Trabajo Fin de Máster

Asignación de camas de hospital a enfermos de urgencias y programados

Alberto Portela Peleteiro

Máster en Técnicas Estadísticas

Curso 2022-2023

Propuesta de Trabajo Fin de Máster

Título en galego: Asignación de camas de hospital a enfermos de urxencias e programados
Título en español: Asignación de camas de hospital a enfermos de urgencias y programados
English title: Assignment of hospital beds to emergency and scheduled patients
Modalidad: Modalidad B
Autor: Alberto Portela Peleteiro, Universidade de Santiago de Compostela
Director: Alejandro Saavedra Nieves, Universidade de Santiago de Compostela
Tutor: Francisco Reyes Santías, Fundación Instituto de Investigación Sanitaria (FIDIS)
Breve resumen del trabajo: En el ámbito hospitalario, la asignación eficiente de camas a pacientes es indispensable para garantizar una atención médica de calidad y una gestión adecuada de los recursos hospitalarios. En este trabajo se busca dar solución a este problema desde un punto de vista matemático introduciendo varios modelos, tanto determinísticos como heurísticos, comparándolos entre sí mediante ejemplos con datos reales. Las conclusiones extraídas de los resultados facilitan la elección de las herramientas adecuadas en función de las necesidades y ayudan a mejorar la calidad de la atención médica en los hospitales.
Recomendaciones:
Otras observaciones:

Don Alejandro Saavedra Nieves, Profesor Ayudante Doctor de la Universidade de Santiago de Compostela y don Francisco Reyes Santías, Responsable de Evaluación Económica de la Fundación Instituto de Investigación Sanitaria (FIDIS), informan que el Trabajo Fin de Máster titulado

Asignación de camas de hospital a enfermos de urgencias y programados

fue realizado bajo su dirección por don Alberto Portela Peleteiro para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 12 de julio de 2023.

SAAVEDRA NIEVES ALEJANDRO - 45873390M
Firmado digitalmente por SAAVEDRA NIEVES ALEJANDRO - 45873390M
Fecha: 2023.07.13 19:27:06 +02'00'

El director:
Don Alejandro Saavedra Nieves

REYES SANTIAS FRANCISCO - 33245231L
Firmado digitalmente por REYES SANTIAS FRANCISCO - 33245231L
Fecha: 2023.07.14 08:52:28 +02'00'

El tutor:
Don Francisco Reyes Santías

PORTELA PELETEIRO ALBERTO - 77465699C
Firmado digitalmente por PORTELA PELETEIRO ALBERTO - 77465699C
Fecha: 2023.07.12 22:36:40 +02'00'

El autor:
Don Alberto Portela Peleteiro

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,...sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen

Resumen en español

El problema de asignación óptima de camas a pacientes en el ámbito hospitalario, denotado por las siglas PBA (en inglés, *patient to bed assignment*), es fundamental para garantizar una atención médica de calidad y una gestión eficiente de los recursos hospitalarios en las sociedades modernas. En este trabajo se proporcionan distintas soluciones al PBA en el Complejo Hospitalario de Santiago de Compostela (CHUS) desde un punto de vista matemático, concretamente desde la perspectiva de la optimización matemática. Para ello, se consideran varias propuestas, basadas en formulaciones de problemas de programación matemática en sus versiones determinísticas y estocásticas. Además, la inmediatez que caracteriza la toma de decisiones en este contexto justifica la propuesta de procedimientos heurísticos para la resolución de las diferentes formulaciones del PBA presentadas. En todos ellos, se han comparado los resultados proporcionados con la realidad diaria del hospital en la base de datos del CHUS proporcionada, lo que permite valorar la idoneidad del uso de cada una de ellas.

English abstract

The optimal bed-to-patient assignment problem in the hospital setting, denoted by the acronym PBA (patient to bed assignment), is fundamental to guarantee quality medical care and efficient management of hospital resources in modern societies. This master thesis provides different solutions to the PBA in the Complejo Hospitalario de Santiago de Compostela (CHUS) from a mathematical point of view, specifically from the perspective of optimization. For this purpose, several proposals are considered, based on formulations of mathematical programming problems in their deterministic and stochastic versions. Moreover, the immediacy that characterises decision-making in this context justifies the proposal of heuristic procedures for the resolution of the different PBA formulations presented. In all of them, the results provided have been compared with the daily reality of the hospital, assessing the suitability of the applicability of each of them.

Prefacio

En el Hospital Clínico Universitario de Santiago de Compostela ingresan diariamente, en media, más de 100 pacientes con patología no ambulatoria, es decir, que requieren de un ingreso en una habitación del hospital para poder tratarla. Esto se traduce en más de 35000 ingresos anuales en el hospital. Además, algunas crisis recientes como el colapso sanitario provocado por el virus del COVID-19 en el año 2020 o el accidente ferroviario de Angrois en el año 2013, motivan la necesidad de mantener un nivel de ocupación hospitalaria no excesivamente elevado, reservando un cierto número de camas libres en previsión de una posible situación de emergencia como las aquí mencionadas. El elevado número de ingresos que se producen anualmente en el hospital, así como la necesidad de mantener un nivel relativamente bajo de ocupación hospitalaria en previsión de evitar un colapso, motivan que la asignación de los pacientes a las camas del hospital se deba realizar de una manera óptima para buscar maximizar el número de patologías tratadas, así como evitar largas listas de espera, que derivarían en un importante empobrecimiento en la calidad de la atención médica.

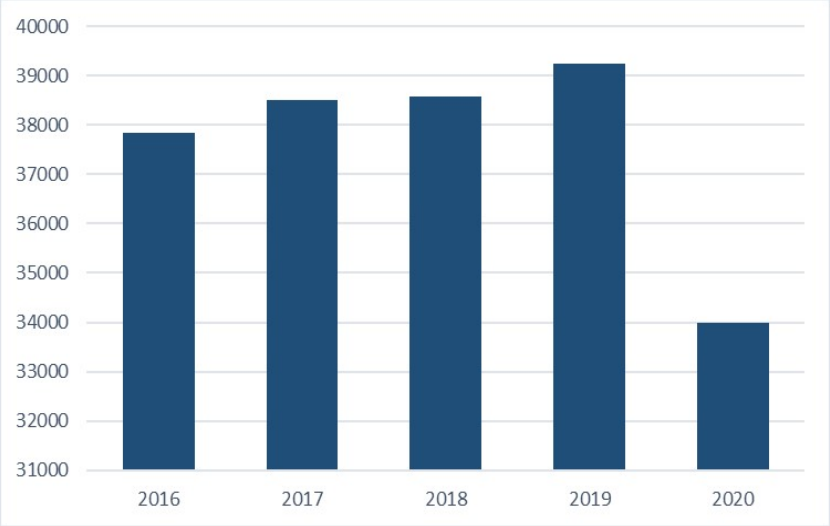


Figura 1: Representación del número total de ingresos anuales en el Hospital Clínico Universitario de Santiago de Compostela entre los años 2016 y 2020.

En las últimas décadas, tal y como se describe en Noonan et al. (2019), se ha producido en los principales países europeos una notable reducción en el número de camas hospitalarias disponibles para su uso. Este hecho, sumado al aumento de la población mundial, provoca que se incremente la presión hospitalaria y, en consecuencia, que la organización óptima en la distribución de las camas sea un desafío trascendental en los complejos hospitalarios en la actualidad.

La búsqueda de un método eficiente de asignación de pacientes a camas del hospital será el com-

ponente principal de este trabajo. Sin embargo, como se verá posteriormente, esta no será una tarea sencilla debido a los muy diversos factores a tener en cuenta. De hecho, históricamente, la organización de un hospital ha sido siempre una de las tareas más complejas dentro del ámbito sanitario, debido al gran número de problemáticas que surgen dentro de uno. Previsión de demanda de pacientes, organización de médicos y enfermeros, distribución de las camas o planificación para los quirófanos son algunas de las innumerables situaciones ante las que se encuentra la dirección de un hospital.

Tradicionalmente, todas las decisiones que se toman en los hospitales (y la asignación de los pacientes a las camas no es una excepción) no se basaban apenas en criterios matemáticos, sino que el componente humano era el predominante en la toma de las mismas. Sin embargo, a partir de la segunda mitad del siglo XX, los grandes avances tanto matemáticos como computacionales provocaron en muchos casos la automatización en la organización hospitalaria.

La implementación de herramientas matemáticas en la gestión diaria de los hospitales es cada vez más común y puede tener un gran impacto en la eficiencia y calidad de los servicios de salud. Algunos ejemplos de cómo se pueden utilizar estas herramientas incluyen la optimización en la asignación de recursos, la predicción de demanda de servicios y la identificación de patrones en grandes conjuntos de datos de pacientes. Además, las herramientas matemáticas también pueden ser útiles en la toma de decisiones clínicas, como en el uso de modelos de predicción para determinar el tratamiento más efectivo para un paciente en particular. En general, el uso de modelos matemáticos en los hospitales puede contribuir a una atención médica más eficiente y de mayor calidad.

En el Hospital Clínico Universitario de Santiago de Compostela, la toma de decisiones en la asignación de camas no ha sido, a lo largo de su trayectoria histórica, muy diferente a la del resto de centros hospitalarios: el componente humano ha sido el predominante teniendo como base fundamental el criterio médico. Sin embargo, la implementación de un modelo matemático eficiente de asignación de pacientes a camas en el hospital no solo liberaría al personal sanitario de esta difícil tarea, sino que evitaría posibles errores humanos que, en un contexto tan crítico como el que tiene lugar en un centro hospitalario, podrían llegar a ser fatales.

Además de poder facilitar el trabajo al personal sanitario y de prevenir errores humanos, la informatización del proceso de asignación reduciría, en muchos casos, el tiempo requerido para ubicar a los pacientes en las camas que más le convengan según su patología. Por otro lado, un buen modelo predictivo de la demanda hospitalaria favorecería la implementación de un modelo que lograse realizar una asignación óptima que tenga en cuenta un posible aumento del número de ingresos en fechas posteriores, debido a diversos factores como un brote de una enfermedad infecciosa. Estas ventajas motivan la creación de estos modelos matemáticos, que serán el objeto de estudio de este trabajo.

Desde un punto de vista matemático, el problema de asignar pacientes a camas (PBA, por sus siglas en inglés) es un problema de optimización que se presenta en el contexto de la atención médica y que consiste en encontrar la asignación óptima de pacientes a camas disponibles en un hospital o centro de atención médica. Este problema puede ser muy complejo debido a la gran cantidad de variables y factores que deben tenerse en cuenta en el momento de asignar a un paciente a una cama. Algunos de estos factores pueden incluir la gravedad de la condición del paciente, la disponibilidad de personal sanitario (médicos, enfermeros, celadores, ...), la disponibilidad de equipo médico especializado y la ubicación de la cama en el hospital.

Debido al gran número de factores que condicionan el modelado matemático para afrontar el PBA, se han diseñado en los últimos años una gran variedad de métodos de resolución de problemas de este estilo. Sin embargo, dentro de los problemas de asignación de pacientes a camas debemos diferenciar dos clases distintas de problemas: *offline* y *online*. Estas dos versiones del PBA se diferencian en el tipo de pacientes a considerar para elaborar el modelo. Por un lado, un PBA offline tan solo busca asignar a las camas disponibles del hospital los pacientes que tengan cita para ingresar en el centro médico el mismo día (pacientes programados). Es decir, el número de pacientes que van a ingresar un cierto día en el hospital es conocido de antemano y, en función de las características de cada paciente, se les asignará a las camas disponibles de manera eficiente. Por otro lado, un PBA online busca una asignación dinámica de pacientes a las camas del hospital, no solo considerando pacientes programados, sino también aquellos que ingresan a través del servicio de urgencias. Este tipo de problemas son mucho

más difíciles de modelar y suelen recurrir al uso de métodos heurísticos para su resolución. En este trabajo emplearemos un enfoque offline pero no solo considerando los pacientes programados, sino también los pacientes procedentes del servicio de urgencias, mediante una predicción de los mismos empleando los datos históricos de ingresos del hospital.

Pese a que el problema matemático para la asignación de pacientes a camas es relativamente reciente, en la última década han sido ideados numerosos enfoques distintos para abordarlo: desde modelos de programación matemática hasta algoritmos heurísticos muy complejos. La primera definición del PBA se recoge en Demeester et al. (2010), así como el primer método para resolver este problema. Esta primera idea consistió en diseñar un algoritmo cuyo objetivo principal era asignar pacientes a los departamentos más adecuados para tratar su patología en la medida en la que fuera posible. Un primer estudio acerca de la complejidad en la resolución del PBA puede consultarse en Vancroonenburg et al. (2011), donde se demuestra que es un problema de optimización combinatoria y que es problema computacionalmente difícil. La idea inicial para afrontar el PBA estaba basada principalmente en modelos de programación entera. Sin embargo, a menudo en problemas online en los que se trabajaba con una gran cantidad de datos, comenzaron a aparecer problemas computacionales provocados por la dificultad en la resolución del PBA. Estos hechos generaron que se buscasen nuevos métodos de resolución del problema, basados frecuentemente en la creación de algoritmos heurísticos. En esta línea surgieron varios métodos heurísticos como un algoritmo de vecinos más cercanos que puede consultarse en Ceschia y Schaerf (2011). Por otro lado, en Mısır et al. (2013) puede verse un análisis de varios métodos heurísticos aplicados sobre varios conjuntos de datos.

Sin embargo, pese a que el uso de métodos heurísticos ha ido progresivamente ganando protagonismo, especialmente para tratar la versión online del PBA, siguen empleándose modelos de programación entera para afrontar el problema. En concreto, en Vancroonenburg et al. (2013) se proponen varios modelos de programación lineal entera que tienen en cuenta no solo las capacidades de las habitaciones sino también la disponibilidad de los quirófanos. Posteriormente, se han empleado diversas técnicas para abordar problemas de gran tamaño como se puede consultar en Range et al. (2014) donde se emplea la técnica de descomposición de Dantzig-Wolfe para resolver un modelo de programación lineal. Más recientemente, también se han propuesto problemas de programación lineal entera multiobjetivo que buscan penalizar violaciones de las restricciones bajo unos ciertos pesos, como se puede ver en Guido y Conforti (2017). Esta última idea será la que se tomará como base para la elaboración de los modelos de este trabajo, en el que se buscará inicialmente un estudio diario de la situación hospitalaria mediante el uso de un modelo multiobjetivo, para posteriormente realizar un estudio a nivel global de la demanda hospitalaria y de cómo actuar en caso de una posible fluctuación elevada en el número de ingresos diarios esperados.

La estructura de este este documento es la que sigue.

El Capítulo 1 consiste en una introducción a la investigación operativa, que permite abordar desde un punto de vista matemático el análisis de los PBA. También se introducen los problemas de programación matemática con sus ramas más destacadas y propiedades. Finalmente, se presenta una breve sección introductoria de los métodos heurísticos.

En el Capítulo 2 se presenta un primer modelo para dar solución al PBA desde un punto de vista de la programación matemática. Además, se presenta un segundo modelo que busca mejorar las propiedades del anterior, pero incurriendo en un aumento en la dimensión del problema. Posteriormente se presenta una comparativa empleando datos reales en escenarios muy diferentes para evaluar el comportamiento de cada uno de los dos modelos.

El Capítulo 3 está dedicado a una nueva rama de la programación matemática: la programación estocástica. Después de una introducción a la misma se diseña un modelo que, teniendo como base los del capítulo anterior, pretende modelar la incertidumbre en la demanda de pacientes en fechas futuras. Tras detallar las componentes de este nuevo modelo se realiza una comparativa entre dos enfoques distintos: uno diario correspondiente a los modelos del capítulo previo y uno más global asociado al modelo estocástico. Además, se introduce un nuevo concepto de solución que pretende recoger las virtudes de los modelos de los dos capítulos y se compara esta nueva solución mediante ejemplos que utilizan datos reales.

En el Capítulo 4 se produce un cambio de enfoque y se introducen varios métodos heurísticos que buscan dar solución al PBA desde un punto de vista más intuitivo para compararlos mediante ejemplos con los métodos de los capítulos anteriores.

Finalmente, en el Capítulo 5 se realiza una conclusión global acerca de los resultados obtenidos a lo largo del trabajo apoyándose en los resultados obtenidos en las comparativas que emplean datos reales.

Además del contenido específico de los capítulos, se presentan tres apéndices que permiten ampliar la materia del trabajo y que se describen a continuación. El Apéndice A contiene una descripción de la base de datos de pacientes empleada en los ejemplos del documento, tanto de una forma global como el estudio particular de cada una de las variables de interés. Por otro lado, en el Apéndice B se realiza una breve descripción de cada uno de los *solvers* empleados para la resolución de los ejemplos. Finalmente, en el Apéndice C se adjunta el código en formato *R* y *AMPL* de los modelos introducidos a lo largo del trabajo.

Capítulo 1

Introducción a la investigación operativa

En este trabajo se introducirán aquellos conceptos específicos de la optimización matemática que permiten desde la modelización del problema PBA hasta la aportación de soluciones analíticas, tanto exactas como aproximadas. Por este motivo, se realizará una revisión de aquellos conceptos que se emplearán a lo largo de esta memoria.

El resto del capítulo se organiza como sigue. La Sección 1.1 realiza un breve recorrido histórico de la investigación operativa. La Sección 1.2 introduce las nociones más generales de la programación matemática como herramienta de la investigación operativa, centrándose en las características de los problemas de este tipo. A continuación, la Sección 1.3 está dedicada a introducir los problemas de programación entera, así como sus propiedades. Finalmente, se realiza una breve presentación de los métodos heurísticos en la Sección 1.4.

1.1. Historia de la investigación operativa

La investigación operativa es una de las principales ramas matemáticas que permiten abordar de forma analítica la toma de decisiones en diferentes contextos. Incluye una amplia variedad de técnicas y, en concreto, en este trabajo aplicaremos este tipo de técnicas a situaciones que surgen dentro del ámbito hospitalario, aunque son múltiples y diversas sus aplicaciones en la vida real.

Durante la Segunda Guerra Mundial, uno de los principales objetivos del bando aliado era asignar recursos extremadamente limitados a diversas operaciones militares de manera eficiente. Bajo este contexto nace la "investigación de operaciones (militares)", disciplina en la cual el matemático y físico estadounidense G. B. Dantzig fue el máximo exponente, por sus trabajos publicados en 1947 y que le convierten en uno de los padres de la optimización. El desarrollo de estas técnicas de "investigación de operaciones militares" es lo que da lugar a lo que hoy se conoce como investigación operativa (IO), área matemática y que tiene como principal objetivo proporcionar herramientas analíticas de alto nivel que permitan ayudar en la toma de decisiones en muchos ámbitos de la sociedad actual, y no sólo en conflictos bélicos como se buscaba en un primer momento. Más concretamente, la investigación operativa es una importante rama de las matemáticas que busca, mediante la construcción de unos ciertos modelos matemáticos que describan de manera fidedigna la realidad, ayudar en la toma de decisiones en un determinado contexto.

Además, en las décadas posteriores a la Segunda Guerra Mundial, la investigación operativa evolucionó de manera significativa, no solo en cuanto al incremento de su aplicabilidad en más contextos, sino también en lo relativo a la cantidad de métodos y técnicas de resolución de problemas. Entre ellos, destaca el Método Símplex, propuesto por Dantzig en el año 1949, que revolucionó la resolución de problemas de programación lineal, de los que hablaremos posteriormente, desde una perspectiva

computacional. Por otro lado, gracias al enorme desarrollo informático de las últimas décadas, la investigación operativa ha dado un gran salto ya que problemas que antes precisaban de un tiempo muy elevado para su resolución (o incluso eran irresolubles) para el ser humano, actualmente se resuelven en tiempos ridículamente pequeños empleando una computadora. Este hecho, sumado a los innumerables métodos y técnicas de investigación operativa para resolver problemas complejos que se han desarrollado en los últimos años, hacen que esta disciplina se haya convertido en un área de estudio indispensable en la gestión empresarial. A su vez, permite modelar y resolver problemas de índole muy diversa, por lo que son es una herramienta ideal para afrontar un problema complejo y que depende de muy diversos factores como es el PBA.

Dentro de la investigación operativa existen una amplia gama de procedimientos de modelado y resolución de problemas. Esto hace que podamos considerar varias ramas dentro de la IO, de entre las que destacan: la programación matemática, la teoría de juegos, la teoría de colas, la teoría de inventarios, la simulación, etc. Cada una de estas técnicas proporciona soluciones a diferentes problemas que se plantean, lo que dota a la investigación operativa de una gran versatilidad en cuanto a su aplicabilidad en situaciones diversas pero que buscan eficiencia en sistemas de decisión multi-agente.

1.2. Introducción a la programación matemática

La *programación matemática* constituye una clase de problemas dentro de la investigación operativa que cuenta con innumerables aplicaciones, pero que a su vez está contenida en una familia mucho más general, la de los problemas de optimización. Un problema de optimización queda determinado por un par (F, c), donde F denota el conjunto de puntos factibles y c es la denominada función de coste a optimizar, esto es, el objetivo del problema consiste en encontrar $x \in F$ tal que, para todo $y \in F$, $c(x) \leq c(y)$.

Centrándonos en el caso de los problemas de programación matemática, el objetivo fundamental en esta clase es que, dada una colección de variables de decisión $\mathbf{x} = (x_1, \ldots, x_n)$, se busca minimizar (o maximizar) una cierta función $f(\mathbf{x})$, conocida como *función objetivo*, sobre el conjunto de posibles soluciones. Este conjunto de soluciones queda determinado por una serie de restricciones de desigualdad, $g_i(\mathbf{x})$, $i = 1, \ldots, m$, y de igualdad, $h_j(\mathbf{x})$, $j = 1, \ldots, l$, que deben satisfacer dichos puntos.

