peso clínico
datos_rd_i = new int[I];
datos_d_i = new int[I];
delta_ijh = new int[I, J[j], H]; // 10% cirugías son especializadas, 30% de los quirófanos están especializados
y_i = new int[I];
int[] MTBT = new int[I];
int[] mp = new int[I]; //Prioridad médica
int[] dwl = new int[I]; //Días en lista de espera
double factor_a = 0.5; //Factor a para calcular el peso clínico



//ETAPA 2
beta_i = new int[I];
t_i_pacu = new int[I];
t_i_icu = new int[I];
t_i_hospitalaria = new int[I];


int p_aleatoria = 0;


for (int i = 0; i < I; i++)
{
  p_aleatoria = aleatorio.Next(0, 3); //Genera aleatoriamente una de las posiciones del vector valores_MTBT
  MTBT[i] = valores_MTBT[p_aleatoria];


  mp[i] = aleatorio.Next(1, 6); //valor aleatorio entre 1 y 5
  dwl[i] = aleatorio.Next(1, MTBT[i]); //En el intervalo [1, MTBT-1]
  datos_w_i[i] = (factor_a * ((double)mp[i] / (double)5) + (1 - factor_a) * ((double)dwl[i] / (double)MTBT[i]));
  datos_d_i[i] = MTBT[i]- dwl[i];
}
```

```
            //Cálculo W_max
            Wmax = new double();
            double w_i_max = new double();
            int mp_max = 5;
            int dwl_max = MTBT[2] - 1;
            int MTBT_max = valores_MTBT[2];
            w_i_max = (factor_a * ((double)mp_max / (double)5) + (1 - factor_a) * ((double)dwl_max / (double)MTBT_max));
            Wmax = w_i_max * (double)I;



            //QUIRÓFANO ESPECIALIZADO(30%)


            double numero_OR_especializado = 0.3 * J[j];
            int OR_especializados = (int)Math.Round(numero_OR_especializado, 0); //Redondeo (PARA AMBOS VALORES DE J SIEMPRE HAY 1 OP ESPECIALIZADO)
            //Console.WriteLine($"{numero_OR_especializado} son {OR_especializados} OR especializados ");



        for (int i = 0; i < I; i++)
            {
              //ETAPA 1
              datos_t_i[i] = aleatorio.Next(90, 121);
              datos_rd_i[i] = aleatorio.Next(1, 4);
              y_i[i] = aleatorio.Next(0, S);

              //DELTA_IJH

              //Cirugía especializada o no
              double[] percent = new double[I];
              bool[] especializada = new bool[I];
              percent[i] = aleatorio.NextDouble();
              if (percent[i] <= 0.1)
              {
                 especializada[i] = true;
              }
            //Console.WriteLine($" La cirugía del paciente{i} con un porcentaje de {percent[i] * 100.0}% es especializada: {especializada[i]}");
```

```
    for (int aux_j = 0; aux_j < J[j]; aux_j++)
    {
        for (int h = 0; h < H; h++)
        {
            //Inicio todos en 0, para luego asignar 1 solo a los que correspondan
            delta_ijh[i, aux_j, h] = 0;

            if (especializada[i] == false)
            {
                delta_ijh[i, aux_j, h] = 1; //En cualquier quirófano y día

            }
            else
            {
                if (aux_j < OR_especializados)
                {
                    delta_ijh[i, aux_j, h] = 1;
                }

            }
           // Console.WriteLine($" delta{i}.{aux_j}.{h}={delta_ijh[i, aux_j, h]}");
        }
    }


//ETAPA 2
//PORCENTAJE PARA EL CALCULO DE BETA I
double percent_ICU = new double();
double percent_PACU = new double();
//Entre el 5 y el 10% del total de pacientes van a ICU
percent_ICU = (aleatorio.Next(5, 10)) / (double)100;
//De los que no van a ICU, entre el 50 y el 100% van a PACU
percent_PACU = (aleatorio.Next(50, 100)) / (double)100;


//Para ver si el paciente está dentro o no de ICU
double percent_beta_i_ICU = aleatorio.NextDouble();
```

```
//para ver si en el caso de que no necesite ICU, el paciente necesita PACU o no
double percent_beta_i_PACU = aleatorio.NextDouble();

//para ver si necesitando PACU, el paciente es del tipo 1 o 2
double percent_tipo_PACU = aleatorio.NextDouble();

if (percent_beta_i_ICU <= percent_ICU)
{
   beta_i[i] = 3;
}
else
{
   if(percent_beta_i_PACU <= percent_PACU)
   {
      if (percent_tipo_PACU <= 0.1) //El 10% de PACU necesita hospitalaria
      {
         beta_i[i] = 2;
      }
      else
      {
         beta_i[i] = 1;
      }
   }
   else
   {
      beta_i[i] = 0;
   }
}

// Console.WriteLine($" el paciente { i} es { beta_i[i]}");

//paciente PACU o PACU+HOSPITALARIA
if (beta_i[i] == 1 || beta_i[i] == 2)
{
   t_i_pacu[i] = aleatorio.Next(1, 4);
}
//paciente ICU
```

```
    if (beta_i[i] == 3)
    {
        t_i_icu[i] = aleatorio.Next(12, 37);
    }

    //paciente HOSPITALARIA
    if (beta_i[i] == 2 || beta_i[i] == 3)
    {
        t_i_hospitalaria[i] = aleatorio.Next(1, 169);
    }
}

/*//////////////////////
 PARÁMETROS RELATIVOS A LAS CAMAS
//////////////////////*/

//CAPACIDAD CAMAS
a_lh = new int[L[l], H];
b_mw = new int[M[m], W];
c_nw = new int[N[n], W];

//Camas PACU (diaria)
for (int l_aux = 0; l_aux < L[l]; l_aux++)
{
    for (int h = 0; h < H; h++)
    {
        a_lh[l_aux, h] = 24; //24h al día
    }
}

//Capacidad cama ICU (Semanal)
for (int m_aux = 0; m_aux < M[m]; m_aux++)
{
    for (int w = 0; w < W; w++)
    {
        b_mw[m_aux, w] = 168; //Está disponible toda la semana
    }
}
```

```
        //Capacidad cama HOSPITALARIA (Semanal)
        for (int n_aux = 0; n_aux < N[n]; n_aux++)
        {
           for (int w = 0; w < W; w++)
           {
              c_nw[n_aux, w] = 168; //Está disponible toda la semana
           }
        }
    }

    /*////////////////////////////////
     MODELO UNIFICACIÓN
    ////////////////////////////////*/
    static void Unificación(int H, int I, int J, int S, int[,] datos_r_jh, int[,] datos_a_sh, int[] datos_u_s, int[] y_i, int[] datos_rd_i, int[] datos_d_i, int[] datos_t_i, double[] datos_w_i, int[,,] delta_ijh, int W, int L, int M, int N, int[] t_i_pacu, int[] t_i_icu, int[] t_i_hospitalaria, int[] beta_i, int[,] a_lh, int[,] b_mw, int[,] c_nw, double Wmax, int a, int b, int[] VS, double t_limite, ref int S_Optima1_Etapa1, ref int S_Factible1_Etapa1, ref int S_Infactible1_Etapa1, ref double Optimo1_Etapa1, ref double Gap1_Etapa1, ref double Bound1_Etapa1, ref double tiempo_ejecución1_Etapa1, ref int S_Optima1_Etapa2, ref int S_Factible1_Etapa2, ref int S_Infactible1_Etapa2, ref double Optimo1_Etapa2, ref double Gap1_Etapa2, ref double Bound1_Etapa2, ref double tiempo_ejecución1_Etapa2)