Formalmente, representaremos un problema de programación matemática como:

$$\begin{array}{rl} \text{minimizar (maximizar)} & f(\mathbf{x}) \\ \text{sujeto a} & g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0,\ j = 1, \ldots, l. \end{array} \tag{1.1}$$

Por simplicidad, supondremos a lo largo de todo el capítulo que los problemas con los que se trabajarán son de minimización, pese a que los resultados que se expondrán se pueden generalizar, con unas ligeras modificaciones, para problemas de maximización.

Los valores de las variables de decisión que cumplen todas las restricciones del problema conforman la llamada *región factible* y que denotaremos por F. Por tanto, el objetivo de un problema de programación matemática es encontrar los puntos de la región factible que proporcionan un mejor resultado al evaluarlos en la función objetivo. En base a esto, decimos que un punto $\mathbf{x} \in \mathbb{R}^n$ de un problema de programación matemática es una *solución factible* si es un punto de la región factible, es decir, si $\mathbf{x} \in F$. Además, diremos que $\mathbf{x} \in \mathbb{R}^n$ es un *óptimo global* si es una solución factible que además proporciona el mejor valor posible de la función objetivo, es decir, para un problema de minimización, si $f(\mathbf{x}) \leq f(\mathbf{y})$, $\forall \mathbf{y} \in F$. Un concepto menos restrictivo es el de *óptimo local*, donde se pide que exista un cierto valor $\varepsilon > 0$ verificando que $f(\mathbf{x}) \leq f(\mathbf{y})$, $\forall \mathbf{y} \in F \cap B(\mathbf{x}, \varepsilon)$[1].

La formulación presentada en (1.1) es muy general, ya que sobre ella no se presupone ninguna característica que afecte a la función objetivo, a las restricciones o incluso a las variables de decisión

[1] Por $B(\mathbf{x}, \varepsilon)$ nos referimos a la bola abierta de centro el punto $\mathbf{x}$ y radio ε.

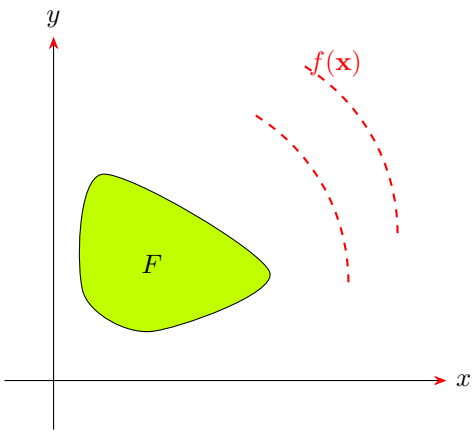


Figura 1.1: Región factible F en dos dimensiones así como dos curvas de nivel de la función objetivo f

del problema. Sin embargo, según el tipo de funciones y de variables que se empleen en el problema se tendrán muy distintas propiedades acerca de la obtención de valores óptimos. En general, para funciones objetivo y restricciones con expresiones complejas la resolución de un problema de programación matemática será también difícil. Por otro lado, veremos a continuación que para algunos tipos de funciones y de variables, la complejidad en su resolución se simplificará notablemente.

Debido a esta variabilidad en las características de las variables y de las funciones que determinan el modelo, dentro de la programación matemática existen diferentes clases de problemas en función de las mismas. Es conocido que las funciones cóncavas y convexas poseen buenas propiedades en lo referente al cálculo de máximos y mínimos, por lo que no parece de extrañar que puedan tener un papel fundamental en problemas de optimización como los que se están tratando. De hecho, una de las ramas más importantes, por las propiedades que poseen los problemas de este tipo, es la programación convexa. Un problema de programación convexa es de la forma (1.1) y en el cual tanto la función objetivo, f, como las restricciones de desigualdad, g_i, son funciones convexas y las restricciones de igualdad, h_j, son funciones lineales.

Este tipo de problemas convexos, pese a provocar una cierta pérdida de generalidad con respecto a los problemas de programación matemática definidos previamente, poseen propiedades teóricas que los hacen realmente interesantes. Antes de comenzar a enunciar estas propiedades, presentaremos dos definiciones básicas que permitirán entender posteriormente la potencia de las propiedades que verifican los problemas de programación convexa.

Definición 1.1 (Combinación convexa) *Dados $\mathbf{x} \in \mathbb{R}^n$ e $\mathbf{y} \in \mathbb{R}^n$, definimos una combinación convexa de estos dos puntos como un punto $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$, con $\lambda \in [0,1]$.*

Definición 1.2 (Conjunto convexo) *Un conjunto F es convexo si dados dos puntos cualesquiera $\mathbf{x} \in F$ e $\mathbf{y} \in F$, cualquier combinación convexa de estos dos puntos está contenida dentro del conjunto F.*

Introducidas estas definiciones, presentamos el primer resultado de interés en problemas de programación convexa.

Lema 1.3 *La región factible F de un problema de programación convexa es un conjunto convexo.*

Esta regularidad de la región factible en problemas de programación convexa favorece la obtención de óptimos debido a las buenas propiedades de los conjuntos convexos para la obtención de puntos extremos. Por otro lado, para problemas en los que existe más de un óptimo global, tenemos una caracterización que nos puede permitir encontrar más puntos óptimos.

Proposición 1.4 *Dado un problema de programación convexa y dos óptimos globales del mismo,* $\mathbf{x}$ *e* $\mathbf{y}$*, toda combinación convexa de estos dos puntos es también un óptimo global del problema.*

Sin embargo, la propiedad más importante que poseen los problemas de programación convexa y que los hace interesantes en cuanto a su aplicabilidad, tiene que ver con la optimalidad local y global de sus soluciones.

Proposición 1.5 *Dado un problema de programación convexa, todo óptimo local del problema es un óptimo global.*

Este resultado simplifica de manera notable la resolución de problemas de esta clase, pues el hecho de encontrar un óptimo local garantiza la optimalidad global del mismo. Por ello, los procedimientos de resolución de problemas de programación convexa se reducen a la búsqueda de un óptimo local del problema. Además, existen algoritmos eficientes para la búsqueda de óptimos locales en problemas de programación matemática, por lo que sería suficiente aplicar uno de estos algoritmos para hallar el óptimo global de un problema de programación convexa. Es por todo ello que la Proposición 1.5, pese a su aparente sencillez, permite valorar la dificultad en la resolución de un problema de programación matemática. Finalmente, presentemos un resultado que caracteriza el conjunto de soluciones de un problema de programación convexa.

Como consecuencia de la Proposición 1.4 y de la definición de conjunto convexo, si el óptimo global es único será trivialmente un conjunto convexo y, si existen varios óptimos, cualquier combinación convexa de ellos también lo será, por lo que el conjunto de óptimos globales resultante será convexo.

Proposición 1.6 *Dado un problema de programación convexa, el conjunto de óptimos globales del problema es un conjunto convexo.*

Si bien, como hemos visto, las funciones convexas presentan muy buenas propiedades de optimalidad, existe un tipo de funciones que, como caso particular de las funciones convexas, dan lugar a unas propiedades todavía mejores en cuanto a la resolución de los problemas: las funciones lineales o afines. Por tanto, un problema como el formulado en 1.1 se dice que es de programación lineal si tanto la función objetivo, f, como las restricciones, g_i y h_j, son funciones lineales.

Como las funciones afines (lineales) son un caso particular de las funciones convexas, todos los resultados enunciados para programación convexa se cumplirán también para programación lineal. Sin embargo, presentamos a continuación algunos de los resultados que surgen específicamente en este contexto. Este considera el concepto de politopo, que es la generalización n-dimensional de un polígono en el plano.

Lema 1.7 *La región factible de un problema de programación lineal es un politopo.*

El hecho de que la región factible de problemas de este tipo sea tan regular hace que surjan una serie de propiedades que caracterizarán las soluciones. En primer lugar, enunciamos un primer resultado, muy importante, que es en esencia la base del método Símplex ideado por Dantzig.

Proposición 1.8 *El conjunto de puntos candidatos a ser óptimos de un problema de programación lineal está compuesto por los puntos extremos del politopo* F*.*

Este resultado, sumado a la regularidad de la región factible, hace que existan algoritmos eficientes para la resolución de problemas de programación lineal. De entre estos algoritmos destaca el *método Símplex*, cuyo procedimiento consiste en moverse por los distintos puntos extremos de la región factible F buscando una mejora progresiva en el valor de la función objetivo. Más allá del ya mencionado método Símplex, existen otros algoritmos populares para la obtención de óptimos en problemas de este estilo, como el algoritmo de Karmarkar.

Si bien los problemas de programación lineal pierden generalidad con respecto a la programación convexa, la sencillez en la resolución de este tipo de problemas hace que a veces sea interesante buscar,

en la medida en que sea posible, una linealización de algunas restricciones (o de la función objetivo) en favor de un aumento en el número de variables o restricciones. Este balance entre el número de variables y restricciones en la posible linealización de un problema es de gran interés, ya que a veces es más eficiente mantener un problema conceptualmente más complejo que buscar linealizarlo aumentando excesivamente el número de variables y restricciones.

Hasta el momento, al definir los problemas de programación convexa y lineal tan solo se han tenido en cuenta las características de la función objetivo y de las restricciones, manteniendo en ambos casos la hipótesis inicial de que las variables de decisión del problema eran reales, es decir, que $\mathbf{x} \in \mathbb{R}^n$. Sin embargo, en muchas situaciones, mantener esta hipótesis no es posible pues, según las características de la situación que se busque modelar, puede que no sea posible trabajar con variables cuyo dominio sea toda la recta real. Esta es la principal motivación de nuevas familias de problemas de programación matemática que quedan caracterizadas por la tipología de variables de decisión con las que se esté tratando.

1.3. Los problemas de programación entera

Tal y como se mencionaba al final de la sección anterior, la consideración de ciertas suposiciones sobre las variables de interés plantea nuevas variantes de los problemas anteriores. Los más relevantes son los conocidos como *problemas de programación entera*, en los que tal y como su nombre indica, se asume que alguna de sus variables de decisión es de tipo entero.

Adquiere interés por su versatilidad y por su aplicabilidad en situaciones tan diversas como la resolución de problemas de rutas de vehículos o de organización de recursos, en los que los bienes pueden no ser fraccionales, y sobre los que es imprescindible introducir en el correspondiente modelo de programación matemática restricciones sobre el carácter entero de sus variables. Muchos de los problemas clásicos de programación matemática responden a esta de definición, entre los que destacamos:

- los problemas de emparejamiento,
- el problema del transporte,
- el problema del viajante,
- el problema de la mochila,
- el problema de asignación.

Esta restricción en el dominio de las variables de decisión permite modelar situaciones complejas a menudo evitando el uso de un elevado número de restricciones. Sin embargo, la gran mayoría de las propiedades teóricas enunciadas anteriormente para la resolución de problemas de programación convexa y lineal dejan de satisfacerse en este nuevo contexto lo que hace que su resolución no sea del todo sencilla. Esta dificultad en la resolución de este tipo de problemas está provocada, en gran medida, por el siguiente resultado, que hace referencia a la naturaleza de la región factible asociada.

Proposición 1.9 *La región factible de un problema de programación entera es un conjunto discreto.*

En un caso sencillo en dos dimensiones como el que se puede ver en la Figura 1.2, la obtención del óptimo es prácticamente inmediata. En este caso, la región factible está compuesta por tan solo ocho puntos: bastaría con evaluar de forma exhaustiva la función objetivo en cada uno de estos puntos y elegir como óptimo el punto que proporcione un mejor valor de dicha función. Sin embargo, a medida que el tamaño del problema aumenta, el hecho de que la región factible de un problema de programación entera sea discreta dejará de ser un beneficio y se convertirá en un grave inconveniente en la obtención de una solución óptima. De hecho, esta pérdida de continuidad en dimensiones elevadas hace que los algoritmos de convergencia clásicos fallen en la búsqueda de óptimos para problemas de este estilo. Además, también se tiene el siguiente resultado negativo, consecuencia directa de la Proposición 1.9.

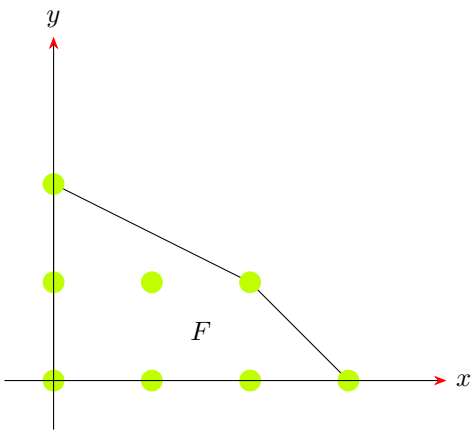


Figura 1.2: Región factible F de un problema de programación entera.

Corolario 1.10 *La región factible de un problema de programación entera no es un conjunto convexo.*

Esta pérdida de la regularidad de la región factible provocada por la falta de continuidad en las variables de decisión hace que las Proposiciones 1.4 y 1.5 dejen, en general, de verificarse, lo que motiva en gran medida la dificultad de resolución de los problemas de programación entera. Para hablar de dificultad en problemas de programación matemática se deben dar las dos siguientes definiciones.

Definición 1.11 (Clase de complejidad NP) *Se dice que un problema pertenece a la clase de complejidad NP si existe un algoritmo que puede verificar cualquier solución a un ejemplo en dicha clase en tiempo polinomial.*

Definición 1.12 (Clase de complejidad NP-hard) *Un problema es NP-hard si cualquier problema en la clase NP se puede reducir a él en tiempo polinomial.*

Desafortunadamente, una gran cantidad de problemas de programación entera se encuadran dentro de la clase de complejidad NP-hard, lo que los hace extremadamente difíciles de resolver.

Sin embargo, y al igual que ocurría para los problemas de programación matemática generales tratados en la sección anterior, existen situaciones particulares, determinadas por las características de la función objetivo y las restricciones, en las que la dificultad de resolución de un problema de programación entera se ve mitigada gracias precisamente a las propiedades de este tipo de funciones. El caso más claro es el de la programación lineal y entera. Por analogía con el caso más general, un problema de programación lineal y entera no es más que un problema de programación entera en el cual tanto la función objetivo como las restricciones son funciones lineales.

El principal inconveniente de esta clase de problemas reside en que se conservan las malas propiedades con respecto a la región factible (sigue siendo discreta y, por tanto, no convexa). Esto hace que algoritmos que se apoyan en la convexidad de la región factible como el método Símplex fracasen en la obtención de óptimos en problemas de este tipo. Sin embargo, las propiedades derivadas por la linealidad de las restricciones favorecen la existencia de técnicas de resolución de estos problemas que se apoyan en la estructura lineal de los mismos. De entre estas técnicas destaca el algoritmo de ramificación y acotación (o *branch and bound*).

Dado un problema de programación lineal y entera, el algoritmo de *branch and bound* se apoya en dos problemas: el original y una versión relajada del mismo obviando la restricción de integrabilidad sobre las variables. La idea de este método es resolver la versión relajada del problema con un método tradicional como el Símplex[2] buscando obtener cotas para el valor de la función objetivo en el óptimo.

[2]Nótese que la versión relajada de un problema de programación lineal entera es un problema de programación lineal, por lo que el método Símplex es adecuado para su resolución.

Una vez obtenida la solución para el problema relajado se selecciona una de las variables no enteras y se generan dos nuevos problemas (uno redondeando el valor de la variable seleccionada hacia arriba y el otro hacia abajo). De esta forma se va formando un árbol con distintas ramas que se van descartando en función de los óptimos obtenidos en cada una de ellas. Esta ramificación y acotación da nombre al algoritmo. Pese a que este método está diseñado para problemas de programación lineal entera, es la base para muchos algoritmos de resolución de problemas de programación entera generales, por lo que su importancia es indiscutible.

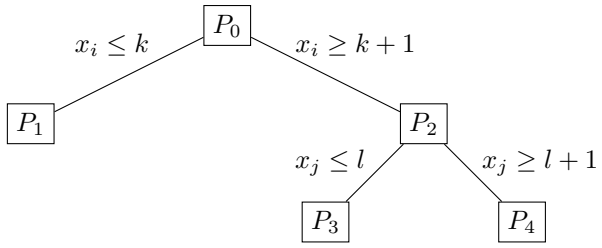


Figura 1.3: Representación del algoritmo de branch and bound.

A modo de resumen, los problemas de programación lineal y entera siguen siendo, en general, difíciles de resolver, pero la existencia de técnicas de resolución como *branch and bound* pueden facilitar enormemente su resolución en algunos casos. Además, el hecho de que se asuman como variables enteras algunas de ellas favorece el modelado de situaciones muy diversas, como las que se tratarán en este trabajo.

1.4. Los métodos heurísticos: una breve introducción

El principal inconveniente que surge con los problemas modelados a través de técnicas de investigación operativa es que, como vimos en el caso de algunos problemas de programación matemática, aportar soluciones no es una tarea sencilla. Pese a la existencia de esta clase de algoritmos de resolución de problemas de programación matemática, existen casos donde la complejidad del problema a tratar hace que sea imposible resolverlo empleando estos algoritmos. En el contexto que nos ocupa, las expresiones de las restricciones o de la función objetivo de un problema de esta clase no acostumbran ser sencillas, lo que hace que su resolución se complique enormemente, incluso a veces pese a la ayuda de un ordenador.

Esta dificultad en la obtención exacta del óptimo del problema motiva la creación de algoritmos generales de búsqueda que permitan proporcionar soluciones en tiempo razonable con la ayuda de las herramientas computacionales. Es por ello que surgen unos métodos llamados *heurísticos*. Estos procedimientos, pese a no asegurar en ocasiones la convergencia a soluciones óptimas (locales o globales), pueden ser muy efectivos en la práctica, ya que a menudo logran una solución "suficientemente buena" para el contexto en el que se necesita.

A lo largo de las últimas décadas se han presentado diferentes definiciones del concepto de método heurístico. Entre ellas, destacamos la presentada en Díaz et al. (2000).

Definición 1.13 (Método heurístico) *Un* método heurístico *es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución.*

Si bien la motivación detrás de la creación de métodos heurísticos ha sido el hecho de su utilidad/aplicabilidad en situaciones en las que los métodos tradicionales emplearían demasiado tiempo en resolver el problema, existen innumerables situaciones donde el uso de herramientas heurísticas es

imprescindible. Por ejemplo, pueden existir condiciones muy difíciles de modelar como restricciones de un problema de programación matemática. En estos casos, es posible que debido a la naturaleza del problema no sea posible encontrar un método exacto de resolución del problema. Sin embargo, el uso de técnicas más intuitivas como los algoritmos heurísticos, permiten abordar el problema y obtener una solución realmente buena y razonable sin tener por qué justificar su optimalidad. Por otro lado, hay veces que es posible combinar el uso de herramientas heurísticas como complemento o paso previo de métodos exactos.

Capítulo 2

Modelo de gestión hospitalaria: un enfoque basado en la optimización matemática

Como avanzábamos en la sección introductoria del trabajo, el problema de asignar pacientes a camas se ha convertido en un gran desafío en entornos de atención médica, ya que a menudo hay una escasez de camas disponibles para atender a los pacientes. Esto puede ser especialmente cierto en situaciones de emergencia o durante brotes de enfermedades infecciosas. Además, una asignación de camas subóptima, podría derivar en un crecimiento notable de las listas de espera provocando una grave pérdida en la calidad de los servicios sanitarios.

Una de las principales ideas que pueden surgir para intentar reducir las listas de espera en un hospital es la incorporación de nuevas camas al sistema. Sin embargo, más allá del claro coste económico que esta construcción supondría, existen más factores por los cuales el incremento en la construcción de camas no supondría un beneficio demasiado relevante en términos de reducción de listas de espera. De hecho, Kroneman y Siegers (2004) realizaron un estudio en 10 países europeos sobre el efecto que supondría un aumento de camas hospitalarias en las tasas de demanda, obteniendo que los países con más camas disponibles para su uso tenían tasas de demanda significativamente más elevadas que el resto. Además, establecieron un valor fijado de 1.44 para la elasticidad del modelo, que no es más que una constante que refleja el crecimiento de la demanda hospitalaria a medida que aumenta el número de camas habilitadas. Más concretamente, una elasticidad superior a 1 significa en este contexto que el crecimiento en la demanda hospitalaria crece incluso más rápido que el número de camas, lo cual haría que las listas de espera no solo no se redujeran, sino que incluso podrían incrementarse.

Teniendo en cuenta dicha constante de elasticidad, Antelo et al. (2015) plantearon un estudio de simulación con datos del Complejo Hospitalario Universitario de Santiago de Compostela (CHUS) en el que llegaron a la conclusión de que, considerando el valor para la elasticidad propuesto en Kroneman y Siegers (2004), ni siquiera una construcción masiva de camas en el hospital ayudaría a disminuir significativamente las listas de espera, e incluso en algunos casos se observaba que estas podrían llegar a aumentar. Por todo ello, no parece que la construcción de nuevas camas resulte necesariamente beneficioso a la hora de reducir las listas de espera, por lo que se debe buscar otro tipo de metodologías a aplicar en este problema. Además, es importante notar que la variación en el comportamiento de la lista de espera en función del número de camas disponibles provoca que no sea posible el uso de la teoría de colas general para estudiar la lista de espera. En este trabajo abordaremos este problema desde una perspectiva diferente, al buscar un método de asignación de los pacientes a las camas del hospital de una manera eficiente en aquellos contextos de alta demanda hospitalaria.

El resto del capítulo se organiza como sigue. La Sección 2.1 propone un primer modelo para resolver el PBA desde un punto de vista de la programación entera. La Sección 2.2 introduce ligeras

modificaciones con el propósito de transformar el problema en uno de programación lineal entera, con el coste de aumentar el número de restricciones del problema. Finalmente, en la Sección 2.3 se realiza una comparativa del comportamiento de ambos modelos bajo tres escenarios distintos empleando diferentes *solvers* y comprobando si la linealización del problema compensa el incremento en el número de restricciones.

2.1. Una primera aproximación al problema de asignación de camas

En esta sección, se propone un primer modelo basado en optimización matemática para abordar el problema de asignar pacientes a camas (PBA). Para la modelización de este problema bajo este enfoque debemos de tener en cuenta la existencia de varios factores a considerar al tratar de asignar pacientes a camas de manera efectiva. Entre ellos, destacamos los siguientes:

- *La gravedad de la condición del paciente.* Cabe plantearse que los métodos de asignación de camas distingan aquellos condicionantes de más gravedad en el diagnóstico de los pacientes, de forma que estos tengan prioridad para obtener una cama.
- *La disponibilidad de camas.* En este contexto es importante tener en cuenta la cantidad de camas disponibles en cada momento y asegurarse de que aquellas que se ocupen lo hagan de la manera más eficiente posible.
- *La ubicación de la cama.* En aras de la eficiencia del sistema de gestión hospitalaria, parece lógico asumir que los pacientes deben ser asignados a camas que estén cerca de los servicios médicos (un laboratorio de diagnóstico o una unidad de cuidados intensivos) que necesitan de acuerdo con su diagnóstico.
- *La duración prevista de la estancia hospitalaria.* En este sentido, es importante también tener en cuenta la duración prevista del tratamiento del paciente al asignarle una cama de acuerdo con su diagnóstico inicial. Una cama ocupada durante un período prolongado de tiempo puede retrasar el tratamiento de otros pacientes.
- *La actividad hospitalaria programada.* Los modelos de gestión hospitalaria más generales deberían considerar tanto la actividad programada con anterioridad, como las intervenciones no urgentes, como el ingreso de pacientes a través de los servicios de urgencias. Así, una gestión eficiente del sistema buscaría evitar la cancelación de aquellas actividades programadas en el hospital. Por ejemplo, se busca evitar, en la medida de lo posible, que un paciente que tenga previsto ingresar un cierto día tenga que posponer su cita.

La base para los modelos de optimización desarrollados en este capítulo está extraída de Guido et al. (2018), donde se desarrolla una metaheurística para abordar el PBA. En esta sección, se planteará un modelo de programación matemática no excesivamente complejo que tenga en cuenta todos los aspectos anteriormente mencionados. A partir de él, buscaremos herramientas que permitan el estudio de la distribución eficiente de un determinado grupo de pacientes en las camas del hospital.

2.1.1. El modelo original

En esta sección plantearemos un primer modelo de optimización que considere los diferentes factores existentes a la hora de realizar una asignación efectiva de pacientes a camas dentro de un hospital. Desde un punto de vista matemático, cada uno de esos criterios puede ser cuantificado de forma que en términos de su optimización tendríamos a priori varias funciones objetivo a optimizar. Esto conllevaría la definición de lo que se conoce como un *problema de optimización multiobjetivo*. La resolución de un problema de este estilo no es en general sencilla. Una de las soluciones más empleadas para enfrentarse

a un problema multiobjetivo es resumir las funciones objetivo en una suma ponderada de las mismas, de manera que sea posible otorgarle más peso a algunas de las funciones objetivo que a otras.

Por tanto, el objetivo del problema de optimización que plantearemos será maximizar una suma ponderada de los diferentes factores de decisión que considerábamos previamente. Más concretamente, en la función objetivo del problema consideraremos el hecho de asignar a los pacientes a áreas del hospital que sean idóneas para el tratamiento de su enfermedad, la gravedad de la condición de cada paciente y el hecho de evitar la cancelación de un paciente programado. Además, debemos considerar una serie de restricciones asociadas a la organización hospitalaria. Por ejemplo, debemos regular circunstancias que regulan el funcionamiento interno de las hospitalizaciones, como que dos personas de sexos opuestos no deberían ocupar una misma habitación, o que el porcentaje total de ocupación del hospital no puede superar un cierto umbral, en previsión de una posible situación de emergencia extrema.

A continuación describiremos los elementos que caracterizan nuestra propuesta de modelo de gestión hospitalaria basada en optimización. Comencemos detallando los distintos conjuntos de datos disponibles y que emplearemos en el modelado del problema.

Conjuntos

En primer lugar, consideramos el conjunto de individuos involucrados en el problema. Entonces,

- P denota al conjunto de pacientes que ingresan en el hospital.

A su vez, dentro del conjunto P, se distinguen los siguientes grupos de pacientes por cuestión de sexo, o del carácter de su ingreso.

- P_f, formado por aquellos pacientes en P de género femenino.
- P_m, formado por aquellos pacientes en P de género masculino.
- P_p denota al conjunto de pacientes programados, es decir, el conjunto de aquellos pacientes que tienen previsto un ingreso hospitalario para un cierto día fijado de antemano.

Sin embargo, además de aquellos relativos a los pacientes, deben considerarse otro tipo de conjuntos:

- C, denota el conjunto de camas disponibles del hospital.
- H, denota el conjunto de habitaciones del hospital con alguna cama libre en ellas.
- H_d, representa el conjunto de aquellas habitaciones con más de una cama.
- D, incluye a todos los departamentos o áreas del hospital.
- N, que considera los diferentes niveles de prioridad asociado a los pacientes. Estos se corresponden con valores que se le asignan a cada paciente en función de su nivel de riesgo y basados en características del mismo, como la gravedad de su condición física o su edad.

Parámetros del modelo

Los parámetros del modelo no son más que unos ciertos valores, prefijados de antemano que servirán como nexo entre las variables de decisión y los conjuntos de datos. Además, también sirven para relacionar conjuntos de datos entre sí, determinar alguna característica de los pacientes y, por tanto, escribir de forma más compacta el problema de optimización.

Los parámetros a considerar son:

$$Ubi_{ch} = \begin{cases} 1, & \text{si la cama } c \in C \text{ está en la habitación } h \in H. \\ 0, & \text{en otro caso.} \end{cases}$$

$$DepHab_{hd} = \begin{cases} 1, & \text{si la habitación } h \in H \text{ está en el departamento } d \in D. \\ 0, & \text{en otro caso.} \end{cases}$$

$$DepOpt_{pd} = \begin{cases} 1, & \text{si el departamento } d \in D \text{ es el óptimo para el paciente } p \in P. \\ 0, & \text{en otro caso.} \end{cases}$$

Estos tres parámetros permiten identificar la ubicación de una cama, de una habitación en un departamento y si un paciente está ubicado en el área correcta dentro del hospital.

- G, representa el número global de camas del hospital.
- O, hace referencia al número de camas del hospital ocupadas en el momento del estudio.
- $RiesgoPac_p$, es un valor numérico que indica el valor de riesgo para cada paciente $p \in P$.
- b_D, es la tasa de beneficio por situar a un paciente en el departamento adecuado.
- b_N, es la tasa de beneficio por ingresar a un paciente con prioridad alta.
- b_P, es la tasa de beneficio por evitar la cancelación de un paciente programado.
- η, es el umbral de saturación máxima permitida en la ocupación hospitalaria. Según la fuente escogida, oscila entre el 75 % y el 95 % y tiene que ver con el porcentaje de camas disponibles para su uso, dejando algunas inhabilitadas en previsión de una posible situación de emergencia.

Variables a considerar en el modelo

Presentemos finalmente las variables de decisión asociadas en nuestro problema. Consideraremos dos tipos distintos. El primer tipo tendrá relación con la cama que se le asigna a cada paciente (si es que se le asigna una) y el segundo tipo se referirá a saber si el paciente ingresa en el departamento que le debería corresponder según su patología.

$$x_{pc} = \begin{cases} 1, & \text{si se le asigna la cama } c \in C \text{ al paciente } p \in P. \\ 0, & \text{en otro caso.} \end{cases}$$

$$y_p = \begin{cases} 1, & \text{si al paciente } p \in P \text{ se le asigna una cama del departamento más apropiado.} \\ 0, & \text{en otro caso.} \end{cases}$$

Formulación del modelo

Una vez introducidos los conjuntos, los parámetros y las variables del modelo, estamos en condiciones de plantear el problema de programación matemática. La base del problema radica, como avanzábamos al inicio de la sección, en maximizar una serie de beneficios obtenidos mediante una buena asignación de los pacientes a las camas.

$$\text{maximizar} \quad \sum_{p\in P} y_p d_D + \sum_{p\in P}\sum_{c\in C} x_{pc} RiesgoPac_p b_N + \sum_{p\in P_p}\sum_{c\in C} x_{pc} b_P \tag{2.1}$$

$$\text{sujeto a} \quad \sum_{c\in C} x_{pc} \leq 1,\ \forall p \in P \tag{2.2}$$

$$\sum_{p\in P} x_{pc} \leq 1,\ \forall c \in C \tag{2.3}$$

$$\frac{1}{G}\left(O + \sum_{p\in P}\sum_{c\in C} x_{pc}\right) \leq \eta \tag{2.4}$$

$$y_p - \sum_{c\in C}\sum_{h\in H}\sum_{d\in D} x_{pc} Ubi_{ch} DepHab_{hd} DepOpt_{pd} = 0,\ \forall p \in P \tag{2.5}$$

$$\sum_{p\in P_p}\sum_{c\in C} x_{pc} \leq |P_p| \tag{2.6}$$

$$\left(\sum_{p\in P_f}\sum_{c\in C} x_{pc} Ubi_{ch}\right) \cdot \left(\sum_{p\in P_m}\sum_{c\in C} x_{pc} Ubi_{ch}\right) = 0,\ \forall h \in H_d \tag{2.7}$$

Por simplicidad, nos referiremos a este problema de optimización como NL-PBA a lo largo del documento. Analicemos brevemente la función objetivo y las restricciones del problema.

- La función objetivo expresada en (2.1) no es más que una suma ponderada de los beneficios obtenidos al cumplirse cada uno de los objetivos asociados a los parámetros b_D, b_N y b_P. El primer sumando recoge el beneficio asociado a asignarle a un paciente el departamento que más le conviene según su condición de salud. El segundo asigna beneficios en función del nivel de riesgo de cada paciente al que se le asigne una cama. Finalmente, el tercer sumando se refiere al beneficio asociado a evitar la cancelación de un paciente programado.
- La restricción en (2.2) asegura que no pueda haber un paciente con más de una cama asignada.
- Por otro lado, la restricción en (2.3) evita que pueda haber una cama con más de un paciente asignado.
- La restricción en (2.4) modela el hecho de que el número total de camas usadas no puede superar el umbral η.
- Para relacionar las variables de decisión (x_{pc} e y_p) empleamos la restricción en (2.5), ya que y_p solo tomará el valor 1 cuando al paciente p se le asigne una cama en una habitación del departamento que es óptimo para tratar su diagnóstico.
- La restricción en (2.6) regula el número total de enfermos programados a los que se le asigna una cama.
- Finalmente, la restricción en (2.7) previene que pueda haber en la misma habitación pacientes de dos géneros distintos.

Estamos entonces ante un problema de programación entera (las variables de decisión toman valores enteros) con $|P|(|C|+1)$ variables binarias y $2|P|+|C|+|H_d|+2$ restricciones. El hecho de que las variables de decisión tan solo puedan tomar valores enteros hace que, como avanzábamos en el capítulo anterior, el problema sea difícil de resolver. Además, si bien ni el número de variables ni el de restricciones es excesivamente alto para un número razonable de pacientes, existe un pequeño problema con esta primera formulación del modelo, y es que mientras que la función objetivo y las restricciones (2.2)-(2.6) son lineales, las $|H_d|$ restricciones que origina la expresión (2.7) no lo son.

2.2. Versión linealizada del modelo

Como hemos visto en el capítulo de preliminares, la linealidad es una propiedad muy interesante a considerar en problemas de programación matemática. Así, buscaremos ahora reformular el problema transformando el bloque de restricciones no lineales por su correspondiente linealización. Aunque intuitivamente, dicha linealización conllevará un incremento en el número de restricciones del problema, la comparativa en la práctica nos servirá para chequear si en este caso compensa el paso de un problema de programación entera general a un problema de programación lineal entera.

El nuevo modelo que se propone es en esencia el de la sección anterior. Tal y como se ha indicado, únicamente se modifica el bloque de restricciones (2.7) por su versión linealizada. El modelo lineal resultante es el siguiente:

$$\text{maximizar} \quad \sum_{p\in P} y_p d_D + \sum_{p\in P}\sum_{c\in C} x_{pc} RiesgoPac_p b_N + \sum_{p\in P_p}\sum_{c\in C} x_{pc} b_P \tag{2.8}$$

$$\text{sujeto a} \quad \sum_{c\in C} x_{pc} \leq 1,\ \forall p \in P \tag{2.9}$$

$$\sum_{p\in P} x_{pc} \leq 1,\ \forall c \in C \tag{2.10}$$

$$\frac{1}{G}\left(O + \sum_{p\in P}\sum_{c\in C} x_{pc}\right) \leq \eta \tag{2.11}$$

$$y_p - \sum_{c\in C}\sum_{h\in H}\sum_{d\in D} x_{pc} Ubi_{ch} DepHab_{hd} DepOpt_{pd} = 0,\ \forall p \in P \tag{2.12}$$

$$\sum_{p\in P_p}\sum_{c\in C} x_{pc} \leq |P_p| \tag{2.13}$$

$$(1 - x_{ic} - x_{jc'}) Ubi_{ch} Ubi_{c'h} \geq 0,\ \forall c \in C, c' \in C, h \in H_d, i \in P_m, j \in P_f \tag{2.14}$$

Para hacer referencia a este nuevo problema de optimización se empleará, a lo largo del trabajo, el término L-PBA. En este nueva formulación del modelo, las $|H_d||C|^2|P_m||P_f|$ restricciones en (2.14) son ahora funciones lineales. De esta forma, hemos transformado el problema de programación entera de la sección anterior en un problema de programación lineal entera para el cual, las técnicas que de *branch and bound* son presentadas como técnica habitual para su resolución.

Sin embargo, si bien el nuevo modelo es puramente lineal en sus restricciones con el mismo número de variables que en el modelo originalmente planteado, el número total de restricciones se ha visto claramente incrementado. De hecho, se han añadido $|H_d|(|C|^2|P_m||P_f|-1)$ restricciones. Esto puede hacer que en escenarios más complejos, con una cantidad grande de pacientes o de camas, el número de restricciones sea excesivamente elevado. La Tabla 2.1 recoge el número de variables y de restricciones de cada una de las dos formulaciones del modelo.

	Variables	**Restricciones**
Modelo original	$\|P\|(\|C\|+1)$	$2\|P\|+\|C\|+\|H_d\|+2$
Modelo linealizado	$\|P\|(\|C\|+1)$	$2\|P\|+\|C\|+\|C\|^2\|P_f\|\|P_m\|\|H_d\|+2$

Tabla 2.1: Número de variables y de restricciones de los dos modelos propuestos en el capítulo.

Además, es conveniente recordar que pese a que los problemas de programación lineal entera tienen mejores propiedades que los problemas de programación entera generales, todavía sigue siendo difícil su resolución en general, pues ya no se verifican propiedades tan importantes en aspectos como la optimalidad (global y local).

2.3. Análisis comparativo: la gestión de las hospitalizaciones

Una vez presentado el marco teórico de ambas formulaciones del modelo de optimización para la gestión hospitalaria, buscaremos analizarlos sobre conjuntos de datos reales para analizar las posibles diferencias en la resolución de cada uno de ellos. Cabe destacar que su aplicabilidad se caracteriza además por la inmediata necesidad de proporcionar los resultados en un contexto tan sensible y urgente como es el de la gestión hospitalaria. Por ese motivo analizaremos comparativamente los tiempos de resolución en ambos casos.

Metodológicamente, los dos modelos presentados en este capítulo permiten abordar el PBA desde el punto de vista de la optimización de forma fidedigna, aunque podría resultar que no fuesen todo lo rápidos que nos gustaría. En esta sección realizaremos un pequeño estudio que nos permitirá evaluar y comparar el rendimiento de cada una de esas formulaciones. Más concretamente, se buscará, sobre una base de datos de pacientes reales, comprobar el comportamiento de ambas versiones del problema de optimización (tiempo de resolución, iteraciones requeridas para su resolución, número de variables y restricciones de cada formulación, etc.) de manera que podremos determinar si realmente existe beneficio al considerar el modelo linealizado pese al gran aumento en el número de restricciones del problema. La base de datos de pacientes a considerar en esta sección contiene a las personas hospitalizadas en el Hospital Clínico Universitario de Santiago de Compostela entre los años 2016 y 2021 y una descripción más detallada de la misma se puede ver en el Apéndice A. Sin embargo, para evitar el ruido provocado por el virus del COVID en los últimos años, para estas comparativas reduciremos el tamaño de la base de datos considerando los pacientes ingresados entre los años 2016 y 2019.

Para evaluar el rendimiento de la propuesta basada en optimización, evaluaremos diferentes escenarios de hospitalización en el año 2016.

- En primer lugar, se considerará un escenario de baja demanda hospitalaria, es decir, con pocos ingresos diarios en una única área.
- A continuación, se abordará la situación opuesta: el caso de una área con un alto número de ingresos diarios.
- Por último, abordaremos el problema completo de la gestión eficiente de las hospitalizaciones en las 32 áreas del hospital compostelano.

2.3.1. Caso I: una situación de baja demanda hospitalaria

Este primer caso aborda la situación surgida en el hospital, concretamente en el área de *Angiología y cirugía vascular*. El área de Angiología y Cirugía Vascular es una especialidad médica que se enfoca

en el diagnóstico y tratamiento de enfermedades que afectan a los vasos sanguíneos, tanto arterias como venas. Se trata de un área donde el número de ingresos diarios no es demasiado alto y que, en concreto, tiene asignadas un total de 20 camas del complejo hospitalario. Teniendo en cuenta su disponibilidad, no se espera una gran afluencia de ingresos diarios en dicha área.

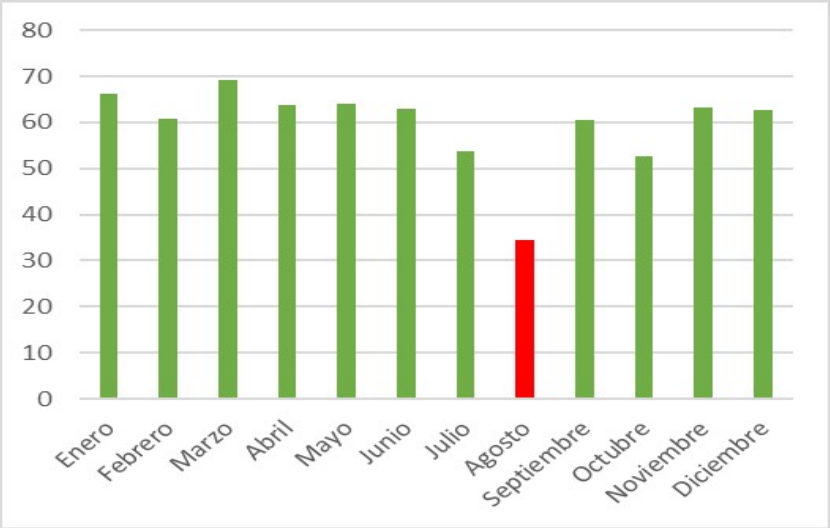


Figura 2.1: Representación del porcentaje de ocupación media del área de *Angiología y cirugía vascular* en cada uno de los meses del año 2016.

Analizando los datos históricos correspondientes al año 2016 (ver Figura 2.1), el porcentaje de ocupación media mensual del área nos permite chequear que se mantiene ligeramente por encima del 60 % a lo largo del año (con picos puntuales rozando el 70 %). Asimismo, podemos observar que en el mes de agosto se produce un descenso significativo en el nivel de ocupación del área, hecho que suele ser un patrón común en el conjunto del hospital. Como en esta primera comparativa entre los modelos buscamos un contexto en el que el número de ingresos diarios sea bajo, nos centraremos en este mes. Dado que la gestión de altas e ingresos diarios puede hacer variar el grado de ocupación, procedemos ahora a ver el número de ingresos diarios en el área el mes de agosto.

Como era previsible, el número de ingresos diarios en este mes es bajo y, de hecho, a la vista de la Figura 2.2, podemos observar que hay bastantes días a lo largo del mes en los que no se produce ningún ingreso en esta área y varios en los que tan solo se produce uno. Como estos casos tampoco tendrían demasiado interés a la hora de comparar los dos modelos, seleccionaremos el día 9 de agosto, en el que se producen cuatro ingresos. La idea será entonces tomar los datos de estos cuatro pacientes que deberían ingresar en el área y ver si en efecto las dos formulaciones del problema obtienen el mismo valor objetivo (consiguen asignar los pacientes a las camas de una manera óptima) y comprobar si hay diferencias entre los tiempos de ejecución de los dos problemas de optimización.

Para poder trabajar con los datos reales se le asignará a cada paciente un nivel de riesgo en una escala del 1 al 10, en función de su diagnóstico principal, de su edad y de su duración del ingreso esperada, y donde 1 representará el riesgo más bajo y 10 el más elevado. Además, se asignará un valor de 75 al beneficio por asignar a un paciente al departamento que le corresponde (b_D), un valor de 10 al beneficio asociado al nivel de riesgo del paciente (b_N) y un valor de 30 al beneficio asociado a evitar la cancelación de un paciente programado (b_P). Además, se considerará un umbral de ocupación máxima permitida en el hospital del 85 %. Para estas situaciones en las que se busca comparar los dos modelos empleando datos de pacientes de un solo área, consideraremos como camas disponibles las que están libres en el área en el momento del estudio, así como un exceso de 20 camas correspondientes a otras áreas a las que se les podría asignar un paciente en el caso de que se completaran las camas asociadas al área de estudio. Como en nuestro caso el área de *Angiología y cirugía vascular* tiene niveles de

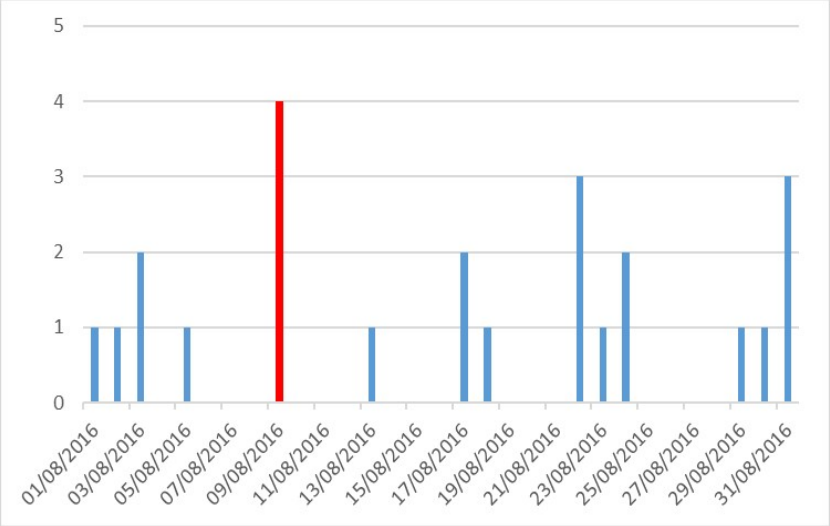


Figura 2.2: Representación del número de ingresos diarios del área de *Angiología y cirugía vascular* en el mes de agosto del año 2016.

ocupación relativamente bajos, tal y como vimos en la Figura 2.1, no se prevé que sea necesario el uso de esas camas supletorias.