    {
       try
       {
          //Creación entorno vacio e iniciación
          GRBEnv env = new GRBEnv("mip1.log");

          //Creación modelo vacío
          GRBModel model = new GRBModel(env);
          model.Set(GRB.StringAttr.ModelName, "Unificación (E1, E2)");


          /*////////////////////
          PRIMERA ETAPA DEL MODELO
          ////////////////////*/

          //Declaración de VARIABLES
          GRBVar[,,] X_ijh = new GRBVar[I, J, H];
```

```
for (int i = 0; i < I; i++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      { X_ijh[i, j, h] = model.AddVar(0, 1, 0, GRB.BINARY, "X_" + i + "_" + j + "_" + h); }
   }
}
GRBVar[,,] Z_sjh = new GRBVar[S, J, H];
for (int s = 0; s < S; s++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      { Z_sjh[s, j, h] = model.AddVar(0, 1, 0, GRB.BINARY, "Z_" + s + "_" + j + "_" + h); }
   }
}

//Definición variable tardiness2
GRBVar[] T_i = new GRBVar[I];
for (int i = 0; i < I; i++)
{
   T_i[i] = model.AddVar(0, GRB.INFINITY, 0, GRB.INTEGER, "T_" + i);
}



//Establecer función objetivo

GRBLinExpr objetivo1 = 0; //Declarada e iniciada

for (int h = 0; h < H; h++)
{
   double numero = (double)1 / (double)(h + 1);
   for (int i = 0; i < I; i++)
   {
      for (int j = 0; j < J; j++)
      {
```

```
            objetivo1 += ((double)1 / (double)Wmax) * (numero * datos_w_i[i] * X_ijh[i, j, h]);

        }
    }
}

for (int i = 0; i < I; i++)
{
    objetivo1 -= ((double)a / (double)(H * Wmax)) * (T_i[i] * datos_w_i[i]);
}
model.SetObjective(objetivo1, GRB.MAXIMIZE);



/*//////////////////////////
 RESTRICCIONES
//////////////////////*/

//Restricción Tardiness pacientes operados en el HP

for (int i = 0; i < I; i++)
{
    GRBLinExpr ladoIzqT_1 = 0;
    ladoIzqT_1 += T_i[i];
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            ladoIzqT_1 -= X_ijh[i, j, h] * h; //Cálculo del día de operación
        }
    }
    ladoIzqT_1 += datos_d_i[i];
    model.AddConstr(ladoIzqT_1 >= 0, "Tardiness_1");
}
//Restricción Tardiness para pacientes no operados en el HP (La máxima tardanza posible)
int Ult_dia = H;
for (int i = 0; i < I; i++)
{
    GRBLinExpr ladoIzqT_2 = 0;
```

```
    ladoIzqT_2 += T_i[i];
    ladoIzqT_2 -= Ult_dia;
    ladoIzqT_2 += datos_d_i[i];
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            ladoIzqT_2 += X_ijh[i, j, h] * (Ult_dia - datos_d_i[i]); //Cálculo del día de operación
        }
    }
    model.AddConstr(ladoIzqT_2 >= 0, "Tardiness_2");
}

//Restricción 2: No se programe cirugía más de una vez en HP.
for (int i = 0; i < I; i++)
{
    GRBLinExpr ladoIzqR2 = 0;
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            ladoIzqR2 += X_ijh[i, j, h];
        }
    }
    model.AddConstr(ladoIzqR2 <= 1, "Restricción 2");
}

//Restricción 3.1: Que se programe la cirugia dsps del release time
for (int i = 0; i < I; i++)
{
    GRBLinExpr ladoIzqR31 = 0;
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < datos_rd_i[i] - 1 && h<H; h++)
        {
            ladoIzqR31 += X_ijh[i, j, h];
        }
```

```
    }
    model.AddConstr(ladoIzqR31 == 0, "Restricción 3.1");
}

//Sin restricción del LASTEST. Lo minimizamos en la F.O.

//Restricción 4: Capacidad OP diaria
for (int j = 0; j < J; j++)
{
    for (int h = 0; h < H; h++)
    {
        GRBLinExpr ladoIzqR4 = 0;
        for (int i = 0; i < I; i++)
        {
            ladoIzqR4 += datos_t_i[i] * X_ijh[i, j, h];
            model.AddConstr(ladoIzqR4 <= datos_r_jh[j, h], "Restricción 4");
        }
    }
}

//Restricción 5: Capacidad cirujano diaria
for (int s = 0; s < S; s++)
{
    for (int h = 0; h < H; h++)
    {
        GRBLinExpr ladoIzqR5 = 0;
        for (int j = 0; j < J; j++)
        {
            for (int i = 0; i < I; i++)
            {
                if (y_i[i] == s)
                { ladoIzqR5 += datos_t_i[i] * X_ijh[i, j, h]; }
            }
        }
        model.AddConstr(ladoIzqR5 <= datos_a_sh[s, h], "Restricción 5");
    }
}
```

```
//Restricción 6: Asignación de quirófanos
for (int s = 0; s < S; s++)
{
   for (int h = 0; h < H; h++)
   {
      GRBLinExpr ladoIzqR6 = 0;
      for (int j = 0; j < J; j++)
      {
         ladoIzqR6 += Z_sjh[s, j, h];
      }

      model.AddConstr(ladoIzqR6 <= datos_u_s[s], "Restricción 6");
   }
}

// Restricción 7.1 : Capacidad quirófano

for (int s = 0; s < S; s++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         GRBLinExpr ladoIzqR71 = 0;

         for (int i = 0; i < I; i++)
         {
            if (y_i[i] == s)
            {
               ladoIzqR71 += datos_t_i[i] * X_ijh[i, j, h];
            }
         }
         ladoIzqR71 -= datos_r_jh[j, h] * Z_sjh[s, j, h];
         model.AddConstr(ladoIzqR71 <= 0, "Restricción 7.1");
      }
   }
}
```

```
//Restricción 7.2:  No asignar cirujano si no hay cirugías
for (int s = 0; s < S; s++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         GRBLinExpr ladoIzqR72 = 0;
         for (int i = 0; i < I; i++)
         {
            if (y_i[i] == s)
            {
               ladoIzqR72 += X_ijh[i, j, h];
            }

         }
         ladoIzqR72 -= Z_sjh[s, j, h];
         model.AddConstr(ladoIzqR72 >= 0, "Restricción 7.2 ");
      }

   }

}

//Restricción 8: Garantiza que cada cirugía se lleve a cabo en un OR adecuado.
for (int i = 0; i < I; i++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         if (delta_ijh[i, j, h] == 0)
         {
            GRBLinExpr ladoIzqR8 = 0;
            ladoIzqR8 = X_ijh[i, j, h];
            model.AddConstr(ladoIzqR8 == 0, "Restricción 8");
         }
```

```
        }
    }
}

//Restricción 11: tardiness >=0

for (int i = 0; i < I; i++)
{
    GRBLinExpr LadoIzq11 = 0;
    LadoIzq11 = T_i[i];
    model.AddConstr(LadoIzq11 >= 0, "Restricción 11");
}