La resolución de cada instancia del problema se hará mediante su implementación en lenguaje AMPL y cuyo código se puede ver en el Apéndice C. A modo de comparativa, se emplearán varios *solvers* existentes para resolver ambos problemas. Tres de ellos están diseñados para resolver problemas de programación entera con restricciones no lineales: *BARON*, *Couenne* y *OCTERACT*. También se usará un cuarto *solver* (*Gurobi*) que si bien su versión general está especializada en la resolución de problemas lineales, tiene una versión cuadrática que permite resolver problemas como los resultantes al emplear el modelo no lineal. Una breve descripción de cada *solver* se puede ver en el Apéndice B.

Para comparar el uso de las dos formulaciones del problema, recogeremos en varias tablas, cada una de ellas asociada a un *solver* diferente, el valor óptimo de la función objetivo obtenido por cada una de los modelos, así como el tiempo medio de resolución, medido en segundos, calculado promediando 100 valores obtenidos aplicando el *solver* con distintos iterantes iniciales y el número de variables y restricciones de cada una de las formulaciones. También se expondrán el número medio de iteraciones del método símplex empleadas en la resolución del problema así como el número mínimo y máximo de iteraciones requeridas por cada *solver* a lo largo de las cien resoluciones del problema. Estos datos se recogen en la Tabla 2.2.

Con respecto a los datos obtenidos es necesario hacer dos aclaraciones. En primer lugar, cada uno de los cuatro *solvers* empleados y en cada una de las cien instancias consideradas para cada uno de ellos, se logra obtener la solución óptima del problema, por lo que tan solo se presentará un único valor para la función objetivo, que estará asociado a la mencionada solución óptima. Por otro lado, es necesario mencionar que tanto BARON como OCTERACT tienen implementado un *presolve* que ayuda a resolver de manera mucho más directa problemas de tamaño no excesivamente elevado, como es el caso. Es por ello que el número de iteraciones necesarias por ambos *solvers* para la resolución de este problema no debe tenerse en cuenta[1], pues es el propio *presolve* el que es capaz de obtener el valor óptimo de la función objetivo.

En vista de los resultados recogidos en las tablas, observamos que ambas formulaciones consiguen llegar al valor óptimo de la función objetivo de manera muy rápida (ambas en menos de un segundo), pese a que ya se aprecia un cierto beneficio de usar el modelo no linealizado tanto en los tiempos medios

[1]Estos valores para el número de iteraciones aparecen señalizados en la tabla con el símbolo *.

Gurobi	NL-PBA	L-PBA
F.obj.	680	680
Tiempo	0.037	0.076
Var.	140	140
Rest.	51	24320
Iter.	18.05	21.2
Iter. mín.	18	21
Iter. máx.	19	22

BARON	NL-PBA	L-PBA
F. obj.	680	680
Tiempo	0.030	0.031
Var.	140	140
Rest.	51	24320
Iter.	0*	0*
Iter. mín.	0*	0*
Iter. máx.	0*	0*

Couenne	NL-PBA	L-PBA
F.obj.	680	680
Tiempo	0.110	0.86
Var.	140	140
Rest.	51	24320
Iter.	37.21	62.40
Iter. mín.	36	58
Iter. máx.	39	69

OCTERACT	NL-PBA	L-PBA
F. obj.	680	680
Tiempo	0.13	0.15
Var.	140	140
Rest.	51	24320
Iter.	0*	0*
Iter. mín.	0*	0*
Iter. máx.	0*	0*

Tabla 2.2: Resumen de los valores característicos en la resolución óptima del problema con los parámetros $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 0.85$. Valores óptimos de la función objetivo alcanzado empleando cada una de las formulaciones del problema y cada uno de los *solvers*, así como el tiempo medio de resolución, el número de variables y de restricciones de los dos modelos y el número medio de iteraciones del método símplex empleadas. Número de iteraciones asociadas a los casos más y menos favorables.

de resolución, que son ligeramente inferiores, como en los números de iteraciones del método símplex necesarias para la resolución del problema, que también son ligeramente inferiores a las del modelo linealizado tanto para Gurobi como para Couenne, para el que la diferencia es algo más significativa. Para explicar estas ligeras diferencias basta fijarse en la fila de las restricciones donde, incluso en un problema tan pequeño como el que nos ocupa (tan solo ingresan cuatro pacientes) la diferencia en el número de restricciones es demasiado grande.

2.3.2. Caso II: una situación de alta demanda hospitalaria

Como caso inicial tratamos un área del hospital en la que el número de ingresos diarios era realmente bajo. En esta sección consideraremos la situación opuesta, es decir, nos centraremos en otra área distinta del hospital en la que la afluencia de pacientes sea mayor. Concretamente, consideraremos el área de *Cirugía general y digestiva*, que tiene asignadas 130 camas, frente a las 20 que tenía asignadas el área del ejemplo anterior. Este aumento en el número de camas hace prever un incremento en el número de ingresos diarios. Veamos, en primer lugar, el nivel de ocupación de esta área a lo largo del año 2016.

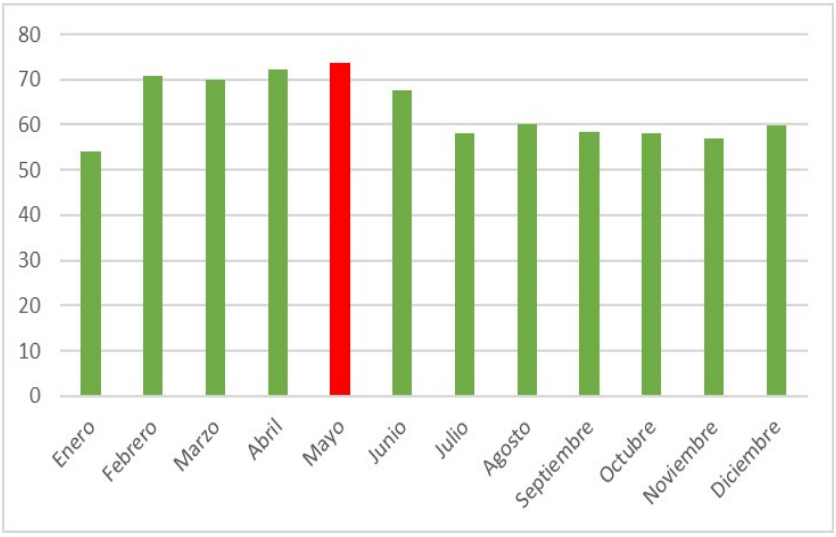


Figura 2.3: Representación del porcentaje de ocupación media del área de *Cirugía general y digestiva* en cada uno de los meses del año 2016.

A la vista de la Figura 2.3 observamos que el mes en el que el nivel de ocupación del área de *Cirugía general y digestiva* es más alta es mayo, por lo que para esta situación en la que buscamos un número de ingresos diarios relativamente alto nos centraremos en este mes. Veamos ahora cómo se distribuyen los ingresos a lo largo de este mes. Dentro de este mes de mayo, nos quedaremos con el día 3, que, como se puede apreciar en la gráfica de la Figura 2.4, es el que cuenta con mayor número de ingresos, con un total de 23 pacientes, de los cuales 9 son mujeres y 14 hombres. Se buscará ahora repetir el procedimiento del ejemplo anterior con los mismos valores para los parámetros ($b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 0.85$). La idea es ver si ahora que aumenta considerablemente el número de ingresos (pese a no ser todavía una cifra elevada), el modelo lineal se ve resentido por el gran número de restricciones que necesita, provocando una diferencia significativa en el tiempo computacional de resolución empleado. Nuevamente, se emplearán los cuatro *solvers* usados en la sección anterior y se resolverá el problema considerando cien instantes iniciales distintos y promediando las soluciones obtenidas para lograr una visión más global del comportamiento de los dos modelos. Recogemos los datos de este nuevo ejemplo en la Tabla 2.3. De nuevo, para todos los *solvers* y todas las instancias del problema se alcanza el valor óptimo de la función objetivo. Además, nuevamente hay que tener en cuenta que algunos números de iteraciones anormalmente bajos están asociados al uso del *presolve* en los *solvers* BARON y OCTERACT. En este caso y a la vista de las tablas se observan diferencias bastante más significativas que en el ejemplo anterior. De hecho, mientras que el número de iteraciones del método símplex no difiere demasiado entre ambas formulaciones (incluso el L-PBA necesita menos para algunos *solvers*), los tiempos medios de ejecución ya sí que son considerablemente distintos. Este hecho es realmente significativo, pues se espera que al realizar un estudio completo de lo que ocurre en un hospital se esperen bastantes más de 23 pacientes diarios ingresados, por lo que el tiempo de resolución asociado al modelo linealizado se podría ver gravemente incrementado. Este aumento en el

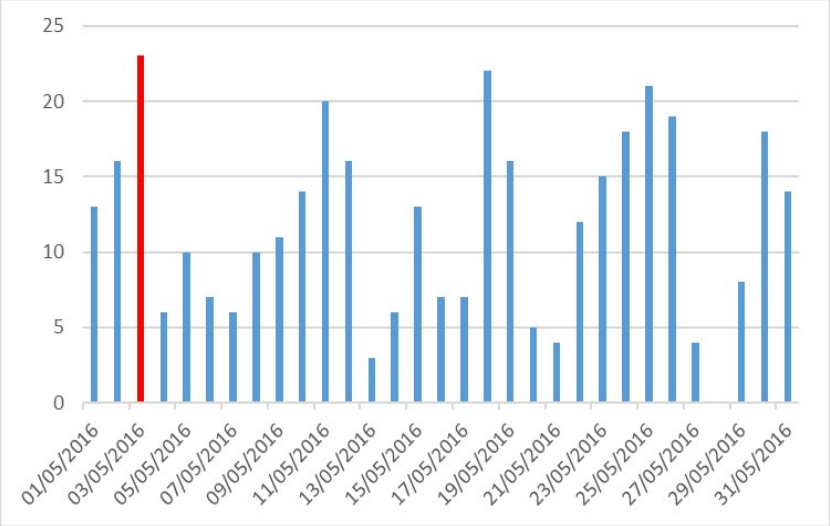


Figura 2.4: Representación del número de ingresos diarios del área de *Cirugía general y digestiva* en el mes de agosto del año 2016.

tiempo de resolución lo podemos nuevamente relacionar con el incremento en el número de restricciones ya que, para un problema relativamente pequeño como el que estamos considerando, son necesarias más de tres millones de restricciones. Esto hace que el tiempo empleado por el *solver* en el proceso de lectura del L-PBA sea muy superior al del NL-PBA. Todo ello hace prever que a la hora de resolver la situación diaria global de un hospital sea más adecuado emplear el modelo no lineal frente al linealizado. Esto se estudiará en un tercer y último caso.

Gurobi	NL-PBA	L-PBA
F.obj.	4005	4005
Tiempo	0.156	24.86
Var.	1081	1081
Rest.	107	3466102
Iter.	412.4	391.8
Iter. mín.	405	390
Iter. máx.	422	394

BARON	NL-PBA	L-PBA
F. obj.	4005	4005
Tiempo	0.56	5.60
Var.	1081	1081
Rest.	107	3466102
Iter.	0*	1*
Iter. mín.	0*	1*
Iter. máx.	0*	1*

2.3.3. Caso III: la gestión global de los ingresos en el hospital

Para evaluar la efectividad y la eficiencia de los modelos de optimización presentados en esta sección se han realizado comparaciones empleando datos asociados a una sola área del hospital. Sin

Couenne	NL-PBA	L-PBA
F.obj.	4005	4005
Tiempo	1.55	175.42
Var.	1081	1081
Rest.	107	3466102
Iter.	231.89	172.75
Iter. mín.	211	170
Iter. máx.	251	179

OCTERACT	NL-PBA	L-PBA
F. obj.	4005	4005
Tiempo	1.24	4.60
Var.	1081	1081
Rest.	107	3466102
Iter.	282.49	315.6
Iter. mín.	277	314
Iter. máx.	284	316

Tabla 2.3: Resumen de los valores característicos en la resolución óptima del problema con los parámetros $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 0.85$. Valores óptimos de la función objetivo alcanzado empleando cada una de las formulaciones del problema y cada uno de los *solvers*, así como el tiempo medio de resolución, el número de variables y de restricciones de los dos modelos y el número medio de iteraciones del método símplex empleadas. Número de iteraciones asociadas a los casos más y menos favorables.

embargo, en esta sección se ampliará el enfoque buscando aplicar ambos modelos a una situación global que considere todas las 32 áreas en las que está dividido el hospital. Es básico realizar esta comparativa global, pues cada área del hospital tiene unas propiedades muy diversas: el número de camas disponibles en cada una de ellas, el porcentaje medio de ocupación del área, la gravedad de los pacientes que ingresan en ella, etc. Por todo ello, parece necesario evaluar la capacidad de cada una de las dos versiones del problema de optimización de asignar camas a los pacientes de una manera óptima en el conjunto global del hospital.

Por tanto, en esta última casuística dejaremos de centrarnos en una área concreta para tratar la situación diaria del hospital desde una visión más global de su demanda de ingresos hospitalarios. Para ello, se utilizará un conjunto de datos mucho más amplio y diverso que en los dos apartados anteriores. El principal problema que surge en esta situación reside en que no podemos considerar al hospital como un conjunto de áreas independientes entre sí, puesto que muchas de ellas están en gran medida relacionadas (material médico similar, personal con formación parecida,...), por lo que parece lógico considerar que si un paciente no puede ingresar en el área que es óptima para tratar su patología, pueda hacerlo en un área con características relativamente similares para que la calidad de la atención médica recibida sea lo más alta posible. Para resolver este inconveniente, se empleará la organización por Institutos de Gestión Clínica con los Servicios Médicos asociados a cada instituto del Hospital Clínic de Barcelona. Esta estructura organizativa nos permitirá realizar una agrupación por proximidad que identificará áreas hospitalarias con características similares donde los pacientes puedan recibir una atención médica de calidad.

Centrándonos ya en la comparativa de cada uno de los dos modelos, analizaremos en primer lugar la ocupación media mensual del Hospital Clínico Universitario de Santiago de Compostela durante el año 2016.

A la vista de la Figura 2.5, se observan unos valores bastante oscilantes de la ocupación global del hospital: desde valores cercanos al 60 % en meses de verano como agosto, a valores superiores al 80 % en los primeros meses del año como febrero, marzo o abril. Para realizar la comparativa entre los

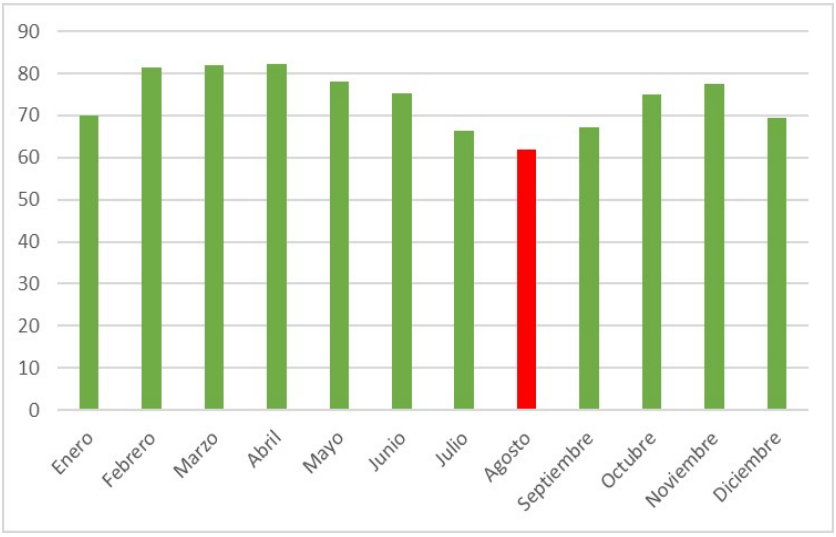


Figura 2.5: Representación del porcentaje de ocupación media del Hospital Clínico Universitario de Santiago de Compostela en cada uno de los meses del año 2016.

modelos, nos centraremos en el mes que menos porcentaje de ocupación hospitalaria presenta: agosto. Dentro de este mes se buscará un día con un número de ingresos intermedio, para el que se espera un número demasiado alto de restricciones asociadas al L-PBA.

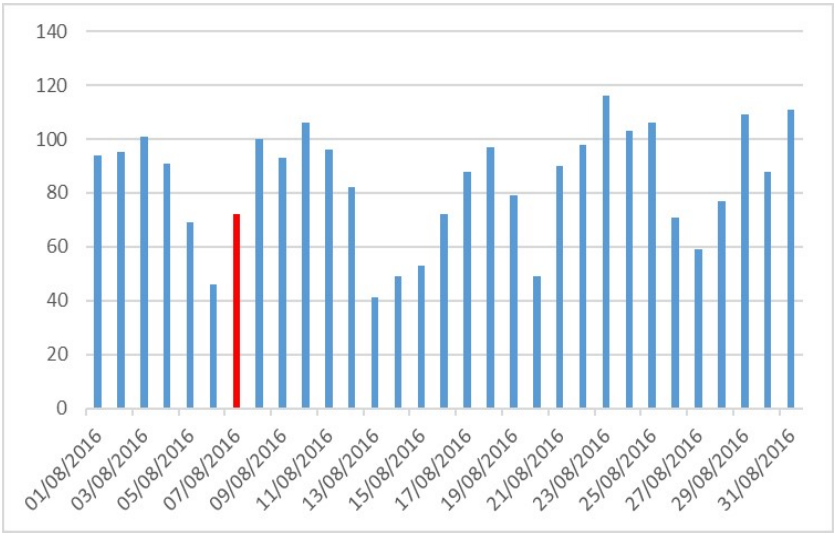


Figura 2.6: Representación del número de ingresos diarios en el Hospital Clínico Universitario de Santiago de Compostela en el mes de agosto del año 2016.

A la vista de la Figura 2.6, se escogerá el día 7 de agosto que, con 72 pacientes ingresados, presenta un número intermedio de hospitalizaciones con respecto a la tendencia del mes. Veremos a continuación que, si bien el número de ingresos no es excesivamente alto, es suficiente para que sea necesario decantarse por el NL-PBA frente a la versión linealizada del mismo.

Para realizar la comparativa de los dos modelos en ese día, se considerarán los mismos valores para los parámetros ($(b_D = 75,\ b_N = 10,\ b_P = 30$ y $\eta = 0.85)$) que en los dos ejemplos anteriores. En

este caso se promediarán, para cada uno de los modelos, los resultados procedentes de 10 realizaciones con distintos puntos de inicio del algoritmo y se recogerán los datos en la Tabla 2.4. En este caso, se empleará el *solver* Gurobi a modo ilustrativo, pues las conclusiones extraídas en este caso son independientes del *solver* considerado.

	NL-PBA	**L-PBA**
Valor de la función objetivo	9335	————
Tiempo medio de resolución	14.22	————
Variables	32760	32760
Restricciones	813	55317412680
Iteraciones promedio	143.9	————
Iteraciones mínimas	143	————
Iteraciones máximas	144	————

Tabla 2.4: Resumen de los valores característicos en la resolución óptima del problema con los parámetros $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 0.85$. Valor óptimo de la función objetivo alcanzado empleando cada una de las formulaciones del problema, tiempo medio de resolución, el número de variables y de restricciones de los dos modelos y el número medio de iteraciones del método símplex empleadas.

En este caso, los resultados son bastante más significativos que en los dos anteriores, ya que el L-PBA no logra obtener la solución óptima del problema en una hora de tiempo límite. De hecho, ni siquiera consigue devolver una solución factible, dado que el *solver* emplea todo el tiempo en la lectura de datos. Esto se debe, al igual que ocurría en las dos situaciones anteriores, al elevadísimo número de restricciones (más de 50.000 millones). Sin embargo, el NL-PBA proporciona la solución óptima en un tiempo relativamente bajo.

Teniendo en cuenta el contexto hospitalario en el que nacen estos modelos, podemos afirmar que el uso del L-PBA es inviable en la práctica, debido a su elevado tiempo de resolución. Como en esta sección estamos analizando la situación hospitalaria diaria, se necesita un modelo que logre obtener una buena solución (idealmente la óptima) en el menor tiempo posible, pues no tendría sentido lograr el valor óptimo cuando ya ha transcurrido gran parte del día, como podría ocurrir con el L-PBA. Por todo ello, concluimos que en este problema concreto de asignación de camas a pacientes del hospital, la linealización del problema general es ineficaz, debido al gran incremento que se produce en el número de restricciones, especialmente para problemas donde tanto el número de pacientes como el de camas disponibles para su uso es elevado. En consecuencia, y a modo de conclusión de esta sección, se elegirá el NL-PBA como método de abordar el problema de asignar pacientes a camas desde el punto de vista de la programación matemática.