//Optimizar el modelo
model.GetEnv().Set(GRB.DoubleParam.TimeLimit, t_limite);
model.Optimize();

//Tiempo de ejecución en segundos
tiempo_ejecución1_Etapa1 = model.Get(GRB.DoubleAttr.Runtime);

//Recogida análisis soluciones
int optimstatus = model.Get(GRB.IntAttr.Status);

//Si el modelo es infactible o ilimitado se reoptimiza
if (optimstatus == GRB.Status.INF_OR_UNBD)
{
    model.GetEnv().Set(GRB.IntParam.Presolve, 0);
    model.Optimize();
    tiempo_ejecución1_Etapa1 = model.Get(GRB.DoubleAttr.Runtime);
    optimstatus = model.Get(GRB.IntAttr.Status);
}

//Modelo óptimo
if (optimstatus == GRB.Status.OPTIMAL)
{
```

```
            Optimo1_Etapa1 = model.Get(GRB.DoubleAttr.ObjVal);
            S_Optima1_Etapa1 = 1;
            S_Factible1_Etapa1 = 0;
            S_Infactible1_Etapa1 = 0;
            Bound1_Etapa1 = Optimo1_Etapa1;
            Gap1_Etapa1 = 0;

        }
        else if (optimstatus == GRB.Status.INFEASIBLE)
        {
            S_Factible1_Etapa1 = 0;
            S_Infactible1_Etapa1 = 1;
            Bound1_Etapa1 = 0;
            Optimo1_Etapa1 = 0;
            Gap1_Etapa1 = 0;
        }
        else if (optimstatus == 9)
        {
            S_Factible1_Etapa1 = 1;
            S_Infactible1_Etapa1 = 0;
            S_Optima1_Etapa1 = 0;
            Bound1_Etapa1 = model.Get(GRB.DoubleAttr.ObjBound); //Lower bound
            Optimo1_Etapa1 = model.Get(GRB.DoubleAttr.ObjVal); //Best bound
            Gap1_Etapa1 = (Math.Abs(Bound1_Etapa1 - Optimo1_Etapa1) /
Math.Abs(Optimo1_Etapa1))*100;
        }

        //Para mostrar por pantalla los valores de Xijh
        int[,,] XC = new int[I, J, H];
        for (int i = 0; i < I; i++)
        {
            for (int j = 0; j < J; j++)
            {
                for (int h = 0; h < H; h++)
                {
                    XC[i, j, h] = (int)X_ijh[i, j, h].Get(GRB.DoubleAttr.X);
                    Console.WriteLine($"El valor de XC_{i}.{j}.{h} es: {XC[i, j, h]} ");
                }
```

```
    }
}

//Para mostrar por pantalla los valores de Zsjh
int[,,] ZC = new int[S, J, H];
for (int s = 0; s < S; s++)
{
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            ZC[s, j, h] = (int)Z_sjh[s, j, h].Get(GRB.DoubleAttr.X);
            Console.WriteLine($"El valor de Z_{s}.{j}.{h} es: {ZC[s, j, h]} ");
        }
    }
}

//Para mostrar por pantalla TARDINESS
int[] TC = new int[I];
for (int i = 0; i < I; i++)
{
    TC[i] = (int)T_i[i].Get(GRB.DoubleAttr.X);
    Console.WriteLine($"El valor de TC_{i} es: {TC[i]} ");
}

/* ////////////////////////
 MUESTRO POR PANTALLA LOS VALORES DE LOS PARÁMETROS ALEATORIOS,
AUNQUE LOS PONDRÍA VER MEDIANTE EL EMPLEO DE BREAK POINTS, ME
RESULTA ASÍ MÁS CÓMODO PARA EL ANÁLISIS
////////////////////////*/

//Mostrar por pantalla el valor de u_s para comprobar R6
for (int s = 0; s < S; s++)
{
    Console.WriteLine($" El valor de u_{s} es: {datos_u_s[s]}");
}

//Mostrar por pantalla el valor de y_i para comprobar R7.1 Y R7.2
```

```
for (int i = 0; i < I; i++)
{
  Console.WriteLine($" El valor de y_{i} es: {y_i[i]}");
}

//Mostrar los valores por pantalla de delta_ijh para comprobar R8
for (int i = 0; i < I; i++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         Console.WriteLine($" El valor de delta_{i}.{j}.{h} es: {delta_ijh[i, j, h]}");
      }
   }
}



/*//////////////////
SEGUNDA
DEL MODELO
//////////////////*/

//Declaración de variables (GUROBI)

//CAMA PACU
GRBVar[,,] A_ilh = new GRBVar[I, L, H];
for (int i = 0; i < I; i++)
{
   for (int l = 0; l < L; l++)
   {
      for (int h = 0; h < H; h++)
      {
         A_ilh[i, l, h] = model.AddVar(0, 1, 0, GRB.BINARY, "A_ilh");
      }
   }

}
```

```
//CAMA ICU
GRBVar[,,] B_imw = new GRBVar[I, M, W];
for (int i = 0; i < I; i++)
{
   for (int m = 0; m < M; m++)
   {
      for (int w = 0; w < W; w++)
      {
         B_imw[i, m, w] = model.AddVar(0, 1, 0, GRB.BINARY, "B_imw");
      }
   }
}

//CAMA HOSPITALARIA
GRBVar[,,] C_inw = new GRBVar[I, N, W];
for (int i = 0; i < I; i++)
{
   for (int n = 0; n < N; n++)
   {
      for (int w = 0; w < W; w++)
      {
         C_inw[i, n, w] = model.AddVar(0, 1, 0, GRB.BINARY, "C_inw");
      }
   }
}


//FUNCIÓN OBJETIVO
GRBLinExpr objetivo2 = 0;

for (int i = 0; i < I; i++)
{
   for (int l = 0; l < L; l++)
   {
      for (int h = 0; h < H; h++)
      {
         objetivo2 += ((double)b / (double)2) * A_ilh[i, l, h];
```

```
        }
    }


    for (int w = 0; w < W; w++)
    {
        for (int m = 0; m < M; m++)
        {
            objetivo2 += ((double)b / (double)2) * B_imw[i, m, w];
        }
        for (int n = 0; n < N; n++)
        {
            objetivo2 += ((double)b / (double)2) * C_inw[i, n, w];
        }
    }
}


model.SetObjective(objetivo2, GRB.MAXIMIZE);

/*////////////////////
 RESTRICCIONES
////////////////////*/

//Restricción 1: Solo una cama PACU por paciente

for (int i = 0; i < I; i++)
{
    GRBLinExpr LadoIzqR1 = 0;
    for (int h = 0; h < H; h++)
    {
        for (int l = 0; l < L; l++)
        {
            LadoIzqR1 += A_ilh[i, l, h];
        }
    }
    model.AddConstr(LadoIzqR1 <= 1, "Restricción 1");
}
```

```
//Restricción 2: Capacidad cama PACU

for (int l = 0; l < L; l++)
{
   for (int h = 0; h < H; h++)
   {
      GRBLinExpr LadoIzqR2 = 0;
      for (int i = 0; i < I; i++)
      {
         LadoIzqR2 += t_i_pacu[i] * A_ilh[i, l, h];
      }
      model.AddConstr(LadoIzqR2 <= a_lh[l, h], "Restricción 2");
   }
}