Capítulo 3

Planificación en la gestión hospitalaria: un enfoque estocástico

A lo largo del capítulo anterior se ha buscado solucionar el problema de asignación diaria de pacientes a camas del hospital, sin tener en cuenta posibles tendencias en periodos específicos del año. Bajo este esquema, se considerarían las posibles variaciones de demanda hospitalarias provocadas por brotes de alguna enfermedad contagiosa como la gripe a lo largo del invierno, por periodos vacacionales, o por situaciones de alarma como ocurrió hace unos años con el virus del COVID-19. Para poder abordar este tipo de situaciones puede tener sentido considerar una visión más global del problema, introduciendo esa incertidumbre provocada por la tendencia en el modelo original de este capítulo.

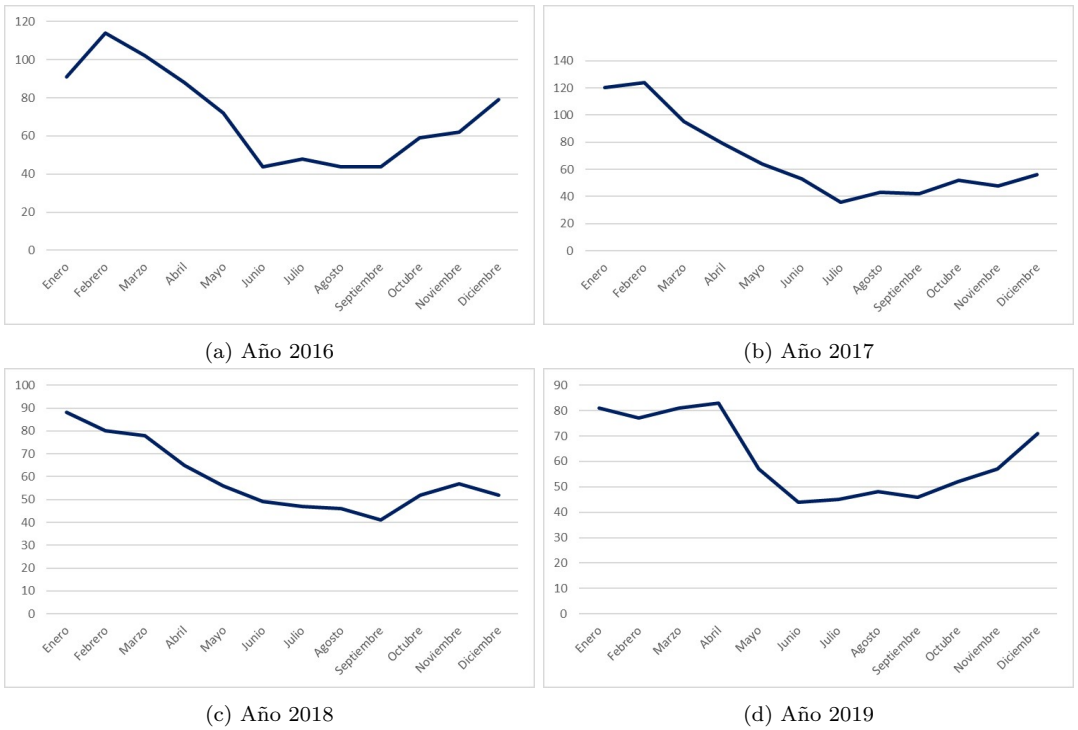
La idea detrás de considerar esta percepción más global del problema es construir un modelo más flexible que no contemple tan solo lo que va a ocurrir un día en concreto, sino que, en base a unas ciertas predicciones de demanda hospitalaria, pueda asignar pacientes a las camas del hospital teniendo en cuenta la posibilidad de que la demanda hospitalaria varíe significativamente. Veamos esto con un ejemplo real.

A menudo, asociamos los meses de invierno con periodos de enfermedades respiratorias. Este incremento en los meses fríos de estas patologías se debe a numerosos factores que hacen que la gran mayoría de los virus respiratorios estacionales presenten picos de contagio en estas épocas del año. De entre estos factores destacan las bajas temperaturas así como el aumento en estos meses de la propensión a estar en espacios cerrados, con escasa ventilación. Todo este tipo de enfermedades, tanto las contagiosas como las que no lo son, pueden provocar ingresos en el área de *Neumología* del hospital, por lo que se prevé que haya diferencias significativas entre las capacidades hospitalarias de esta área en los meses de invierno con respecto a meses más calurosos. En la Figura 3.1 se pueden ver las tendencias en la ocupación anual entre 2016 y 2019 (para evitar la situación excepcional provocada por el coronavirus).

Como se preveía, existe una diferencia significativa en todos los años entre la ocupación de los primeros meses del año (enero, febrero y marzo) y la de los demás meses. En particular, la diferencia es incluso más destacada si consideramos el claro descenso de ocupación en los meses de verano. De hecho, la ocupación del área en los primeros meses de cada año es realmente elevada, llegando a superar en algún caso el 100 % de ocupación[1].

A la vista de las gráficas anteriores parece claro que, al menos a nivel de área, existe una evidente tendencia en lo referente al porcentaje de ocupación de la misma a lo largo del año. Motivado por este hecho, parece tener sentido poder aprovechar esta información sobre la fluctuación en el nivel de ocupación hospitalaria para elaborar un modelo más complejo que permita obtener soluciones para el

[1] Entendemos que se supera el 100 % de ocupación del área hospitalaria si no solo se ocupan la totalidad de las camas asociadas a esta área del hospital sino que es necesario ingresar a pacientes en otras áreas que no serían las óptimas para tratar su patología.



(a) Año 2016 (b) Año 2017

(c) Año 2018 (d) Año 2019

Figura 3.1: Evolución en el porcentaje de ocupación del área de Neumología

PBA que puedan anticipar grandes cambios en la demanda hospitalaria de días o semanas posteriores al estudio.

Para poder introducir esta variabilidad de la demanda hospitalaria en el modelo de programación matemática de este capítulo, emplearemos una nueva rama de la programación matemática, conocida como *programación estocástica*. La ventaja principal en el uso de este tipo de programación reside en la capacidad de modelar la incertidumbre en problemas en los que alguno de los parámetros o de los datos son desconocidos.

El resto del capítulo se organiza como sigue. La Sección 3.1 recoge una pequeña introducción a la programación estocástica, con una definición formal de lo que se entiende por un problema de esta rama, así como sus características comparadas con las de un problema de programación matemática general, para finalmente motivar la importancia de esta rama dentro del contexto del PBA. La Sección 3.2 presenta un modelo de programación estocástica basado en los modelos del capítulo anterior así como un ejemplo de aplicación a un conjunto de datos reales. Finalmente, en la Sección 3.3 se presenta una nueva idea de solución que busca reunir mitigar tanto las desventajas de los modelos del segundo capítulo como las de los modelos de programación estocástica.

3.1. Introducción a la programación estocástica

En la actualidad, un gran número de problemas de toma de decisiones, y en particular, muchos de los problemas de la investigación operativa, involucran incertidumbre en algunos de los parámetros o en los datos que emplean. La programación matemática general, que emplea enfoques determinísticos, a menudo no logra modelar de manera exitosa la incertidumbre inherente en estos problemas. Es por ello que, en dichos casos, la programación estocástica se presenta como una herramienta útil y valiosa para modelar y resolver problemas de optimización bajo una cierta incertidumbre.

La *programación estocástica* es una rama de la IO que se centra en la optimización de problemas

en presencia de incertidumbre. La principal diferencia con respecto a los modelos determinísticos de programación matemática habituales, que suponen un conocimiento total de los parámetros del problema, reside en el hecho de que los modelos estocásticos incorporan la incertidumbre asociada a la variabilidad de alguno de dichos parámetros.

Más concretamente, mientras que los modelos generales de programación matemática asumen que los parámetros del problema son fijos y conocidos, la programación estocástica utiliza variables aleatorias y distribuciones de probabilidad para modelar la incertidumbre en los parámetros. Además, los modelos de programación estocástica permiten considerar una amplia gama de escenarios en función de la variabilidad de los parámetros, así como la posibilidad de evaluar el impacto de la incertidumbre en la obtención de las soluciones óptimas. Sin embargo, si bien este tipo de modelos permiten dar solución a problemas como el PBA desde un punto de vista más global, la dificultad en su resolución es mayor, pudiendo provocar problemas computacionales.

Expongamos ahora la formulación de un problema de programación estocástica para analizar las diferencias y semejanzas con respecto a un problema determinístico. Para esta formulación se ha tomado como referencia la notación empleada en Birge y Louveaux (2011).

Un *problema de programación estocástica* admite la siguiente formulación:

$$\begin{aligned} \text{minimizar} \quad & f(\mathbf{x}, \xi) = \mathbb{E}_{\xi}[Q(\mathbf{x}, \xi)] \\ \text{sujeto a} \quad & R(\mathbf{x}, \xi) \leq 0, \end{aligned} \tag{3.1}$$

donde, al igual que ocurría en los problemas de la forma (1.1), $\mathbf{x}$ denota el conjunto de variables de decisión del problema y f a la función objetivo a minimizar. Sin embargo, en esta nueva formulación aparecen dos nuevos términos que facilitan el modelado de la incertidumbre. El primero de ellos es ξ, que hace referencia a los parámetros aleatorios del problema y que está presente tanto en la función objetivo del problema como en sus diferentes restricciones. Estos parámetros, si bien no son controlables, suponemos que tienen distribuciones de probabilidad conocidas, lo que facilitará en gran medida el trabajo con ellos. El otro término diferente que observamos en la formulación (3.1) tiene que ver con la función objetivo del problema, y es que la función a minimizar (o maximizar) en este tipo de problemas es el valor esperado de una función $Q(x, \xi)$, que involucra tanto a las variables de decisión como a los parámetros aleatorios. Finalmente, el conjunto de funciones $R(x, \xi)$ denota a las restricciones del problema. Si bien por norma general se suelen considerar como restricciones de desigualdad, también pueden convertirse en restricciones de igualdad con el uso de unas ciertas variables auxiliares conocidas con el nombre de variables de holgura.

Presentemos ahora en la Tabla 3.1 una comparativa entre la función objetivo y las restricciones de un problema de programación matemática determinístico y un problema de programación estocástica para poder analizar las ventajas e inconvenientes del uso de cada uno.

	Función objetivo	**Restricciones**
Modelo determinístico	$f(x)$	$g_i(x) \leq 0,\ i = 1, \ldots, m$ $h_j(x) \leq 0,\ j = 1, \ldots, l$
Modelo estocástico	$f(x, \xi) = \mathbb{E}_{\xi}[Q(x, \xi)]$	$R(x, \xi) \leq 0$

Tabla 3.1: Función objetivo y restricciones de un problema de programación matemática determinístico y de uno estocástico.

Ventajas de la programación estocástica

Existen varios factores clave que permiten motivar el uso de modelos de programación estocástica frente a modelos determinísticos generales. De entre ellos, destacamos los siguientes:

- *Mitigación de situaciones conflictivas.* El hecho de tener en cuenta la incertidumbre sobre los parámetros del problema permite a este tipo de modelos anticipar posibles situaciones críticas y poder solventarlas con éxito anticipándose incluso antes de que ocurran. En el contexto hospitalario en el que se está trabajando en este documento, disponer de sistemas predictivos fiables de la demanda hospitalaria en tiempos futuros puede hacer que interese asignar de una manera distinta pacientes a camas de manera que se reserven camas en previsión de un drástico aumento de pacientes con patologías con condición grave.

- *Robustez de las soluciones.* Nuevamente, incorporar la incertidumbre al modelo facilita que, incluso en los casos en los que los parámetros tomen valores atípicos con respecto a su función de distribución, las soluciones obtenidas sean en general preferibles a las obtenidas mediante procedimientos puramente determinísticos. Esta ausencia de sensibilidad ante variaciones de los datos permite que las soluciones puedan adaptarse a escenarios muy diversos. En resumen, las soluciones determinísticas pueden funcionar muy bien para escenarios muy específicos y, por el contrario, las estocásticas se comportarán mejor en un rango de valores de los parámetros aleatorios mucho más amplio.

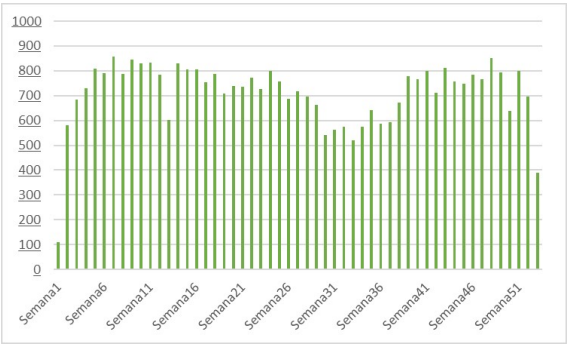
- *Comprensión de la incertidumbre.* Si bien realizar un análisis exhaustivo del comportamiento de los parámetros aleatorios puede ser costoso, este estudio puede conducir a una mejora en la comprensión de la incertidumbre que permita realizar mejoras en el propio modelo de programación estocástica.

Inconvenientes de la programación estocástica

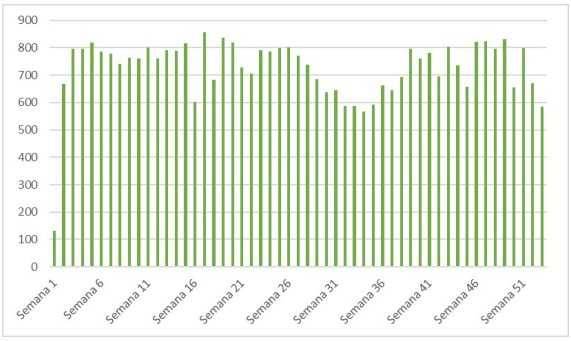
El uso de estos modelos de programación estocástica presenta algunas dificultades a considerar en su empleo. En primer lugar, es necesario recopilar y analizar una gran cantidad de datos históricos que permitan modelar la incertidumbre del problema, lo cual puede llegar a ser una tarea costosa. Por otro lado, la consideración de la incertidumbre sobre el modelo provoca un aumento de la complejidad computacional en la resolución de estos problemas. Pese a todo, es habitual que estos inconvenientes se vean paliados por las ventajas derivadas de introducir incertidumbre en los parámetros del modelo.

Centrándonos ya en el contexto hospitalario de este trabajo, y más concretamente en el PBA, se puede ver que la incorporación de la incertidumbre en la demanda diaria de pacientes al modelo de la sección anterior puede ayudar a reflejar situaciones en la que la demanda hospitalaria varíe sustancialmente a lo largo de un cierto periodo de tiempo. La idea detrás de esta clase de modelos estocásticos es proporcionar soluciones realistas que permitan minimizar tanto el coste de la asignación de recursos hospitalarios como el riesgo de colapso a largo plazo, asociado a un brote de una cierta enfermedad o a una catástrofe. Además, esta adaptabilidad de los modelos ante muy diversas situaciones permite que sean ideales para tratar situaciones imprevistas como con las que se tiene que lidiar a menudo en la actividad diaria de un hospital.

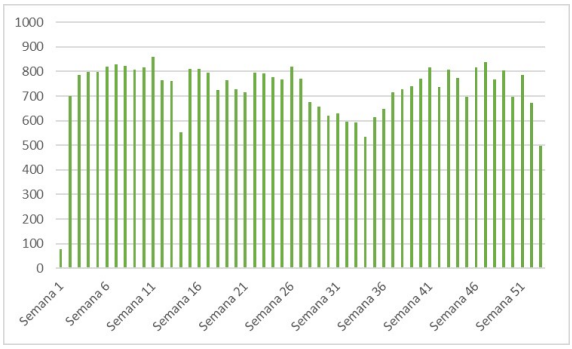
A la vista de las gráficas de la Figura 3.2 observamos que la variación en la tendencia de los ingresos hospitalarios no solo se produce a nivel interno en cada una de las áreas en el hospital, sino que se puede apreciar una clara tendencia global. En concreto, entre las semanas 30 y 40 de cada año se observa un valle, asociado al decrecimiento de ingresos y de actividad hospitalaria en los meses de verano. De hecho, también es posible observar que en las primeras semanas del año es en las que se produce un mayor incremento en la demanda hospitalaria, debido, en gran medida, a la tendencia de las enfermedades respiratorias infecciosas a presentar picos de contagio en estas épocas del año. Además, incluso en un año con gran incertidumbre como el 2020, debido a la influencia del COVID-19, es posible observar claras tendencias en el número de ingresos semanales, coincidentes con los picos de contagio de dicho virus. Por todo ello, parece clara la existencia de patrones en la tendencia de la



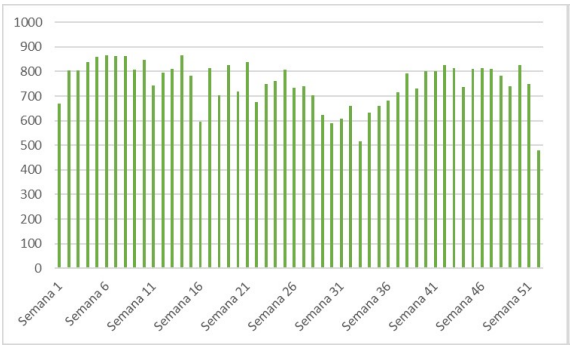
(a) Año 2016



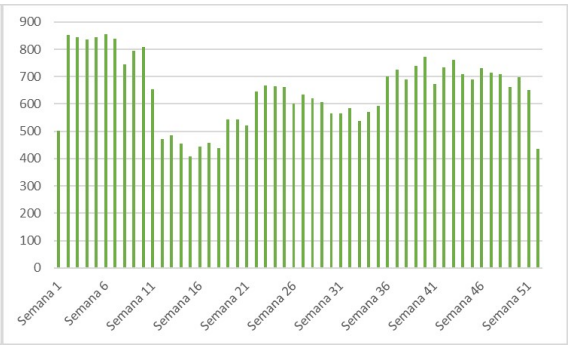
(b) Año 2017



(c) Año 2018



(d) Año 2019



(e) Año 2020

Figura 3.2: Evolución del número de ingresos semanales en el Hospital Clínico Universitario de Santiago de Compostela entre los años 2016 y 2020.

demanda hospitalaria y un modelo de programación estocástica permitirá modelar la incertidumbre provocada por la variabilidad de la misma.

Por otro lado, y como ya avanzábamos anteriormente, será preciso un análisis de los datos históricos de demanda de ingresos para poder modelar con precisión la incertidumbre en la cantidad de pacientes diarios que demandan ingresar en el hospital. Además, esta estimación de la incertidumbre debe ser precisa, pues de otro modo la calidad de las soluciones obtenidas podría ser notablemente inferior a la que se lograría con un modelo determinístico como el de la sección anterior. Además, un modelado exhaustivo de la incertidumbre puede derivar en un aumento substancial de variables y/o de restricciones en el problema, lo que podría provocar un aumento significativo en el coste computacional de resolución de este tipo de problemas.

A modo de resumen, frente a un enfoque diario de resolución del PBA, la programación estocástica ofrece una mayor capacidad para adaptarse a la incertidumbre y gestionar ciertos riesgos, lo que puede derivar en una asignación más eficiente de camas y otros recursos hospitalarios. Sin embargo, también requiere un mayor esfuerzo en términos de recopilación y análisis de datos, así como de recursos computacionales. Veremos posteriormente si este aumento en el coste computacional compensa la incorporación de la incertidumbre al modelo.

3.2. El modelo de programación estocástica

Una vez introducidas las ideas que fundamentan la programación estocástica, estamos en condiciones de presentar la versión del NL-PBA, incluyendo ahora la incertidumbre sobre la demanda hospitalaria en fechas posteriores.

Por la complejidad de la notación requerida para su correcta formulación, presentaremos en primer lugar un caso sencillo, con únicamente dos etapas, que permita la familiarización con la notación. Además, se presenta como aplicación un ejemplo sencillo, con datos reales que facilita la comprensión del propio modelo.

3.2.1. El problema de programación estocástica: un caso particular

Como avanzábamos en la sección introductoria de este capítulo, la condición indispensable en el modelado de la incertidumbre en un problema de programación estocástica pasa por suponer que los parámetros desconocidos del modelo siguen una cierta distribución que es conocida de antemano.

En este caso, supondremos tres situaciones para la demanda hospitalaria en una fecha posterior: que aumente en un $t_1\,\%$, con probabilidad $q^{1,1}$, que se mantenga constante con probabilidad $q^{1,2}$, o que descienda en un $t_2\,\%$, con probabilidad $1 - q^{1,1} - q^{1,2}$. Teniendo en cuenta estas tres situaciones, supondremos que la demanda hospitalaria seguirá una distribución discreta que le asignará probabilidad $q^{1,1}$ al evento asociado a que la demanda hospitalaria aumente, $q^{1,2}$ al asociado a que la demanda se mantenga constante, y $1 - q^{1,1} - q^{1,2}$ al asociado a que la demanda descienda.

Si bien la consideración del número de ingreso diarios como una variable discreta con tan solo tres escenarios posibles es demasiado restrictiva, es suficiente para dar una idea general del modelo y de su aplicación a situaciones reales.

Escenario 1	**Escenario 2**	**Escenario 3**
Aumento de la demanda en un $t_1\,\%$ con probabilidad $q^{1,1}$	Demanda constante con probabilidad $q^{1,2}$	Decrecimiento de la demanda en un $t_2\,\%$ con probabilidad $(1 - q^{1,1} - q^{1,2})$

Tabla 3.2: Comparativa de los distintos tipos de escenarios posibles en función de la variación de la demandas.

Una vez planteados los distintos tipos de escenarios a considerar y la distribución que sigue la demanda hospitalaria, estamos en condiciones de introducir los conjuntos de datos empleados en el modelo, así como los parámetros y las variables de decisión. La gran mayoría de estos elementos tendrán base en los modelos del capítulo anterior.