//Restricción 3: Que solo se asigne cama PACU si la necesita y ha sido operado ese mismo día

for (int i = 0; i < I; i++)
{
   for (int h = 0; h < H; h++)
   {
      GRBLinExpr LadoIzqR3 = 0;
      if (beta_i[i] == 1 | beta_i[i] == 2)
      {

         for (int l = 0; l < L; l++)
         {
            LadoIzqR3 += A_ilh[i, l, h];
         }

         for (int j = 0; j < J; j++)
         {
            LadoIzqR3 -= XC[i, j, h];
         }

      }
      model.AddConstr(LadoIzqR3 == 0, "Restricción 3");
```

```
    }
}

// Restricción prueba 1
for (int i = 0; i < I; i++)
{
    for (int l = 0; l < L; l++)
    {
        for (int h = 0; h < H; h++)
        {
            if (beta_i[i] != 1 & beta_i[i] != 2)
            {
                GRBLinExpr LADOIZQR11 = 0;
                LADOIZQR11 = A_ilh[i, l, h];
                model.AddConstr(LADOIZQR11 == 0, "Restricción LADOIZQR11");
            }
        }
    }
}


//Restricción 4: Solo una cama ICU por paciente

for (int i = 0; i < I; i++)
{
    GRBLinExpr LadoIzqR4 = 0;
    for (int w = 0; w < W; w++)
    {
        for (int m = 0; m < M; m++)
        {
            LadoIzqR4 += B_imw[i, m, w];
        }
    }
    model.AddConstr(LadoIzqR4 <= 1, "Restricción 4");
}
```

```
//Restricción 5: Capacidad cama ICU

for (int m = 0; m < M; m++)
{
   for (int w = 0; w < W; w++)
   {
      GRBLinExpr LadoIzqR5 = 0;
      for (int i = 0; i < I; i++)
      {
         LadoIzqR5 += t_i_icu[i] * B_imw[i, m, w];
      }
      model.AddConstr(LadoIzqR5 <= b_mw[m, w], "Restricción 5");
   }
}

//Restricción 6: Que solo se asigne cama ICU si la necesita y ha sido operado en esa semana

for (int i = 0; i < I; i++)
{
   int h = 0; //Inicializo aquí la h para que cuando cambie de semana no me empiece a contar la h desde 0 y siga por el día donde iba.
   for (int w = 0; w < W; w++)
   {

      if (beta_i[i] == 3)
      {
         GRBLinExpr LadoIzqR6 = 0;

         for (int m = 0; m < M; m++)
         {
            LadoIzqR6 += B_imw[i, m, w];
         }

         while (h < H && (VS[h] == w))
         {
            for (int j = 0; j < J; j++)
            {
               LadoIzqR6 -= XC[i, j, h];
```

```
            }

            h++;
          }
          model.AddConstr(LadoIzqR6 == 0, "Restricción 6");
        }
      }
    }

    // Restricción prueba 2
    for (int i = 0; i < I; i++)
    {
      for (int m = 0; m < M; m++)
      {
        for (int w = 0; w < W; w++)
        {
          if (beta_i[i] != 3)
          {
            GRBLinExpr LADOIZQR12 = 0;
            LADOIZQR12 = B_imw[i, m, w];
            model.AddConstr(LADOIZQR12 == 0, "Restricción LADOIZQR12");
          }
        }
      }
    }

    //Restricción 7: Solo una cama HOSPITALARIA por paciente

    for (int i = 0; i < I; i++)
    {
      for (int w = 0; w < W; w++)
      {
        GRBLinExpr LadoIzqR7 = 0;
        for (int n = 0; n < N; n++)
        {
          LadoIzqR7 += C_inw[i, n, w];
        }
        model.AddConstr(LadoIzqR7 <= 1, "Restricción 7");
```

```
    }
  }

  //Restricción 8: Capacidad cama HOSPITALARIA

  for (int n = 0; n < N; n++)
  {
    for (int w = 0; w < W; w++)
    {
      GRBLinExpr LadoIzqR8 = 0;
      for (int i = 0; i < I; i++)
      {
        LadoIzqR8 += t_i_hospitalaria[i] * C_inw[i, n, w];
      }
      model.AddConstr(LadoIzqR8 <= c_nw[n, w], "Restricción 8");
    }
  }

  //Restricción 9.1: Que no asigne cama hospitalaria si no ha estado previamente en PACU

  for (int i = 0; i < I; i++)
  {
    int h = 0; //Inicializo aquí la h para que cuando cambie de semana no me empiece a contar la h desde 0 y siga por el día donde iba.
    for (int w = 0; w < W; w++)
    {
      if (beta_i[i] == 2)
      {
        GRBLinExpr LadoIzqR91 = 0;

        for (int n = 0; n < N; n++)
        {
          LadoIzqR91 += C_inw[i, n, w];
        }
        while (h < H && (VS[h] == w))
        {

          for (int l = 0; l < L; l++)
```

```
            {
                LadoIzqR91 -= A_ilh[i, l, h];
            }

            h++;
        }

        model.AddConstr(LadoIzqR91 == 0, "Restricción 9.1");
    }
  }
}

//Restricción 9.2: Que no asigne cama hospitalaria si no ha estado previamente en ICU


for (int i = 0; i < I; i++)
{
  for (int w = 0; w < W; w++)
  {
    if (beta_i[i] == 3)
    {
      GRBLinExpr LadoIzqR92 = 0;

      for (int n = 0; n < N; n++)
      {
        LadoIzqR92 += C_inw[i, n, w];
      }

      for (int m = 0; m < M; m++)
      {
        LadoIzqR92 -= B_imw[i, m, w];
      }
      model.AddConstr(LadoIzqR92 == 0, "Restricción 9.2");
    }

  }
}
```

```
// Restricción prueba 3
for (int i = 0; i < I; i++)
{
   for (int n = 0; n < N; n++)
   {
      for (int w = 0; w < W; w++)
      {
         if (beta_i[i] != 2 & beta_i[i] != 3)
         {
            GRBLinExpr LADOIZQR13 = 0;
            LADOIZQR13 = C_inw[i, n, w];
            model.AddConstr(LADOIZQR13 == 0, "Restricción LADOIZQR13");
         }
      }
   }
}

//Restricción 10
for (int i = 0; i < I; i++)
{
   for (int l = 0; l < L; l++)
   {
      for (int h = 0; h < H; h++)
      {
         if (beta_i[i] == 0)
         {
            GRBLinExpr LadoIzq10 = 0;
            LadoIzq10 = A_ilh[i, l, h];
            model.AddConstr(LadoIzq10 == 0, "Restricción 10");
         }

      }
   }

   for (int w = 0; w < W; w++)
   {
      for (int m = 0; m < M; m++)
      {
```

```
                if (beta_i[i] == 0)
                {
                    GRBLinExpr LadoIzq10 = 0;
                    LadoIzq10 = B_imw[i, m, w];
                    model.AddConstr(LadoIzq10 == 0, "Restricción 10");
                }
            }

            for (int n = 0; n < N; n++)
            {
                if (beta_i[i] == 0)
                {
                    GRBLinExpr LadoIzq10 = 0;
                    LadoIzq10 = C_inw[i, n, w];
                    model.AddConstr(LadoIzq10 == 0, "Restricción 10");
                }

            }
        }

    }

    model.GetEnv().Set(GRB.DoubleParam.TimeLimit, t_limite/(double)2);

    if (S_Infactible1_Etapa1 != 1)
    {
        model.Optimize();
    } else
    {
        Console.WriteLine("La primera etapa del modelo no tiene solución, por lo tanto, esta etapa no se puede resolver");
    }