Sin embargo, será necesario introducir nuevos elementos asociados a la situación hospitalaria cambiante a lo largo de las distintas etapas. En concreto, y a diferencia de lo que ocurría en el contexto de los modelos del segundo capítulo, se prevé que a lo largo de las distintas etapas de estudio algunos pacientes reciban el alta hospitalaria, dando lugar a un nuevo número de camas disponibles en cada

situación. Por todo ello, no solo será necesaria una nueva notación asociada a las etapas y a los escenarios, sino que también se introducirán nuevos conjuntos, variables y parámetros y también nuevas restricciones en el problema de optimización. Más concretamente, representaremos la etapa con un primer superíndice y el escenario correspondiente con un segundo. Nótese que para un modelo dos etapas y tres escenarios posibles en cada una de ellas, para una etapa i, existen 3^i situaciones distintas, ya que cada escenario dependerá de las situaciones previas de demanda del hospital. Son estas situaciones las que inducen nuevos conjuntos de pacientes específicos para cada una de ellas. Para familiarizar al lector con la notación, en la Figura 3.3 se puede ver un esquema del problema de programación estocástica en dos etapas y tres escenarios, incluyendo la notación para los conjuntos de pacientes en cada uno de los nodos.

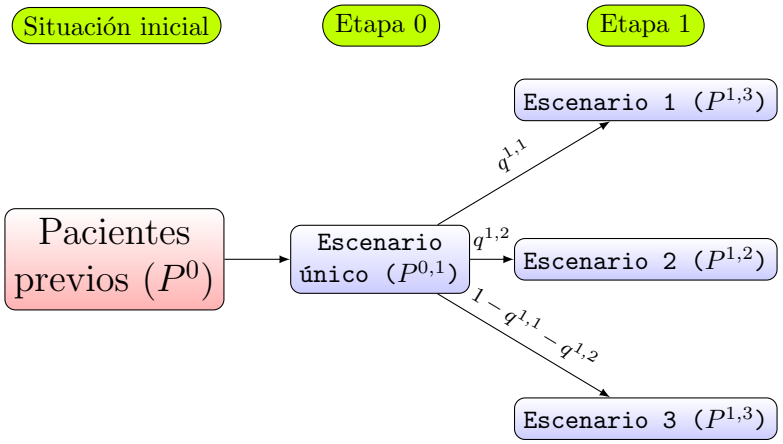


Figura 3.3: Esquema del problema de programción estocástica en dos etapas y tres escenarios.

La situación inicial hace referencia a la ocupación hospitalaria en el momento previo al estudio de las etapas posteriores. Para esta situación se tiene el conjunto de pacientes que están en el momento ingresados en el hospital en sus correspondientes camas. La Etapa 0 representa la asignación del conjunto de pacientes que es previsto que lleguen al hospital, es decir, la etapa considerada en los modelos del capítulo previo; y, finalmente, la Etapa 1 hace referencia a cada uno de los tres escenarios que pueden tener lugar en fechas posteriores.

Una vez presentado este esquema ilustrativo, pasemos a detallar los diferentes conjuntos de datos de este nuevo modelo.

Conjuntos

En primer lugar, consideramos los conjuntos de pacientes para cada una de las etapas y cada uno de los escenarios.

- P^0 denota el conjunto de pacientes ingresados en una cama del hospital antes de la etapa inicial.
- $P^{0,1}$ denota el conjunto de pacientes en la etapa inicial.
- $P^1 := \bigcup_{l<1,\ j\in\{1,\dots,3^l\}} P^{l,j}$ es el conjunto de pacientes que ingresaron o pretendían ingresar en el hospital antes de la etapa 1.
- $P^{1,j}$ denotas a los conjuntos de los pacientes en la primera etapa y en el escenario j, con $j = 1, 2, 3$.
- P_f^0 denota el conjunto de pacientes de género femenino ingresados en una cama del hospital antes de la etapa inicial.

- $P_f^{0,1}$ es el conjunto de pacientes de género femenino en la etapa inicial.
- $P_f^1 := \bigcup_{l<1} P_f^{l,j}$ denota al conjunto de pacientes de género femenino que ingresaron o pretendían ingresar en el hospital antes de la etapa 1.
- $P_f^{1,j}$, con $j = 1, 2, 3$, denota al conjunto de pacientes de género femenino en la primera etapa y en el escenario j.
- P_m^0 es el conjunto de pacientes de género masculino ingresados en una cama del hospital antes de la etapa inicial.
- $P_m^{0,1}$ representa al conjunto de pacientes de género masculino en la etapa inicial.
- $P_m^1 := \bigcup_{l<1\ j\in\{1,\dots,3^l\}} P_m^{l,j}$ denota a los pacientes de género masculino que ingresaron o pretendían ingresar en el hospital antes de la etapa 1.
- $P_m^{1,j}$, con $j = 1, 2, 3$, representa el conjunto de pacientes de género masculino en la primera etapa y en el escenario j.
- $P_p^{0,1}$ es el conjunto de pacientes cuyos ingresos están programados en la etapa inicial.
- $P_p^{1,j}$, con $j = 1, 2, 3$, denota el conjunto de pacientes programados en la primera etapa y en el escenario j.

Sin embargo, además de los conjuntos relativos a los pacientes, deben considerarse otro tipo de conjuntos. Estos son:

- T representa el conjunto de las camas totales del hospital/departamento hospitalario.
- H denota el conjunto de habitaciones del hospital.
- H_d representa el conjunto de aquellas habitaciones con más de una cama.
- D incluye a todos los departamentos o áreas del hospital.
- N considera los diferentes niveles de prioridad asociado a los pacientes. Estos se corresponden con valores que se le asignan a cada paciente en función de su nivel de riesgo y basados en características del mismo, como la gravedad de su condición física o su edad.

Se puede observar que por cada conjunto de datos asociados a los pacientes que desean ingresar, es necesario crear nuevos conjuntos asociados a cada uno de los escenarios de demanda hospitalaria previstos, así como a las etapas de estudio. Por otro lado, y en contraposición a lo que se presentaba en los modelos de programación matemática general, se considerará el total de camas y no solo las disponibles en la etapa inicial. Esto se debe a la necesidad de tener en cuenta camas en las que hay pacientes ingresados que se prevé que puedan recibir el alta hospitalaria en etapas posteriores. Por todo ello, será necesario tener una previsión de la duración de la estancia tanto de los pacientes que ya están ingresados en el hospital como de los que pretenderán ingresar en cada una de las distintas situaciones.

Parámetros del modelo

Nuevamente, se presentarán modificaciones en la notación de la misma clase de las empleadas para definir los conjuntos de datos. Además, se introducirá un nuevo conjunto de parámetros relacionado con la duración de estancia esperada de cada paciente.

$$Ubi_{th} = \begin{cases} 1, & \text{si la cama } t \in T \text{ está en la habitación } h \in H. \\ 0, & \text{en otro caso.} \end{cases}$$

$$DepHab_{hd} = \begin{cases} 1, & \text{si la habitación } h \in H \text{ está en el departamento } d \in D. \\ 0, & \text{en otro caso.} \end{cases}$$

$$DepOpt_{pd}^{0,1} = \begin{cases} 1, & \text{si el departamento } d \in D \text{ es óptimo para el paciente } p \in P^{0,1}. \\ 0, & \text{en otro caso.} \end{cases}$$

$$\forall j \in \{1,2,3\},\ DepOpt_{pd}^{1,j} = \begin{cases} 1, & \text{si el departamento } d \in D \text{ es óptimo para el paciente } p \in P^{1,j}. \\ 0, & \text{en otro caso.} \end{cases}$$

$$X_{pt}^{0} = \begin{cases} 1, & \text{si el paciente } p \in P^0 \text{ ocupa la cama } t \in T \text{ antes de la etapa inicial.} \\ 0, & \text{en otro caso.} \end{cases}$$

- F_p^0 denota, para cada paciente $p \in P^0$, el valor de la duración esperada de su estancia en el hospital, medida en número de etapas. Para los pacientes ingresados previamente en el hospital se considera su duración esperada de la estancia como el número de etapas, a partir del momento del estudio, que se prevé que continúe ingresado.
- $F_p^{0,1}$ representa, para cada paciente $p \in P^{0,1}$, el valor de la duración esperada de su estancia en el hospital, medida en número de etapas.
- $RiesgoPac_p^{0,1}$ es, para cada paciente $p \in P^{0,1}$, el valor de riesgo $n \in N$ asociado.
- $RiesgoPac_p^{1,j}$, con $j \in \{1,2,3\}$, asocia a cada paciente $p \in P^{1,j}$ el valor de riesgo $n \in N$ asociado.
- b_D representa la tasa de beneficio por situar a un paciente en el departamento adecuado.
- b_N representa la tasa de beneficio por ingresar a un paciente con prioridad alta.
- b_P representa la tasa de beneficio por evitar la cancelación de un paciente programado.
- η representa el umbral de saturación máxima permitida en la ocupación hospitalaria.
- $q^{1,j}$, con $j = 1,2,3$, denota la distribución de probabilidades para cada uno de los escenarios posibles.

Nuevamente observamos que el número de parámetros se incrementa. Si bien esto no tendría porque derivar en un aumento en el número de restricciones, sí que podría provocar que el tiempo de lectura de los datos del problema y el de manejo de la incertidumbre aumentase.

Variables del modelo

En el segundo capítulo se consideraban dos tipos distintos de variables de decisión para este problema. El primer tipo guardaba relación con la cama que se le asigna a cada paciente (si es que se le

asigna una) y el segundo tipo se refería a saber si el paciente ingresa en el departamento que le debería corresponder según su patología.

Sin embargo, en este modelo estocástico es necesario considerar un nuevo tipo de variables relacionadas con la ocupación de las camas antes de cada etapa.

$$x_{pt}^{0,1} = \begin{cases} 1, & \text{si se le asigna la cama } t \in T \text{ al paciente } p \in P^{0,1} \text{ en la etapa inicial.} \\ 0, & \text{en otro caso.} \end{cases}$$

$$\forall j \in \{1,2,3\},\ x_{pt}^{1,j} = \begin{cases} 1, & \text{si se le asigna la cama } t \in T \text{ al paciente } p \in P^{1,j} \text{ en la etapa 1.} \\ 0, & \text{en otro caso.} \end{cases}$$

$$y_{p}^{0,1} = \begin{cases} 1, & \text{si al paciente } p \in P^{0,1} \text{ se le asigna una cama del departamento más apropiado.} \\ 0, & \text{en otro caso.} \end{cases}$$

$$\forall j \in \{1,2,3\},\ y_{p}^{1,j} = \begin{cases} 1, & \text{si al paciente } p \in P^{1,j} \text{ se le asigna una cama} \\ & \quad \text{del departamento más apropiado.} \\ 0, & \text{en otro caso.} \end{cases}$$

$$z_{pt}^{0} = \begin{cases} 1, & \text{si la cama } t \in T \text{ está ocupada por el paciente } p \in P^{0} \text{ antes de la etapa inicial.} \\ 0, & \text{en otro caso.} \end{cases}$$

$$z_{pt}^{1} = \begin{cases} 1, & \text{si la cama } t \in T \text{ está ocupada por el paciente } p \in P^{1} \text{ antes de la etapa 1.} \\ 0, & \text{en otro caso.} \end{cases}$$

En este caso, el número de variables del modelo se ve aumentado aproximadamente en un 300 %. Este porcentaje es en realidad variable, pues depende del tamaño de los conjuntos de datos de pacientes. Sin embargo, consideraremos esto como una aproximación suponiendo que $|P^{1,j}| = |P^{0,1}|$, $\forall j \in \{1,2,3\}$. Este incremento, al contrario del que se daba para los conjuntos de datos y los parámetros del modelo, puede ser significativo, ya que influirá en el tiempo de resolución del problema.

Formulación del modelo estocástico

Una vez introducidos los conjuntos de datos, los parámetros y las variables, estamos en condiciones de definir el modelo de programación estocástica en dos etapas y considerando tres situaciones.

$$\begin{aligned}\text{maximizar } & \sum_{p\in P^{0,1}} y_p^{0,1} d_D + \sum_{p\in P^{0,1}}\sum_{t\in T} x_{pt}^{0,1} RiesgoPac_p^{0,1} b_N + \sum_{p\in P_p^{0,1}}\sum_{t\in T} x_{pt}^{0,1} b_P \\ & + q^{1,1}\left(\sum_{p\in P^{1,1}} y_p^{1,1} d_D + \sum_{p\in P^{1,1}}\sum_{t\in T} x_{pt}^{1,1} RiesgoPac_p^{1,1} b_N + \sum_{p\in P_p^{1,1}}\sum_{t\in T} x_{pt}^{1,1} b_P\right) \\ & + q^{1,2}\left(\sum_{p\in P^{1,2}} y_p^{1,2} d_D + \sum_{p\in P^{1,2}}\sum_{t\in T} x_{pt}^{1,2} RiesgoPac_p^{1,2} b_N + \sum_{p\in P_p^{1,2}}\sum_{t\in T} x_{pt}^{1,2} b_P\right) \\ & + (1-q^{1,1}-q^{1,2})\left(\sum_{p\in P^{1,3}} y_p^{1,3} d_D + \sum_{p\in P^{1,3}}\sum_{t\in T} x_{pt}^{1,3} RiesgoPac_p^{1,3} b_N \right. \\ & \left. + \sum_{p\in P_p^{1,3}}\sum_{t\in T} x_{pt}^{1,3} b_P\right)\end{aligned} \tag{3.2}$$

$$\text{sujeto a } \sum_{t\in T} x_{pt}^{i,j} \le 1,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\},\ \forall p\in P^{i,j} \tag{3.3}$$

$$\sum_{p\in P^{i,j}} x_{pt}^{i,j} \le 1,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\},\ \forall t\in T \tag{3.4}$$

$$\frac{1}{|T|}\left(\sum_{p\in P^{i,j}}\sum_{t\in T} x_{pt}^{i,j} + \sum_{p'\in P^0}\sum_{t\in T} z_{p't}^{i}\right) \le \eta,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\} \tag{3.5}$$

$$y_p^{i,j} - \sum_{t\in T}\sum_{h\in H}\sum_{d\in D} x_{pt}^{i,j} Ubi_{th} DepHab_{hd} DepOpt_{pd}^{i,j} = 0,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\}, \tag{3.6}$$

$$\forall p\in P^{i,j} \tag{3.7}$$

$$\sum_{p\in P_p^{i,j}}\sum_{t\in T} x_{pt}^{i,j} \le |P_p^{i,j}|,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\} \tag{3.8}$$

$$\Big(\sum_{p\in P_f^{i,j}}\sum_{t\in T} x_{pt}^{i,j} Ubi_{th}\Big)\Big(\sum_{p\in P_m^{i,j}}\sum_{t\in T} x_{pt}^{i,j} Ubi_{th}\Big) = 0,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\},\ \forall h\in H_d \tag{3.9}$$

$$\Big(\sum_{p\in P_f^{i}}\sum_{t\in T} z_{pt}^{i} Ubi_{th}\Big)\Big(\sum_{p\in P_m^{i,j}}\sum_{t\in T} x_{pt}^{i,j} Ubi_{th}\Big) = 0,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\},\ \forall h\in H_d \tag{3.10}$$

$$\Big(\sum_{p\in P_f^{i,j}}\sum_{t\in T} x_{pt}^{i,j} Ubi_{th}\Big)\Big(\sum_{p\in P_m^{i}}\sum_{t\in T} z_{pt}^{i} Ubi_{th}\Big) = 0,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\},\ \forall h\in H_d \tag{3.11}$$

$$z_{pt}^0 + z_{pt}^1 = X_{pt}^0 \,\text{mín}\{F_p^0, 2\},\ \forall p\in P^0,\ \forall t\in T \tag{3.12}$$

$$z_{pt}^1 = x_{pt}^{0,1} \,\text{mín}\{F_p^{0,1}-1, 1\},\ \forall p\in P^{0,1},\ \forall t\in T \tag{3.13}$$

$$z_{pt}^0 \ge z_{pt}^1,\ \forall p\in P^0,\ \forall t\in T \tag{3.14}$$

$$z_{pt}^1 - x_{pt}^{0,1} \le 0,\ \forall p\in P^{0,1},\ \forall t\in T \tag{3.15}$$

$$\Big(\sum_{p\in P^i} z_{pt}^i\Big)\Big(\sum_{p'\in P^{i,j}} x_{pt}^{i,j}\Big) = 0,\ \forall i\in\{0,1\},\ \forall j\in\{1,\dots,3^i\},\ \forall t\in T \tag{3.16}$$

Se denotará a este problema de optimización como PBA-s a lo largo del trabajo. Tanto la estructura de la función objetivo como la de las restricciones ha cambiado, por lo que es necesario pararse a entender las diferencias explicando el porqué de cada modificación.

- En primer lugar, la función objetivo en (3.2) mantiene los tres primeros sumandos similares a los del modelo de programación matemática del capítulo anterior. Sin embargo, posteriormente se introducen nuevos sumandos correspondientes a los nuevos escenarios de demanda de hospitalización. Cada uno de ellos está multiplicado por las probabilidades $q^{1,j}$, que representan los incrementos en la función objetivo asociados a la asignación de los pacientes de la primera etapa a las camas disponibles del hospital.

- Para las restricciones, si bien algunos bloques se mantienen prácticamente idénticos, es necesario introducir algunos nuevos, por lo que se detallarán todos los bloques, tanto los nuevos como los que no lo son.

  – Al igual que ocurría para el NL-PBA, el bloque de restricciones en 3.3 modela el hecho de que a un paciente que ingresa en la etapa i no se le puede asignar más de una cama.

  – El bloque de restricciones en 3.4 tampoco es nuevo, y controla que no se le asigne la misma cama a dos pacientes que ingresan en la etapa i.

  – Para las restricciones en 3.5 se sigue buscando que el número total de camas usadas no supere el umbral η. Sin embargo, es necesario considerar en este caso también las variables z que representan la ocupación de una cama antes de cada etapa.

  – Los bloques 3.6 y 3.8 también son similares a los del modelo general y son necesarios para relacionar las variables x e y y para regular el número total de pacientes programados que ingresan en el hospital, respectivamente.

  – Las restricciones en 2.7 regulaban que no hubiera en la misma habitación dos personas de distinto género. Sin embargo, debido a la inclusión de las variables z, estas restricciones se transforman en los bloques 3.9, 3.10 y 3.11, que modelan no solo el hecho de que no pueden ingresar en la misma etapa dos personas del distinto sexo en la misma habitación, sino también que una persona no podrá ingresar en una cama si en esa misma habitación ya hay ingresada otra persona de sexo opuesto.

- Hasta el momento, se han detallado las nuevas versiones de las restricciones del modelo general. Sin embargo, es necesario añadir bloques nuevos de restricciones con objeto de relacionar las variables z con las x así como de actualizar los valores de las variables z en función de los valores de los parámetros F.

  – Más concretamente, las restricciones en 3.12 inicializan las variables z asignadas a pacientes ya ingresados en el hospital, de manera que estas variables tomarán el valor 1 según la duración esperada de la estancia del paciente que ocupa la cama.

  – Por otro lado, el bloque de restricciones en 3.13 sirve para modelar en cada etapa i las actualizaciones de las variables z en función de las asignaciones x_{pt} y de la duración esperada de la estancia del paciente.

  – Los bloques de restricciones en 3.14 y 3.15 relacionan las variables entre sí. Más específicamente, las restricciones 3.14 regulan los valores que puede tomar cada variable z_{pt} en función de la situación del paciente en la etapa previa, mientras que las restricciones 3.15 relacionan las variables x de la etapa inicial con las variables z de la primera etapa.

  – Finalmente, el bloque de restricciones en 3.16 controla el hecho de que si hay algún paciente ocupando la cama t antes de la etapa i, esta cama no podrá ser asignada a un paciente que ingresa en dicha etapa.

Una vez introducido el problema, no es difícil ver que, incluso para una situación de tamaño relativamente pequeño como la de esta sección, el número de restricciones y variables ha aumentado considerablemente con respecto al NL-PBA presentado en el capítulo 2.1. Al igual que ocurría con el L-PBA, este incremento en la complejidad del problema será significativo en términos del tiempo de resolución del mismo. De nuevo, se comprobará esto con un ejemplo con un conjunto de datos reales.

3.2.2. Comparativa con el NL-PBA

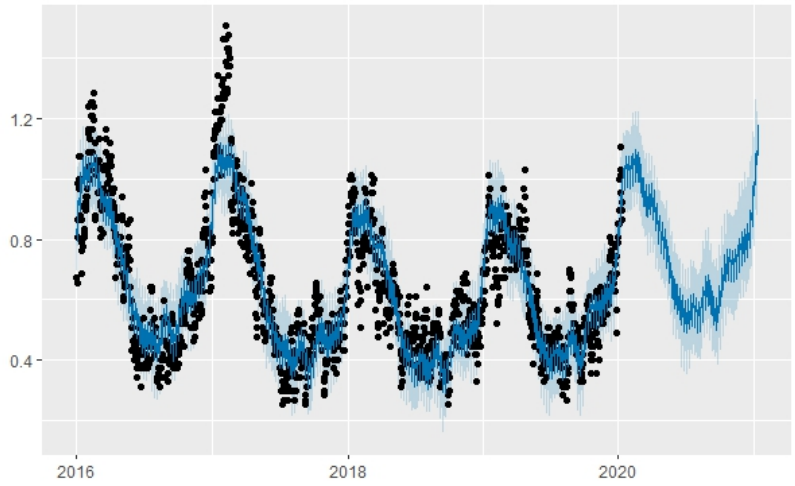
El modelo estocástico presentado en la sección anterior tiene como objetivo proporcionar una asignación de los pacientes a las camas teniendo en cuenta la incertidumbre asociada a la variabilidad de demanda hospitalaria prevista en fechas posteriores. Este modelado de la incertidumbre puede ser de especial utilidad en situaciones en las que se perciba una cierta tendencia al alza en el número de casos asociados a un área particular del hospital, debido a distintas casuísticas como el pico de contagio de una cierta enfermedad. El objetivo de esta sección será analizar si, en efecto, un buen estudio de la demanda de pacientes facilita que el PBA-s logre un incremento significativo en el valor de la función objetivo.