    //Tiempo de ejecución en segundos
    tiempo_ejecución1_Etapa2 = model.Get(GRB.DoubleAttr.Runtime);

    //Recogida análisis soluciones
    int optimstatus2 = model.Get(GRB.IntAttr.Status);
```

```
//Si el modelo es infactible o ilimitado se reoptimiza
if (optimstatus2 == GRB.Status.INF_OR_UNBD)
{
   model.GetEnv().Set(GRB.IntParam.Presolve, 0);
   model.Optimize();
   tiempo_ejecución1_Etapa2 = model.Get(GRB.DoubleAttr.Runtime);
   optimstatus2 = model.Get(GRB.IntAttr.Status);
}

//Modelo óptimo
if (optimstatus2 == GRB.Status.OPTIMAL)
{
   Optimo1_Etapa2 = model.Get(GRB.DoubleAttr.ObjVal);
   S_Optima1_Etapa2 = 1;
   S_Factible1_Etapa2 = 0;
   S_Infactible1_Etapa2 = 0;
   Bound1_Etapa2 = Optimo1_Etapa2;
   Gap1_Etapa2 = 0;

}
else if (optimstatus2 == GRB.Status.INFEASIBLE)
{
   S_Factible1_Etapa2 = 0;
   S_Infactible1_Etapa2 = 1;
   Bound1_Etapa2 = 0;
   Optimo1_Etapa2 = 0;
   Gap1_Etapa2 = 0;
}
else if (optimstatus2 == 9)
{
   S_Factible1_Etapa2 = 1;
   S_Infactible1_Etapa2 = 0;
   S_Optima1_Etapa2 = 0;
   Bound1_Etapa2 = model.Get(GRB.DoubleAttr.ObjBound); //Lower bound
   Optimo1_Etapa2 = model.Get(GRB.DoubleAttr.ObjVal); //Best bound
   Gap1_Etapa2 = (Math.Abs(Bound1_Etapa2 - Optimo1_Etapa2) /
Math.Abs(Optimo1_Etapa2))*100; //Holgura
```

```
}



//Mostrar por pantalla los valores de A_ilh
int[,,] A = new int[I, L, H];
for (int i = 0; i < I; i++)
{
   for (int l = 0; l < L; l++)
   {
      for (int h = 0; h < H; h++)
      {
         A[i, l, h] = (int)A_ilh[i, l, h].Get(GRB.DoubleAttr.X);
         Console.WriteLine($"El valor de A_{i}.{l}.{h} es: {A[i, l, h]} ");
      }
   }
}

//Mostrar por pantalla los valores de B_imw
int[,,] B = new int[I, M, W];
for (int i = 0; i < I; i++)
{
   for (int m = 0; m < M; m++)
   {
      for (int w = 0; w < W; w++)
      {
         B[i, m, w] = (int)B_imw[i, m, w].Get(GRB.DoubleAttr.X);
         Console.WriteLine($"El valor de B_{i}.{m}.{w} es: {B[i, m, w]} ");
      }
   }
}

//Mostrar por pantalla los valores de C_inw
int[,,] C = new int[I, N, W];
for (int i = 0; i < I; i++)
{
   for (int n = 0; n < N; n++)
   {
```

```
            for (int w = 0; w < W; w++)
            {
               C[i, n, w] = (int)C_inw[i, n, w].Get(GRB.DoubleAttr.X);
               Console.WriteLine($"El valor de C_{i}.{n}.{w} es: {C[i, n, w]} ");
            }
         }
      }

      //Mostrar por pantalla valores parámetros
      for (int i = 0; i < I; i++)
      {
         Console.WriteLine($"beta_i{i}={beta_i[i]}");
      }
      for (int i = 0; i < I; i++)
      {
         Console.WriteLine($"t_pacu{i}={t_i_pacu[i]}");
      }

      for (int i = 0; i < I; i++)
      {
        Console.WriteLine($"t_icu{i}={t_i_icu[i]}");
      }

      for (int i = 0; i < I; i++)
      {
         Console.WriteLine($"t_hospitalaria{i}={t_i_hospitalaria[i]}");
      }


      model.Dispose();
      env.Dispose();

   }
   catch (GRBException e) { Console.WriteLine("Código de error: " + e.ErrorCode + " " + e.Message); }
}


/*////////////////////////////////
```

```
MODELO INTEGRACIÓN
////////////////////////////////*/
static void Integración(int H, int I, int J, int S, int[,] datos_r_jh, int[,] datos_a_sh, int[] datos_u_s, int[] y_i, int[] datos_rd_i, int[] datos_d_i, int[] datos_t_i, double[] datos_w_i, int[,,] delta_ijh, int W, int L, int M, int N, int[] t_i_pacu, int[] t_i_icu, int[] t_i_hospitalaria, int[] beta_i, int[,] a_lh, int[,] b_mw, int[,] c_nw, double Wmax, int a, int b, int[] VS, double t_limite, ref int S_Optima2, ref int S_Factible2, ref int S_Infactible2, ref double Optimo2, ref double Gap2, ref double Bound2, ref double tiempo_ejecución2)
{
  try
  {
    //Creación entorno vacio e iniciación
    GRBEnv env = new GRBEnv("mip1.log");

    //Creación modelo vacío
    GRBModel model = new GRBModel(env);
    model.Set(GRB.StringAttr.ModelName, "Integración (E1+E2)");

    /*//////////////////////////
    DECLARACIÓN DE VARIABLES
    //////////////////////////*/

    //ETAPA 1

    //PROGRAMACIÓN CIRUGÍA I EN OP J EL DÍA H
    GRBVar[,,] X_ijh = new GRBVar[I, J, H];
    for (int i = 0; i < I; i++)
    {
      for (int j = 0; j < J; j++)
      {
        for (int h = 0; h < H; h++)
        { X_ijh[i, j, h] = model.AddVar(0, 1, 0, GRB.BINARY, "X_" + i + "_" + j + "_" + h); }
      }
    }

    //PROGRAMACIÓN CIRUJANO S EN OP J EL DÍA H

    GRBVar[,,] Z_sjh = new GRBVar[S, J, H];
    for (int s = 0; s < S; s++)
    {
```

```
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        { Z_sjh[s, j, h] = model.AddVar(0, 1, 0, GRB.BINARY, "Z_" + s + "_" + j + "_" + h); }
    }
}

//Definición tardiness
GRBVar[] T_i = new GRBVar[I];
for (int i = 0; i < I; i++)
{
    T_i[i] = model.AddVar(0, GRB.INFINITY, 0, GRB.INTEGER, "T_" + i);
}

//Es necesario crear esta variable objetivo para poder unificar las dos funciones del objetivo (ETAPA
1)
GRBVar objetivo1 = new GRBVar();
objetivo1 = model.AddVar(0, GRB.INFINITY, 1, GRB.CONTINUOUS, "Objetivo 1");
//El -1 es el término con el que esta variable entra multiplicando en la F.O

//ETAPA 2
//CAMA PACU
GRBVar[,,] A_ilh = new GRBVar[I, L, H];
for (int i = 0; i < I; i++)
{
    for (int l = 0; l < L; l++)
    {
        for (int h = 0; h < H; h++)
        {
            A_ilh[i, l, h] = model.AddVar(0, 1, 0, GRB.BINARY, "A_ilh");
        }
    }
}

//CAMA ICU
GRBVar[,,] B_imw = new GRBVar[I, M, W];
for (int i = 0; i < I; i++)
{
```

```
            for (int m = 0; m < M; m++)
            {
                for (int w = 0; w < W; w++)
                {
                    B_imw[i, m, w] = model.AddVar(0, 1, 0, GRB.BINARY, "B_imw");
                }
            }
        }

        //CAMA HOSPITALARIA
        GRBVar[,,] C_inw = new GRBVar[I, N, W];
        for (int i = 0; i < I; i++)
        {
            for (int n = 0; n < N; n++)
            {
                for (int w = 0; w < W; w++)
                {
                    C_inw[i, n, w] = model.AddVar(0, 1, 0, GRB.BINARY, "C_inw");
                }
            }
        }

        //variable objetivo para poder unificar las dos funciones del objetivo (ETAPA 2)
        GRBVar objetivo2 = new GRBVar();
        objetivo2 = model.AddVar(0, GRB.INFINITY, ((double)b / (double)2), GRB.CONTINUOUS,
"Objetivo 1");



        /*////////////////////////////
        FUNCIÓN OBJETIVO INTEGRADA
        ////////////////////////////*/

        model.Set(GRB.IntAttr.ModelSense, -1); //Se pone -1 porque estamos MAX
        /*////////////////////
        RESTRICCIONES DEL OBJETIVO
        ////////////////////*/
        ///
        //ETAPA 1 + Minimización del tardiness.
```

```
GRBLinExpr restr_objetivo1 = 0; //Declarada e iniciada

for (int h = 0; h < H; h++)
{
   double numero = (double)1 / (double)(h + 1);
   for (int i = 0; i < I; i++)
   {
      for (int j = 0; j < J; j++)
      {
         restr_objetivo1 += ((double)1 / (double)Wmax) * (numero * datos_w_i[i] * X_ijh[i, j, h]);
      }
   }
}

for (int i = 0; i < I; i++)
{
   restr_objetivo1 -= ((double)a / (double)(H * Wmax)) * (T_i[i] * datos_w_i[i]);
}
model.AddConstr(objetivo1 - restr_objetivo1 == 0, "Restricción Objetivo 1");


//ETAPA 2

GRBLinExpr restr_objetivo2 = 0;

for (int i = 0; i < I; i++)
{
   for (int l = 0; l < L; l++)
   {
      for (int h = 0; h < H; h++)
      {
         restr_objetivo2 += A_ilh[i, l, h];
      }
   }


   for (int w = 0; w < W; w++)
```

```
    {
        for (int m = 0; m < M; m++)
        {
            restr_objetivo2 += B_imw[i, m, w];
        }
        for (int n = 0; n < N; n++)
        {
            restr_objetivo2 += C_inw[i, n, w];
        }
    }
}

model.AddConstr(objetivo2 - restr_objetivo2 == 0, "Restricción Objetivo 2");


/*//////////////////////////////
 RESTRICCIONES DEL MODELO INTEGRADO
//////////////////////////////////*/

//Restricciones TARDINESS

//Restricción tardiness para pacientes operados en el horizonte H
for (int i = 0; i < I; i++)
{
    GRBLinExpr ladoIzqT_1 = 0;
    ladoIzqT_1 += T_i[i];
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            ladoIzqT_1 -= X_ijh[i, j, h] * h; //Cálculo del día de operación
        }
    }
    ladoIzqT_1 += datos_d_i[i];
    model.AddConstr(ladoIzqT_1 >= 0, "Tardiness_1");
}

//Restricción tardiness para pacientes no operados en H
```

```
int Ult_dia = H;
for (int i = 0; i < I; i++)
{
   GRBLinExpr ladoIzqT_2 = 0;
   ladoIzqT_2 += T_i[i];
   ladoIzqT_2 -= Ult_dia;
   ladoIzqT_2 += datos_d_i[i];
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         ladoIzqT_2 += X_ijh[i, j, h] * (Ult_dia - datos_d_i[i]); //Cálculo del día de operación
      }
   }
   model.AddConstr(ladoIzqT_2 >= 0, "Tardiness_2");
}

//ETAPA 1

//Restricción 2: No se programe cirugía más de una vez en HP.
for (int i = 0; i < I; i++)
{
   GRBLinExpr ladoIzqR2 = 0;
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         ladoIzqR2 += X_ijh[i, j, h];
      }
   }
   model.AddConstr(ladoIzqR2 <= 1, "Restricción 2");
}

//Restricción 3.1: Que se programe la cirugia dsps del release time
for (int i = 0; i < I; i++)
{
   GRBLinExpr ladoIzqR31 = 0;
   for (int j = 0; j < J; j++)
```

```
    {
        for (int h = 0; h < datos_rd_i[i] - 1 && h<H; h++)
        {
            ladoIzqR31 += X_ijh[i, j, h];
        }
    }
    model.AddConstr(ladoIzqR31 == 0, "Restricción 3.1");
}

//Restricción 4: Capacidad OP diaria
for (int j = 0; j < J; j++)
{
    for (int h = 0; h < H; h++)
    {
        GRBLinExpr ladoIzqR4 = 0;
        for (int i = 0; i < I; i++)
        {
            ladoIzqR4 += datos_t_i[i] * X_ijh[i, j, h];
            model.AddConstr(ladoIzqR4 <= datos_r_jh[j, h], "Restricción 4");
        }
    }
}

//Restricción 5: Capacidad cirujano diaria
for (int s = 0; s < S; s++)
{
    for (int h = 0; h < H; h++)
    {
        GRBLinExpr ladoIzqR5 = 0;
        for (int j = 0; j < J; j++)
        {
            for (int i = 0; i < I; i++)
            {
                if (y_i[i] == s)
                { ladoIzqR5 += datos_t_i[i] * X_ijh[i, j, h]; }
            }
        }
        model.AddConstr(ladoIzqR5 <= datos_a_sh[s, h], "Restricción 5");
```

```
    }
}

//Restricción 6: Asignación de quirófanos
for (int s = 0; s < S; s++)
{
    for (int h = 0; h < H; h++)
    {
        GRBLinExpr ladoIzqR6 = 0;
        for (int j = 0; j < J; j++)
        {
            ladoIzqR6 += Z_sjh[s, j, h];
        }
        model.AddConstr(ladoIzqR6 <= datos_u_s[s], "Restricción 6");
    }
}

// Restricción 7.1 : Capacidad quirófano

for (int s = 0; s < S; s++)
{
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            GRBLinExpr ladoIzqR71 = 0;

            for (int i = 0; i < I; i++)
            {
                if (y_i[i] == s)
                {
                    ladoIzqR71 += datos_t_i[i] * X_ijh[i, j, h];
                }
            }
            ladoIzqR71 -= datos_r_jh[j, h] * Z_sjh[s, j, h];
```

```
            model.AddConstr(ladoIzqR71 <= 0, "Restricción 7.1");
        }
    }
}

//Restricción 7.2: No asignar cirujano si no hay cirugías
for (int s = 0; s < S; s++)
{
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            GRBLinExpr ladoIzqR72 = 0;
            for (int i = 0; i < I; i++)
            {
                if (y_i[i] == s)
                {
                    ladoIzqR72 += X_ijh[i, j, h];
                }