En primer lugar, el PBA-s debe nacer en un contexto en el que exista un método predictivo eficaz para la demanda hospitalaria. Para la elaboración de este modelo predictivo se empleará la herramienta *Prophet* ideada en Taylor y Letham (2018) y que tiene como objetivo realizar predicciones de datos cronológicos. Se recurrirá a esta herramienta debido a la dificultad de encontrar un modelo ARIMA no demasiado complejo que modele el número de ingresos diarios en el hospital. El código en lenguaje R asociado a la predicción del número de ingresos se puede ver en el Apéndice C.

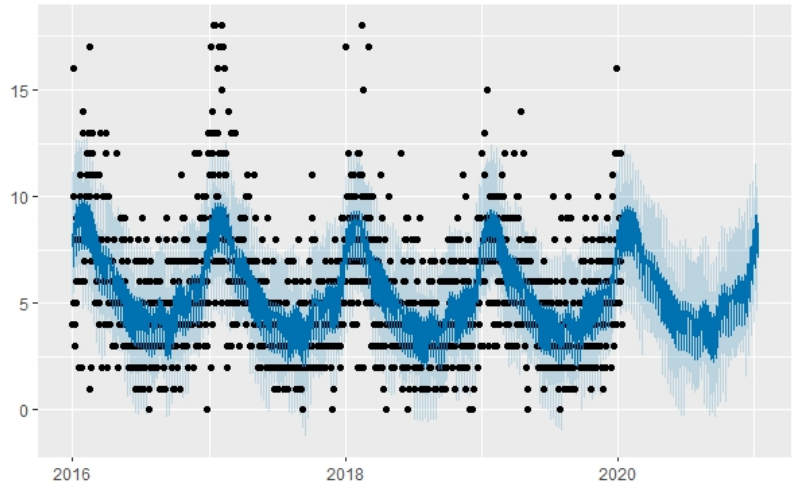
Para buscar una situación idónea en la que el uso del modelo estocástico se espera que produzca una notable mejoría con respecto al NL-PBA, basta con observar de nuevo la Figura 3.1 en la que se presentan gráficas asociadas al porcentaje de ocupación del área de Neumología entre los años 2016 y 2019. Se observaba la existencia una gran diferencia entre los primeros meses del año, en los que algunas enfermedades respiratorias estacionales tienen picos de contagio, y los demás meses. Además, en estos primeros meses del año se observaban ocupaciones del área muy elevadas, que incluso podían llegar a sobrepasar el 100 %[2]. Esta tendencia anual que presenta el valor del porcentaje de ocupación del área de Neumología, así como la saturación parcial o total del área durante los primeros meses del año, hacen que esta sea una situación ideal para realizar la comparativa entre el PBA-s y el NL-PBA del segundo capítulo.

En la Figura 3.4a se pueden ver en color azul las predicciones propuestas para el nivel de ocupación del área de Neumología. Además, en esta gráfica se puede apreciar no solo que las primeras semanas de cada año son las más críticas, sino que existen tendencias claras en el resto de los meses. Es por ello que la fecha inicial que se ha escogido es el domingo 12 de enero de 2020. Esta elección se debe no solo a la ubicación del día en el calendario, sino también al propio día de la semana, ya que, a menudo, los domingos son días con pocos ingresos en el área (y en el global del hospital también) y los lunes son días con mucha afluencia de pacientes. Por todo ello, compararemos el uso del modelo estocástico en dos etapas y tres escenarios sobre este día, es decir, se tomará el día 12/01/2020 como etapa inicial y el día 13/01/2020 como primera etapa. Antes de comenzar el día 12, hay ingresados en el área de Neumología un total de 62 pacientes ocupando en torno a un 93 % de las 67 camas habilitadas para dicha área, por lo que solo hay 5 camas disponibles asociadas al área de Neumología. Además, se sabe que el propio día 12 buscarán ingresar 4 pacientes con patologías diversas.

[2]Esto ocurre cuando no solo se ocupan todas las camas del área asociada, sino que es necesario ingresar a pacientes en otras áreas que no le corresponderían según su patología.



(a) Evolución de predicciones del porcentaje de ocupación del área de Neumología tomando como base los datos hasta el domingo 12 de enero de 2020.



(b) Evolución de predicciones del número de ingresos diarios en el área de Neumología tomando como base los datos hasta el domingo 12 de enero de 2020.

Figura 3.4: Predicciones, en color azul oscuro, y bandas de confianza, en color azul claro, tanto para el porcentaje de ocupación como para el número de ingresos diarios del área de Neumología, empleando la herramienta *Prophet*.

Por otro lado, la predicción para el día 13 de enero es que un total de 9 pacientes[3] buscarán ingresar en el área de Neumología, es decir, el modelo predictivo anticipa que la demanda hospitalaria aumentará en un 125 %. Este será el primer escenario que se considerará para el modelo estocástico y se le otorgará una probabilidad del 80 %. Además, también se tendrá en cuenta el escenario en el que la demanda hospitalaria se mantenga constante (4 pacientes) y al que se le otorgará una probabilidad del 15 % y, finalmente, un tercer escenario, con probabilidad del 5 % en el que la demanda hospitalaria decaerá un 25 %, teniendo tan solo 3 pacientes a ingresar el día 13. Estos escenarios están recogidos en la Tabla 3.3.

Escenario 1	**Escenario 2**	**Escenario 3**
Aumento de la demanda hospitalaria en un 125 % con probabilidad 0.8.	Demanda hospitalaria constante con probabilidad 0.15.	Decrecimiento de la demanda hospitalaria en un 25 % con probabilidad 0.05.

Tabla 3.3: Comparativa de los distintos tipos de escenarios posibles en función de la variación de la demandas.

Una vez presentados los distintos escenarios y las probabilidades asociadas a cada uno de ellos, se puede resolver el problema empleando el PBA-s en dos etapas y tres escenarios presentado en la sección anterior. El código en lenguaje AMPL de este ejemplo se puede consultar en el Apéndice C. La idea de esta comparativa es buscar evitar, en la medida de lo posible, la saturación del área de Neumología, por lo que, al contrario que en los ejemplos de las Secciones 2.3.1 y 2.3.2, no se considerará un número de camas añadidas en otra área en previsión de que no queden camas disponibles. Además, como se busca estudiar el posible colapso del área, se tomará $\eta = 1$. Para realizar la comparativa entre las funciones objetivo asociadas a cada uno de los dos modelos debemos definir previamente qué entendemos por función objetivo del NL-PBA al considerar varias etapas. Con respecto a esto y a lo largo de todo el trabajo, se considerará como valor de la función objetivo el asociado a la etapa inicial sumado a los asociados a las etapas posteriores promediados en función de la probabilidad de cada escenario.

En la Tabla 3.4 se puede ver la comparativa entre los resultados obtenidos para cada uno de los enfoques del problema. En ella se puede observar que, pese a que tanto el número de variables y de restricciones es más elevado que si consideramos el enfoque diario, esta diferencia no es demasiado significativa en los tiempos de resolución del problema, ya que, a excepción del *solver* Couenne para el cual la diferencia es algo más significativa, se obtiene la solución del problema en pocas décimas de segundo. Sin embargo, el detalle más importante de esta tabla está en la fila del valor de la función objetivo, ya que el uso del PBA-s logra aumentar en 27.5 unidades el valor de esta función. Esto se debe a que el modelo predictivo permite anticipar un incremento en el número de ingresos que facilita el aumento en el valor de la función objetivo gracias a la flexibilidad del modelo estocástico. Más concretamente, la solución óptima consiste en no ingresar en la etapa inicial a dos pacientes que sí tienen una cama asignada en la solución asociada al modelo original, para asignarles esas camas a dos pacientes que ingresan el día posterior con un cuadro más grave y cuyos ingresos, por lo tanto, producen una mejora notable en el valor de la función objetivo.

Si bien el PBA-s no tiene por qué siempre producir una mejora en el valor de la función objetivo, este es un claro ejemplo en el cual la implementación de un modelo predictivo eficaz sumado al uso del modelo estocástico supone un beneficio sustancial en el valor de la función objetivo y, por tanto, en la calidad de los servicios proporcionados por el hospital. Sin embargo, como se verá en secciones posteriores, no todo son ventajas al considerar el enfoque estocástico, y es que si bien en la Tabla 3.4 no se observan diferencias muy significativas en el tiempo de resolución pese a que el número de

[3] A modo de curiosidad, el día 13 de enero, ingresaron 9 pacientes en el área de Neumología.

	NL-PBA	**PBA-s**
Variables	104	454
Restricciones	67	330
Valor de la función objetivo	1230	1257.5
Tiempo medio de resolución	$0.08 - 0.11 - 0.04 - 0.03$	$0.03 - 0.22 - 2.65 - 0.03$
Número medio de iteraciones	$33 - 0^* - 22.86 - 0^*$	$20.5 - 1^* - 64.3 - 1^*$
Número mínimo de iteraciones	$32 - 0^* - 22 - 0^*$	$20 - 1^* - 63 - 1^*$
Número máximo de iteraciones	$34 - 0^* - 23 - 0^*$	$21 - 1^* - 65 - 1^*$

Tabla 3.4: Comparativa del número de variables y de restricciones de los modelos asociados a cada uno de los distintos enfoques, así como el valor de la función objetivo alcanzado y los tiempos de resolución empleando cuatro *solvers* diferentes: Gurobi, BARON, Couenne y OCTERACT, en el orden respectivo. Se han considerado cien instantes iniciales distintos para cada *solver*. Los valores de los parámetros del modelo empleados son: $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 1$.

variables y restricciones es más de cuatro veces más que las que se tenían para el NL-PBA, es posible que para otro tipo de escenarios estas diferencias en el tamaño de problema provocasen un incremento grave en el tiempo de resolución del problema. Para poder analizar estas posibles desventajas, se debe formular primero un modelo estocástico general para un número k de etapas y considerando m posibles escenarios distintos para cada etapa.

3.2.3. El problema de programación estocástica: formulación general

Una vez presentada la versión reducida del modelo estocástico para el caso de dos etapas vamos a formalizar el modelo de programación estocástica para el caso general, en el que se tenga un número general k de etapas, es decir, en el que el horizonte de estudio del PBA sea tan lejano como se desee. Además, se mantendrá la suposición de que el número de ingresos seguirá una distribución discreta, pero sujeta en este caso a m escenarios distintos en cada una de las k etapas. El hecho de poder considerar un número general de escenarios posibles dota de versatilidad al modelo para poder reflejar de manera fidedigna la realidad. Pese a todo, estas mejoras en la adaptabilidad del modelo incrementarán la dificultad en su resolución.

Con respecto a la ofrecida en la sección anterior, la formulación general del problema de programación estocástica no es tan sencilla en cuanto a su escritura, si bien la estructura se mantiene. La notación para este modelo será similar a la introducida para el modelo previo, empleando superíndices que reflejen no solo la etapa correspondiente sino también el escenario asociado. Sin embargo, ahora será algo más tediosa, especialmente en cuanto a la función objetivo, debido a que un número general de etapas y de escenarios dificulta en cierta medida el trabajo con los superíndices.

Para reflejar esta dificultad en la notación, y con propósito de ayudar a la comprensión de la expresión de la función objetivo del problema, se puede ver en la Figura 3.5 un diagrama en árbol para un modelo en tres etapas y con m escenarios distintos en cada una de ellas. En este diagrama en árbol se recogerán, además de los escenarios posibles en cada etapa, la notación asociada a los mismos para los conjuntos de pacientes. El resto de conjuntos, parámetros y variables mantendrán una notación

análoga.

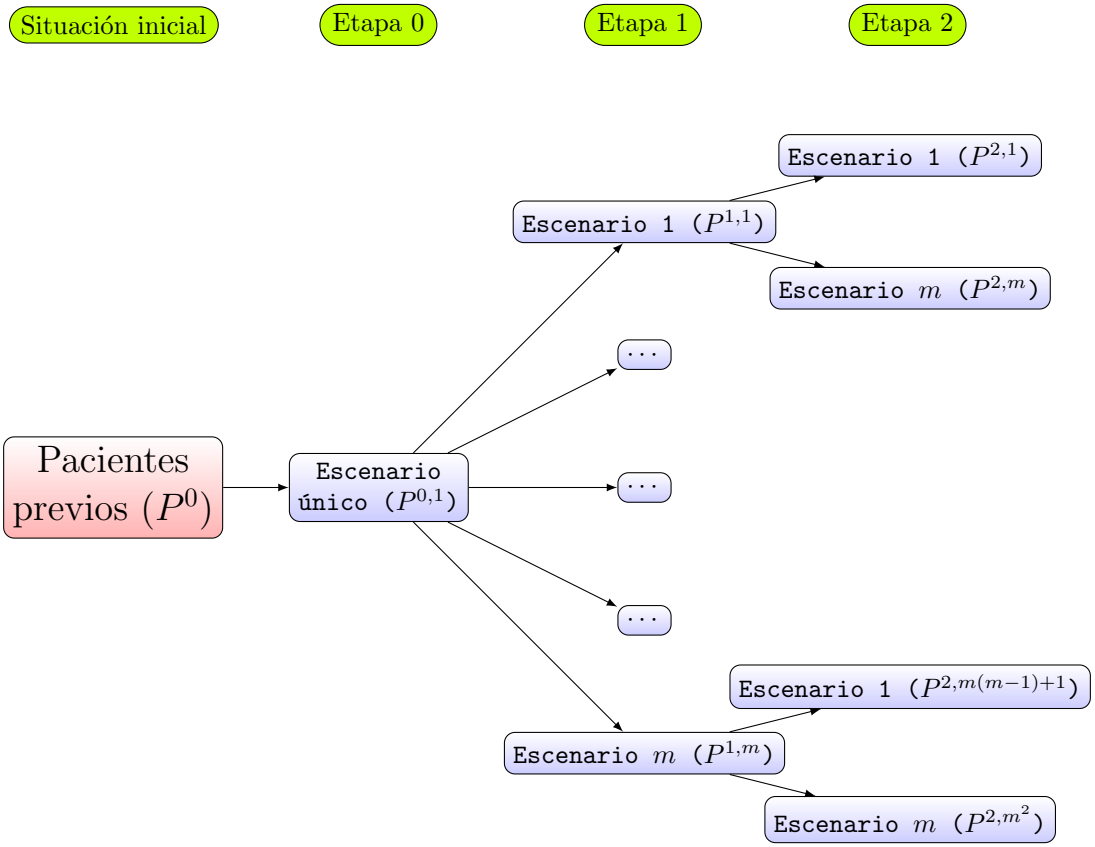


Figura 3.5: Esquema del problema de programación estocástica en tres etapas y m escenarios.

Más allá de la notación, el principal problema que presentan este tipo de problemas generales de programación estocástica reside en las dimensiones asociadas. De hecho, en el diagrama 3.5 se puede ver como el número de conjuntos de pacientes sufre un crecimiento exponencial de la forma m^i, donde m es el número de escenarios e i el número de etapa correspondiente. Esto puede provocar que no sea posible dotar de un elevado número de escenarios a la variable aleatoria discreta que describe la incertidumbre en la demanda de pacientes, pues esto provocaría un incremento muy elevado en el número de conjuntos del problema. Además, tampoco sería viable considerar horizontes muy lejanos en el tiempo, pues de nuevo un número elevado de etapas dispararía el número de conjuntos de datos del problema. Si bien un incremento en el número de conjuntos no tendría por qué producir un incremento significativo en el tiempo de resolución del problema, es común que la creación de nuevos conjuntos de datos supongan un incremento en el número de variables o restricciones y, como se verá a continuación, este problema no será una excepción.

Con el propósito de evitar la repetición en la definición de los conjuntos, parámetros y variables del modelo, que se definen de forma análoga al modelo en dos etapas, pasemos a definir el problema de programación estocástica en el caso general, con k etapas y m escenarios.

$$
\begin{aligned}
\text{maximizar } & \sum_{p\in P^{0,1}} y_p^{0,1} d_D + \sum_{p\in P^{0,1}} \sum_{t\in T} x_{pt}^{0,1} RiesgoPac_p^{0,1} b_N + \sum_{p\in P^{0,1}} \sum_{t\in T} x_{py}^{0,1} Prog_p^{0,1} b_P \\
& + q^{1,1} \left(\sum_{p\in P^{1,1}} y_p^{1,1} d_D + \sum_{p\in P^{1,1}} \sum_{t\in T} x_{pt}^{1,1} RiesgoPac_p^{1,1} b_N + \sum_{p\in P^{1,1}} \sum_{t\in T} x_{pc}^{1,1} Prog_p^{1,1} b_P \right. \\
& + q^{2,1} \left(\cdots + q^{k-1, m^{k-3}-m+1} (\cdots) + \cdots + \left(1 - \sum_{i=m^{k-3}-m+1}^{m^{k-3}-1} q^{k-1,i} \right) (\cdots) \right) + \cdots \\
& + \left(1 - \sum_{i=1}^{m-1} q^{2,i} \right) \left(\cdots + q^{k-1, m^{k-2}-m+1} (\cdots) + \cdots + \left(1 - \sum_{i=m^{k-2}-m+1}^{m^{k-2}-1} q^{k-1,i} \right) (\cdots) \right) \Bigg) \\
& + q^{1,2} \left(\sum_{p\in P^{1,2}} y_p^{1,2} d_D + \sum_{p\in P^{1,2}} \sum_{t\in T} x_{pt}^{1,2} RiesgoPac_p^{1,2} b_N \right. \\
& \left. + \sum_{p\in P^{1,2}} \sum_{t\in T} x_{pc}^{1,2} Prog_p^{1,2} b_P + \cdots \right) + \cdots + \left(1 - \sum_{i=1}^{m-1} q^{1,i} \right) \left(\sum_{p\in P^{1,m}} y_p^{1,m} d_D \right. \\
& \left. + \sum_{p\in P^{1,m}} \sum_{t\in T} x_{pt}^{1,m} RiesgoPac_p^{1,m} b_N + \sum_{p\in P^{1,m}} \sum_{t\in T} x_{pt}^{1,m} Prog_p^{1,m} b_P + \cdots \right) \\
\text{sujeto a } & \sum_{t\in T} x_{pt}^{i,j} \leq 1,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\},\ \forall p \in P^{i,j} \\
& \sum_{p\in P^{i,j}} x_{pt}^{i,j} \leq 1,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\},\ \forall t \in T \\
& \frac{1}{|T|} \left(\sum_{p\in P^{i,j}} \sum_{t\in T} x_{pt}^{i,j} + \sum_{p'\in P^0} \sum_{t\in T} z_{p't}^{i} \right) \leq \eta,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\} \\
& y_p^{i,j} - \sum_{t\in T} \sum_{h\in H} \sum_{d\in D} x_{pt}^{i,j} Ubi_{th} DepHab_{hd} DepOpt_{pd}^{i,j} = 0, \\
& \qquad \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\},\ \forall p \in P^{i,j} \\
& \sum_{p\in P_p^{i,j}} \sum_{t\in T} x_{pt}^{i,j} \leq |P_p^{i,j}|,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\} \\
& \Big(\sum_{p\in P_f^{i,j}} \sum_{t\in T} x_{pt}^{i,j} Ubi_{th} \Big) \Big(\sum_{p\in P_m^{i,j}} \sum_{t\in T} x_{pt}^{i,j} Ubi_{th} \Big) = 0,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\},\ \forall h \in H_d \\
& \Big(\sum_{p\in P_f^{i}} \sum_{t\in T} z_{pt}^{i} Ubi_{th} \Big) \Big(\sum_{p\in P_m^{i,j}} \sum_{t\in T} x_{pt}^{i,j} Ubi_{th} \Big) = 0,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\},\ \forall h \in H_d \\
& \Big(\sum_{p\in P_f^{i,j}} \sum_{t\in T} x_{pt}^{i,j} Ubi_{th} \Big) \Big(\sum_{p\in P_m^{i}} \sum_{t\in T} z_{pt}^{i} Ubi_{th} \Big) = 0,\ \forall i \in \{0,\ldots,k-1\},\ \forall j \in \{1,\ldots,m^i\},\ \forall h \in H_d \\
& \sum_{l=1}^{k} z_{pt}^{l-1} = X_{pt}^{0} \min\{F_p^0, k\},\ \forall p \in P^0,\ \forall t \in T
\end{aligned}
$$

$$\sum_{l=i+2}^{k} z_{pt}^{l-1} = x_{pt}^{i,j} \text{ mín}\{F_p^{i,j} - 1, k-i-1\}, \forall i \in \{0,\dots,k-2\}, \forall j \in \{1,\dots,m^i\},$$
$$\forall p \in P^{i,j}, \forall t \in T$$
$$z_{pt}^{i} \geq z_{pt}^{i+1}, \forall i \in \{0,\dots,k-2\}, \forall p \in P^i, \forall t \in T$$
$$z_{pt}^{i+1} - x_{pt}^{i,j} \leq 0, \forall i \in \{0,\dots,k-2\} \forall j \in \{1,\dots,m^i\}, \forall p \in P^{i,j}, \forall t \in T$$
$$\left(\sum_{p \in P^i} z_{pt}^{i}\right)\left(\sum_{p' \in P^{i,j}} x_{pt}^{i,j}\right) = 0, \forall i \in \{0,\dots,k-1\}, \forall j \in \{1,\dots,m^i\}, \forall t \in T$$

Cometiendo un pequeño abuso en la notación, se empleará el término PBA-s para referirnos tanto al modelo en dos etapas y tres escenarios como a este modelo más general. Como avanzábamos en la sección introductoria de este capítulo, si bien la programación estocástica permite modelar la incertidumbre asociada, en este caso, a la demanda hospitalaria diaria, presenta el inconveniente de incrementar, a veces en gran medida, el número de variables y restricciones del modelo. Veamos este incremento en un pequeño análisis descriptivo del modelo estocástico con respecto al NL-PBA del capítulo anterior. A modo de recordatorio y como se puede ver en la Tabla 2.1, el NL-PBA está formado por $|P|(|C|+1)$ variables y $2|P| + |C| + |H_d| + 2$ restricciones. Veamos ahora qué ocurre con el modelo de programación estocástica en k etapas y considerando m escenarios diferentes.