            }
            ladoIzqR72 -= Z_sjh[s, j, h];
            model.AddConstr(ladoIzqR72 >= 0, "Restricción 7.2 ");
        }
    }
}

//Restricción 8: Garantiza que cada cirugía se lleve a cabo en un OR adecuado.
for (int i = 0; i < I; i++)
{
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            if (delta_ijh[i, j, h] == 0)
            {
                GRBLinExpr ladoIzqR8 = 0;
                ladoIzqR8 = X_ijh[i, j, h];
```

```
                model.AddConstr(ladoIzqR8 == 0, "Restricción 8");
            }
        }
    }
}

//Restricción 11: tardiness >= 0;
for (int i = 0; i < I; i++)
{
    GRBLinExpr LadoIzq11 = 0;
    LadoIzq11 = T_i[i];
    model.AddConstr(LadoIzq11 >= 0, "Restricción 11");
}

//ETAPA 2

//Restricción 1: Solo una cama PACU por paciente

for (int i = 0; i < I; i++)
{
    GRBLinExpr LadoIzqR1 = 0;
    for (int h = 0; h < H; h++)
    {
        for (int l = 0; l < L; l++)
        {
            LadoIzqR1 += A_ilh[i, l, h];
        }

    }
    model.AddConstr(LadoIzqR1 <= 1, "Restricción 1");
}

//Restricción 2: Capacidad cama PACU

for (int l = 0; l < L; l++)
{
    for (int h = 0; h < H; h++)
    {
```

```
        GRBLinExpr LadoIzqR2 = 0;
        for (int i = 0; i < I; i++)
        {
            LadoIzqR2 += t_i_pacu[i] * A_ilh[i, l, h];
        }
        model.AddConstr(LadoIzqR2 <= a_lh[l, h], "Restricción 2");
    }
}

//Restricción 3: Que solo se asigne cama PACU si la necesita y ha sido operado ese mismo día

for (int i = 0; i < I; i++)
{
    for (int h = 0; h < H; h++)
    {
        GRBLinExpr LadoIzqR3 = 0;
        if (beta_i[i] == 1 | beta_i[i] == 2)
        {

            for (int l = 0; l < L; l++)
            {
                LadoIzqR3 += A_ilh[i, l, h];
            }

            for (int j = 0; j < J; j++)
            {
                LadoIzqR3 -= X_ijh[i, j, h];
            }

        }
        model.AddConstr(LadoIzqR3 == 0, "Restricción 3");

    }
}

// Restricción 11
for (int i = 0; i < I; i++)
{
```

```
    for (int l = 0; l < L; l++)
    {
        for (int h = 0; h < H; h++)
        {
            if (beta_i[i] != 1 & beta_i[i] != 2)
            {
                GRBLinExpr LADOIZQR11 = 0;
                LadoIzqR11 = A_ilh[i, l, h];
                model.AddConstr(LADOIZQR11 == 0, "Restricción LADOIZQR11");
            }
        }
    }
}


//Restricción 4: Solo una cama ICU por paciente

for (int i = 0; i < I; i++)
{
    GRBLinExpr LadoIzqR4 = 0;
    for (int w = 0; w < W; w++)
    {
        for (int m = 0; m < M; m++)
        {
            LadoIzqR4 += B_imw[i, m, w];
        }
    }
    model.AddConstr(LadoIzqR4 <= 1, "Restricción 4");
}

//Restricción 5: Capacidad cama ICU

for (int m = 0; m < M; m++)
{
    for (int w = 0; w < W; w++)
    {
        GRBLinExpr LadoIzqR5 = 0;
        for (int i = 0; i < I; i++)
```

```
        {
            LadoIzqR5 += t_i_icu[i] * B_imw[i, m, w];
        }
        model.AddConstr(LadoIzqR5 <= b_mw[m, w], "Restricción 5");
    }
}

//Restricción 6: Que solo se asigne cama ICU si la necesita y ha sido operado en esa semana

for (int i = 0; i < I; i++)
{
    int h = 0;
    for (int w = 0; w < W; w++)
    {
        if (beta_i[i] == 3)
        {
            GRBLinExpr LadoIzqR6 = 0;

            for (int m = 0; m < M; m++)
            {
                LadoIzqR6 += B_imw[i, m, w];
            }
            while (h < H && (VS[h] == w))
            {
                for (int j = 0; j < J; j++)
                {
                    LadoIzqR6 -= X_ijh[i, j, h];
                }
                h++;
            }
            model.AddConstr(LadoIzqR6 == 0, "Restricción 6");
        }
    }
}

// Restricción 12
for (int i = 0; i < I; i++)
{
```

```
    for (int m = 0; m < M; m++)
    {
        for (int w = 0; w < W; w++)
        {
            if (beta_i[i] != 3)
            {
                GRBLinExpr LADOIZQR12 = 0;
                LADOIZQR12 = B_imw[i, m, w];
                model.AddConstr(LADOIZQR12 == 0, "Restricción LADOIZQR12");
            }
        }
    }
}

//Restricción 7: Solo una cama HOSPITALARIA por paciente

for (int i = 0; i < I; i++)
{
    for (int w = 0; w < W; w++)
    {
        GRBLinExpr LadoIzqR7 = 0;
        for (int n = 0; n < N; n++)
        {
            LadoIzqR7 += C_inw[i, n, w];
        }
        model.AddConstr(LadoIzqR7 <= 1, "Restricción 7");
    }
}

//Restricción 8: Capacidad cama HOSPITALARIA

for (int n = 0; n < N; n++)
{
    for (int w = 0; w < W; w++)
    {
        GRBLinExpr LadoIzqR8 = 0;
        for (int i = 0; i < I; i++)
        {
```

```
            LadoIzqR8 += t_i_hospitalaria[i] * C_inw[i, n, w];
        }
        model.AddConstr(LadoIzqR8 <= c_nw[n, w], "Restricción 8");
    }
}

//Restricción 9.1: Que no asigne cama hospitalaria si no ha estado previamente en PACU

for (int i = 0; i < I; i++)
{
    int h = 0;
    for (int w = 0; w < W; w++)
    {
        if (beta_i[i] == 2)
        {
            GRBLinExpr LadoIzqR91 = 0;

            for (int n = 0; n < N; n++)
            {
                LadoIzqR91 += C_inw[i, n, w];
            }

            while (h < H && (VS[h] == w))
            {

                for (int l = 0; l < L; l++)
                {
                    LadoIzqR91 -= A_ilh[i, l, h];
                }

                h++;
            }

            model.AddConstr(LadoIzqR91 == 0, "Restricción 9.1");
        }
    }
}
```

```
//Restricción 9.2: Que no asigne cama hospitalaria si no ha estado previamente en ICU

for (int i = 0; i < I; i++)
{
   for (int w = 0; w < W; w++)
   {
      if (beta_i[i] == 3)
      {
         GRBLinExpr LadoIzqR92 = 0;

         for (int n = 0; n < N; n++)
         {
            LadoIzqR92 += C_inw[i, n, w];
         }

         for (int m = 0; m < M; m++)
         {
            LadoIzqR92 -= B_imw[i, m, w];
         }
         model.AddConstr(LadoIzqR92 == 0, "Restricción 9.2");
      }
   }
}

// Restricción 13
for (int i = 0; i < I; i++)
{
   for (int n = 0; n < N; n++)
   {
      for (int w = 0; w < W; w++)
      {
         if (beta_i[i] != 2 & beta_i[i] != 3)
         {
            GRBLinExpr LADOIZQR13 = 0;
            LADOIZQR13 = C_inw[i, n, w];
            model.AddConstr(LADOIZQR13 == 0, "Restricción LADOIZQR13");
         }
      }
```

```
    }
}

//Restricción 10
for (int i = 0; i < I; i++)
{
    for (int l = 0; l < L; l++)
    {
        for (int h = 0; h < H; h++)
        {
            if (beta_i[i] == 0)
            {
                GRBLinExpr LadoIzq10 = 0;
                LadoIzq10 = A_ilh[i, l, h];
                model.AddConstr(LadoIzq10 == 0, "Restricción 10");
            }
        }
    }

    for (int w = 0; w < W; w++)
    {
        for (int m = 0; m < M; m++)
        {
            if (beta_i[i] == 0)
            {
                GRBLinExpr LadoIzq10 = 0;
                LadoIzq10 = B_imw[i, m, w];
                model.AddConstr(LadoIzq10 == 0, "Restricción 10");
            }
        }

        for (int n = 0; n < N; n++)
        {
            if (beta_i[i] == 0)
            {
                GRBLinExpr LadoIzq10 = 0;
                LadoIzq10 = C_inw[i, n, w];
                model.AddConstr(LadoIzq10 == 0, "Restricción 10");
```

```
            }
        }
    }

}


/*//////////////////////////
 OPTIMIZACIÓN DEL MODELO
//////////////////////////*/
model.GetEnv().Set(GRB.DoubleParam.TimeLimit, t_limite);
model.Optimize();

//Tiempo de ejecución en segundos
tiempo_ejecución2 = model.Get(GRB.DoubleAttr.Runtime);

//Recogida análisis soluciones
int optimstatus = model.Get(GRB.IntAttr.Status);

//Si el modelo es infactible o ilimitado se reoptimiza
if(optimstatus==GRB.Status.INF_OR_UNBD)
{
    model.GetEnv().Set(GRB.IntParam.Presolve, 0);
    model.Optimize();
    tiempo_ejecución2 = model.Get(GRB.DoubleAttr.Runtime);
    optimstatus = model.Get(GRB.IntAttr.Status);
}

//Modelo óptimo
if (optimstatus==GRB.Status.OPTIMAL)
{
    Optimo2 = model.Get(GRB.DoubleAttr.ObjVal);
    S_Optima2 = 1;
    S_Factible2 = 0;
    S_Infactible2 = 0;
    Bound2 = Optimo2;
    Gap2 = 0;
```

```
}
else if(optimstatus == GRB.Status.INFEASIBLE)
{
   S_Factible2 = 0;
   S_Infactible2 = 1;
   Bound2 = 0;
   Optimo2 = 0;
   Gap2 = 0;
} else if (optimstatus==9)
{
   S_Factible2 = 1;
   S_Infactible2 = 0;
   S_Optima2 = 0;
   Bound2 = model.Get(GRB.DoubleAttr.ObjBound); //Lower bound
   Optimo2 = model.Get(GRB.DoubleAttr.ObjVal); //Best bound
   Gap2 = (Math.Abs(Bound2 - Optimo2) / Math.Abs(Optimo2))*100;
}


/*//////////////////////////
 IMPRIMIR POR PANTALLA
//////////////////////////*/

//ETAPA 1

/*//////////////////////////
  VARIABLES
//////////////////////////*/

// Xijh
int[,,] XC = new int[I, J, H];
for (int i = 0; i < I; i++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
         XC[i, j, h] = (int)X_ijh[i, j, h].Get(GRB.DoubleAttr.X);
```

```
            Console.WriteLine($"El valor de XC_{i}.{j}.{h} es: {XC[i, j, h]} ");
        }
    }
}

// Zsjh
int[,,] ZC = new int[S, J, H];
for (int s = 0; s < S; s++)
{
    for (int j = 0; j < J; j++)
    {
        for (int h = 0; h < H; h++)
        {
            ZC[s, j, h] = (int)Z_sjh[s, j, h].Get(GRB.DoubleAttr.X);
            Console.WriteLine($"El valor de Z_{s}.{j}.{h} es: {ZC[s, j, h]} ");
        }
    }
}



// Ti
int[] TC = new int[I];
for (int i = 0; i < I; i++)
{
    TC[i] = (int)T_i[i].Get(GRB.DoubleAttr.X);
    Console.WriteLine($"El valor de TC_{i} es: {TC[i]} ");
}

/*//////////////////
 PARÁMETROS
NOTA: Lo puedo ver mediante break points pero así me resulta más comodo
//////////////////*/

// u_s
for (int s = 0; s < S; s++)
{
   Console.WriteLine($" El valor de u_{s} es: {datos_u_s[s]}");
}
```

```
//y_i
for (int i = 0; i < I; i++)
{
  Console.WriteLine($" El valor de y_{i} es: {y_i[i]}");
}

//delta_ijh
for (int i = 0; i < I; i++)
{
   for (int j = 0; j < J; j++)
   {
      for (int h = 0; h < H; h++)
      {
        Console.WriteLine($" El valor de delta_{i}.{j}.{h} es: {delta_ijh[i, j, h]}");
      }
   }
}

//ETAPA 2

/*//////////////////////////
 VARIABLES
//////////////////////////*/

// A_ilh
int[,,] A = new int[I, L, H];
for (int i = 0; i < I; i++)
{
   for (int l = 0; l < L; l++)
   {
      for (int h = 0; h < H; h++)
      {
         A[i, l, h] = (int)A_ilh[i, l, h].Get(GRB.DoubleAttr.X);
         Console.WriteLine($"El valor de A_{i}.{l}.{h} es: {A[i, l, h]} ");
      }
   }
}
```

```
// B_imw
int[,,] B = new int[I, M, W];
for (int i = 0; i < I; i++)
{
   for (int m = 0; m < M; m++)
   {
      for (int w = 0; w < W; w++)
      {
         B[i, m, w] = (int)B_imw[i, m, w].Get(GRB.DoubleAttr.X);
         Console.WriteLine($"El valor de B_{i}.{m}.{w} es: {B[i, m, w]} ");
      }
   }
}

// C_inw
int[,,] C = new int[I, N, W];
for (int i = 0; i < I; i++)
{
   for (int n = 0; n < N; n++)
   {
      for (int w = 0; w < W; w++)
      {
         C[i, n, w] = (int)C_inw[i, n, w].Get(GRB.DoubleAttr.X);
         Console.WriteLine($"El valor de C_{i}.{n}.{w} es: {C[i, n, w]} ");
      }
   }
}

/*//////////////////
PARÁMETROS
//////////////////*/
for (int i = 0; i < I; i++)
{
   //Console.WriteLine($"beta{i}={beta_i[i]}");
}
for (int i = 0; i < I; i++)
{
```

```
            // Console.WriteLine($"t_pacu{i}={t_i_pacu[i]}");
          }
          for (int i = 0; i < I; i++)
          {
            // Console.WriteLine($"t_icu{i}={t_i_icu[i]}");
          }
          for (int i = 0; i < I; i++)
          {
            // Console.WriteLine($"t_hospitalaria{i}={t_i_hospitalaria[i]}");
          }


          //Vaciar modelo
          model.Dispose();
          env.Dispose();
        }
        catch (GRBException e) { Console.WriteLine("Código de error: " + e.ErrorCode + " " + e.Message); }

      }


    }
}
```