Comencemos estudiando el nuevo número de variables. Si bien en el PBA-s se siguen empleando el mismo tipo de variables de decisión (x e y) que en los modelos del capítulo anterior, también se ha añadido un nuevo bloque de variables, z, que no estaban presentes. Además, pese a que las variables x se definen de manera similar para este modelo, están indexadas en conjuntos distintos, pues mientras que para el NL-PBA tan solo se consideraba el conjunto de camas disponibles en el instante inicial, en este modelo se tiene en cuenta el conjunto total de camas, T. Por todo ello, se puede ver que se ha pasado de $|P|(|C|+1)$ variables, donde $|C|$ representa el número de camas disponibles en el instante inicial, a un total de $\sum_{i=0}^{k-1}\sum_{j=1}^{m^i} |P^{i,j}|(|T|+1) + \sum_{i=0}^{k-1} |P^i||T|$, que es un número notablemente mayor.

Para las restricciones, a modo de evitar dificultad en la expresión, supondremos a modo de aproximación que el número de pacientes es constante y lo denotaremos por $|P|$, es decir, $|P| := |P^{i,j}|$, $\forall i \in \{1,\dots,k-1\}$, $\forall j \in \{1,\dots,m^i\}$. Pese a considerar esta aproximación, también se produce un aumento considerable en el número de restricciones, pasando de las $2|P| + |C| + |H_d| + 2$ restricciones del modelo original a un total de $\frac{1-m^k}{1-m}\left[2|P| + 2|T| + 2 + 3|H_d| + |T||P|\right] + k|P||T|$[4].

	NL-PBA	**PBA-s**
Variables	$\|P\|(\|C\|+1)$	$\sum_{i=0}^{k-1}\sum_{j=1}^{m^i} \|P^{i,j}\|(\|T\|+1) + \sum_{i=0}^{k-1} \|P^i\|\|T\|$
Restricciones	$2\|P\| + \|C\| + \|H_d\| + 2$	$\frac{1-m^k}{1-m}\left[2\|P\| + 2\|T\| + 2 + 3\|H_d\| + \|T\|\|P\|\right] + k\|P\|\|T\|$

Tabla 3.5: Número de variables y de restricciones del modelo de programación matemática original versus el modelo estocástico.

[4]La expresión $\frac{1-m^k}{1-m}$ proviene del resultado de la serie geométrica $\sum_{i=0}^{k-1} m^i$.

Pese a que tanto la expresión para el número total de variables del modelo estocástico como la del número de restricciones son relativamente complejas, no es difícil ver que se ha producido un gran aumento en ambos casos. Este aumento puede provocar problemas computacionales como los que se veían para el caso del L-PBA si se consideran horizontes muy lejanos en el tiempo (un número elevado de etapas de estudio) o si los conjuntos de pacientes tienen un tamaño considerable.

3.2.4. Comparativa para el modelo general

Para comprobar si efectivamente un aumento en el número de etapas del modelo incrementa drásticamente el tamaño del problema de optimización, aplicaremos de nuevo el modelo estocástico, pero esta vez ante un modelo en cuatro etapas y considerando tan solo dos escenarios distintos en cada etapa. Al igual que en el ejemplo de la Sección 3.2.2 tan solo tendremos en cuenta un área del hospital con intención de estudiar su posible saturación. En este caso se busca que los conjuntos de datos tengan tamaño relativamente elevado, para poder ilustrar el posible aumento en el tiempo de resolución del problema, por lo que se tomarán datos procedentes del área del hospital con más camas asignadas: el área de Medicina Interna. Esta área tiene picos que superan los 40 ingresos diarios, por lo que es esperable una gran afluencia de pacientes. Por simplicidad, se tomarán como referencia las mismas fechas de la comparativa de la Sección 3.2.2, ampliando en este caso dos días más asociados a las etapas extra consideradas en este ejemplo.

Al igual que ocurría para el área de Neumología, los primeros meses del año son de mucha actividad hospitalaria en el área de Medicina Interna, por lo que se prevé una importante tendencia al alza en el número de casos diarios. Como se avanzaba previamente, en esta comparativa se considerarán tan solo dos escenarios disponibles para cada etapa: el primero de ellos estará asociado a un incremento en la demanda de pacientes, al que se le asignará una probabilidad del 80 %, y el segundo de ellos a una estabilidad del número de ingresos, al que se le asignará una probabilidad del 20 %. Para elaborar las predicciones de la demanda de pacientes se seguirá utilizando la herramienta *Prophet*, con la variación de que, al estar considerando en este caso un número superior de etapas, se elaborarán las predicciones de manera individual para cada nodo del árbol de escenarios. En la Figura 3.6 se puede ver un diagrama de árbol que representa los distintos conjuntos de datos de pacientes en función de cada escenario posible, indicando su notación y el número de pacientes que los conforman. Es de especial importancia notar que incluso para un problema con un número de etapas no demasiado elevado y para un número muy pequeño de escenarios, el número de conjuntos de datos empieza a ser considerable. En concreto, tan solo teniendo en cuenta los conjuntos de la forma $P^{i,j}$, son necesarios 15 conjuntos, frente a los 4 del ejemplo de la Sección 3.2.2[5].

Los parámetros para este ejemplo serán los considerados a lo largo de este trabajo, manteniendo el valor de $\eta = 1$ para la constante que regula el nivel total de saturación permitido, ya que se pretende estudiar el posible colapso del área de Medicina Interna. Los resultados de aplicar el modelo estocástico a los datos de ingresos del área de Medicina Interna, empleando los cuatro *solvers* considerados a lo largo del documento, están recogidos en la Tabla 3.6. En esta tabla también se recogen los resultados asociados a resolver la situación real de manera diaria.

A la vista de la tabla se observan los primeras desventajas en el uso de la programación estocástica, frente a un enfoque diario, y es que si bien es cierto que la solución obtenida empleando el PBA-s logra una mejoría de 55 unidades en el valor de la función objetivo frente a la solución asociada al modelo original, el rendimiento del método dista de ser ideal. En primer lugar, el número de variables se ha multiplicado por un número cercano al medio millar mientras que el aumento en el número de restricciones es todavía más notable, que se multiplican por más de 13.000. Esto provoca que tres de los cuatro *solvers* empleados a lo largo de este documento para resolver problemas de este tipo no sean capaces de resolver el problema en tiempos razonables, incurriendo incluso algunos de ellos en problemas de memoria. Además, para el único *solver* que logra obtener la solución en un tiempo

[5]Además, a estos conjuntos de pacientes hay que añadir los asociados a personas de género femenino y masculino, pacientes programados, etc.

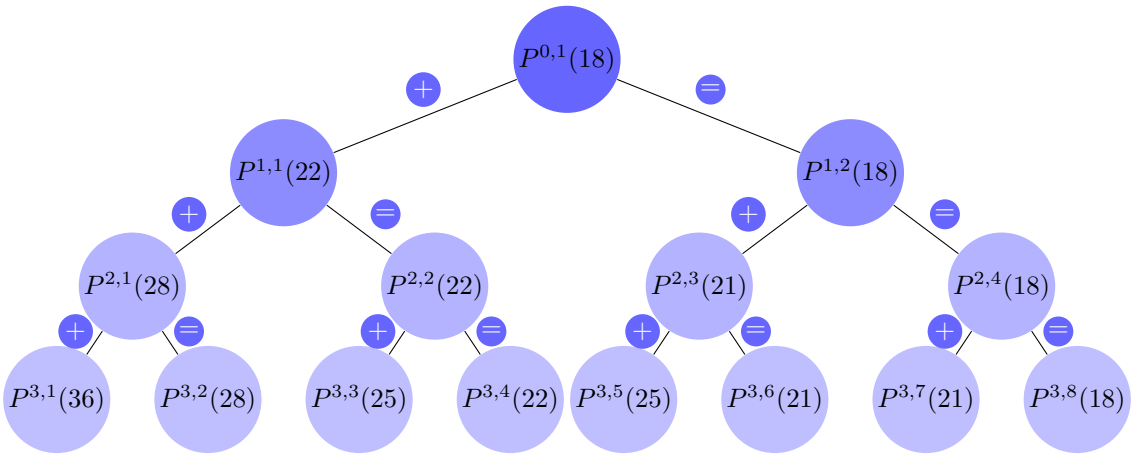


Figura 3.6: Representación en árbol de los conjuntos de pacientes del ejemplo en cuatro etapas y dos escenarios. Además de los nombres de cada conjunto se representa entre paréntesis el número de ingresos esperados para la etapa y el escenario correspondiente. Los signos "+" indican un incremento en el número de pacientes para ese escenario y los signos "=" una estabilización en el crecimiento de la demanda hospitalaria.

	NL-PBA	**PBA-s**
Variables	786	375843
Restricciones	398	5323184
Valor de la función objetivo	2240	2295 – — – — – —
Tiempo medio de resolución	1.18 – 1.09 – 4.82 – 1.98	906.36 – — – — – —
Número medio de iteraciones	19 – 0* – 33 – 0*	47 – — – — – —
Número mínimo de iteraciones	19 – 0* – 33 – 0*	47 – — – — – —
Número máximo de iteraciones	19 – 0* – 33 – 0*	47 – — – — – —

Tabla 3.6: Comparativa del número de variables y de restricciones de los modelos asociados a cada uno de los distintos enfoques, así como el valor de la función objetivo alcanzado y los tiempos de resolución empleando cuatro *solvers* diferentes: Gurobi, BARON, Couenne y OCTERACT, en el orden respectivo.Se han considerado diez instantes iniciales distintos para cada *solver*. Los valores de los parámetros del modelo empleados son: $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 1$.

razonable, Gurobi[6], las diferencias en los tiempos de ejecución son notables con respecto al enfoque diario. Todas estas aclaraciones acerca de los resultados obtenidos en esta comparativa hacen pensar

[6]Las diferencias entre Gurobi y los tres *solvers* restantes en la aplicación a este problema podría residir en el tipo de no linealidades que permite resolver Gurobi, que son muy específicas, con respecto a los demás *solvers* de problemas de programación entera no lineal que permiten no linealidades más complejas, como se puede consultar en el Apéndice B.

que el modelo estocástico así definido será muy sensible al aumento en el número de etapas y de escenarios, así como del número de pacientes. En particular, para un estudio hospitalario en el que se busque proporcionar una asignación de camas óptima, probablemente en el conjunto global del hospital, con cierta rapidez, no parece que el modelo estocástico así definido sea capaz de predecir tendencias en la demanda de pacientes a largo plazo, debido al drástico incremento en el tamaño del problema.

3.3. Simplificando el modelo estocástico general: el *expected value*

En la Tabla 3.5 se observaba que, para horizontes muy lejanos en el tiempo (muchas etapas) o para problemas en los que se considera un número elevado de escenarios en cada etapa, la resolución del modelo estocástico aparece como un desafío computacional demasiado significativo. Esto provoca que el beneficio asociado al modelado de la incertidumbre pueda no ser tal, debido a un incremento en el tiempo de resolución del problema. La idea en esta sección es proporcionar una solución que mantenga las propiedades de mejora en el valor de la función objetivo asociadas al modelado de la incertidumbre del número de ingresos, pero que no presente un incremento demasiado significativo en el número de restricciones.

Bajo esta idea nace un nuevo enfoque bastante común en problemas de programación estocástica: el *expected value* (EV) o valor esperado. La idea, como su nombre indica, es buscar el valor esperado de la variable que modela el número de ingresos diarios. Una vez se tenga este valor se colapsará el árbol que contiene a los distintos escenarios posibles con sus probabilidades asociadas, obteniendo un único escenario para cada etapa, en el que la demanda será el valor esperado de la demanda promediando las demandas en cada escenario con sus correspondientes probabilidades. Usando la formulación general del problema de programación estocástica sobre el diagrama colapsado, la solución que el *expected value* proporciona puede interpretarse como aproximación de la solución del problema de partida.

Para el caso en el que se tenga un problema con dos etapas y tres escenarios como el de la Sección 3.2.1, el esquema del problema de programación estocástica se puede ver en la Figura 3.7.

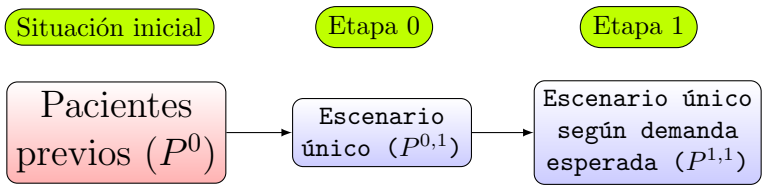


Figura 3.7: Esquema del problema de programación estocástica en dos etapas y tres escenarios empleando el expected value.

La idea detrás del colapso en las ramas del árbol busca simplificar la información de entrada al problema y reducir el número de conjuntos de datos, así como el número de variables y restricciones del problema, facilitando la resolución del mismo. La desventaja que tiene este tipo de solución es la pérdida de información con respecto al modelo de programación estocástica inicial, ya que el valor esperado no es tan representativo sobre la variable que regula el número de ingresos diarios como la propia distribución de la variable. Sin embargo, para situaciones en las que se presuponga una eficacia casi total del modelo predictivo, el *expected value* podría ser una solución muy interesante otorgándole todo el peso al escenario asociado a la predicción de la demanda hospitalaria.

Veamos ahora las propiedades de esta nueva solución sobre los ejemplos considerado en las Secciones 3.2.2 y 3.2.4. Comencemos por la comparativa para el modelo en dos etapas y tres escenarios. En ella se consideraba un primer escenario en el que la demanda de pacientes era de 9 con una probabilidad

del 80 %; un segundo escenario en el que el número de pacientes a ingresar se mantenía en 4, con una probabilidad del 15 % ; y, finalmente, un tercer escenario en el que la demanda de ingresos descendía hasta un total de 3 pacientes, con probabilidad del 5 %. Teniendo en cuenta estos datos, la demanda esperada de pacientes para el día 13 de enero es de:

$$EV(\text{Demanda de pacientes}) = 0.8 \cdot 9 + 0.15 \cdot 4 + 0.05 \cdot 3 = 7.95 \approx 8 \text{ pacientes.}$$

Teniendo en cuenta esta demanda esperada, el problema resultante para el caso de que se consideren dos etapas y los tres escenarios recogidos en la Tabla 3.3 es el siguiente:

$$\begin{aligned}
\text{maximizar} \quad & \sum_{p\in P^{0,1}} y_p^{0,1} d_D + \sum_{p\in P^{0,1}} \sum_{t\in T} x_{pt}^{0,1} RiesgoPac_p^{0,1} b_N + \sum_{p\in P_p^{0,1}} \sum_{t\in T} x_{pt}^{0,1} b_P \\
& + \sum_{p\in P^{1,1}} y_p^{1,1} d_D + \sum_{p\in P^{1,1}} \sum_{t\in T} x_{pt}^{1,1} RiesgoPac_p^{1,1} b_N + \sum_{p\in P_p^{1,1}} \sum_{t\in T} x_{pt}^{1,1} b_P
\end{aligned}$$

$$\begin{aligned}
\text{sujeto a} \quad & \sum_{t\in T} x_{pt}^{i,1} \leq 1,\ \forall i \in \{0,1\},\ \forall p \in P^{i,1} \\
& \sum_{p\in P^{i,1}} x_{pt}^{i,1} \leq 1,\ \forall i \in \{0,1\},\ \forall t \in T \\
& \frac{1}{|T|}\left(\sum_{p\in P^{i,1}} \sum_{t\in T} x_{pt}^{i,1} + \sum_{p'\in P^0} \sum_{t\in T} z_{p't}^{i} \right) \leq \eta,\ \forall i \in \{0,1\} \\
& y_p^{i,1} - \sum_{t\in T} \sum_{h\in H} \sum_{d\in D} x_{pt}^{i,1} Ubi_{th} DepHab_{hd} DepOpt_{pd}^{i,1} = 0,\ \forall i \in \{0,1\},\ \forall p \in P^{i,1} \\
& \sum_{p\in P_p^{i,1}} \sum_{t\in T} x_{pt}^{i,1} \leq |P_p^{i,1}|,\ \forall i \in \{0,1\} \\
& \left(\sum_{p\in P_f^{i,1}} \sum_{t\in T} x_{pt}^{i,1} Ubi_{th} \right)\left(\sum_{p\in P_m^{i,1}} \sum_{t\in T} x_{pt}^{i,1} Ubi_{th} \right) = 0,\ \forall i \in \{0,1\},\ \forall h \in H_d \\
& \left(\sum_{p\in P_f^{i}} \sum_{t\in T} z_{pt}^{i} Ubi_{th} \right)\left(\sum_{p\in P_m^{i,1}} \sum_{t\in T} x_{pt}^{i,1} Ubi_{th} \right) = 0,\ \forall i \in \{0,1\},\ \forall h \in H_d \\
& \left(\sum_{p\in P_f^{i,1}} \sum_{t\in T} x_{pt}^{i,1} Ubi_{th} \right)\left(\sum_{p\in P_m^{i}} \sum_{t\in T} z_{pt}^{i} Ubi_{th} \right) = 0,\ \forall i \in \{0,1\},\ \forall h \in H_d \\
& z_{pt}^0 + z_{pt}^1 = X_{pt}^0 \,\text{mín}\{F_p^0, 2\},\ \forall p \in P^0,\ \forall t \in T \\
& \sum_{l=i+2}^{k} z_{pt}^{l-1} = x_{pt}^{i,1} \,\text{mín}\{F_p^{i,1} - 1, k-i-1\},\ \forall i \in \{0,1\},\ \forall p \in P^{i,1},\ \forall t \in T \\
& z_{pt}^0 \geq z_{pt}^1,\ \forall p \in P^0,\ \forall t \in T \\
& z_{pt}^1 - x_{pt}^{0,1} \leq 0,\ \forall p \in P^{0,1},\ \forall t \in T \\
& \left(\sum_{p\in P^{i}} z_{pt}^{i} \right)\left(\sum_{p'\in P^{i,1}} x_{pt}^{i,1} \right) = 0,\ \forall i \in \{0,1\},\ \forall t \in T
\end{aligned}$$

Para este problema cada conjunto, parámetro o variable de la forma $A^{1,1}$ hace referencia al escenario asociado al valor de demanda esperado para la segunda etapa. El resto del problema de optimización

es idéntico al de la Sección 3.2.1. Además, es sencillo ver que el número de variables y de restricciones ha disminuido significativamente, pues el segundo de los dos superíndices permanece fijado al tratar con un único escenario en cada etapa.

Para estudiar el comportamiento de este nuevo modelo, se aplicará al mismo conjunto de datos del ejemplo de la Sección 3.2.2 para comparar los valores de la función objetivo óptimos que proporciona cada uno de los tres modelos: NL-PBA, PBA-s y *expected value*. Además, también se realizará una comparativa entre los números de restricciones y de variables de cada modelo, así como de los tiempos medios de resolución empleando distintos *solvers*. Como las iteraciones de los *solvers* no eran excesivamente elevadas en ninguno de los ejemplos considerados, se omitirán en esta comparativa.

	NL-PBA	**PBA-s**	**Expected value**
Variables	104	454	302
Restricciones	67	330	152
Función objetivo	1230	1257.5	1240
Tiempo Baron	0.11	0.22	0.14
Tiempo Couenne	0.04	2.65	0.43
Tiempo Octeract	0.03	0.03	0.03
Tiempo Gurobi	0.08	0.03	0.03

Tabla 3.7: Comparativa del número de variables y de restricciones de tres modelos asociados a cada uno de los distintos enfoques, así como el valor de la función objetivo alcanzado y los tiempos de resolución empleando cuatro *solvers* diferentes: Baron, Couenne, Octeract y Gurobi. Los valores de los parámetros del modelo empleados son: $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 1$.

En la Tabla 3.7 se recogen los resultados de realizar la comparativa de la Sección 3.2.2 añadiendo la solución del *expected value*. Como se podía preveer, el valor de la función objetivo ha disminuido con respecto al asociado al modelo estocástico. Sin embargo, el valor es ligeramente superior al que se obtiene mediante el estudio diario. Más concretamente, este incremento en 10 unidades del valor de la función objetivo se debe a que la solución del *expected value* propone ingresar a un paciente el segundo día con una unidad más de riesgo que un paciente que ingresa el primer día al resolver el NL-PBA. Esto implica que, pese a que la diferencia en la función objetivo no parece excesivamente elevada, sí es significativa, ya que se logra ingresar a un paciente con cuadro más grave del que se ingresaría con una perspectiva diaria. Además, si bien el número de variables y de restricciones de este nuevo modelo son superiores a las del NL-PBA, el incremento es bastante menos significativo que el que tenía lugar para el PBA-s. Esto se ve ligeramente reflejado en los tiempos medios de ejecución con cada *solver*, que son algo inferiores a los del modelo estocástico, pese a que estas diferencias no son demasiado significativas debido a que todos los tiempos son relativamente pequeños.

Veamos ahora qué ocurre para el caso de la Sección 3.2.4 en el que se consideraban cuatro etapas y dos escenarios. De manera análoga al caso anterior, se calcula el valor esperado de pacientes a ingresar en cada etapa.

$$
\begin{aligned}
EV(13/01/2020) &= 0.8 \cdot 22 + 0.2 \cdot 18 = 21.2 \approx 21 \text{ pacientes.} \\
EV(14/01/2020) &= 0.8 \cdot (0.8 \cdot 28 + 0.2 \cdot 22) + 0.2 \cdot (0.8 \cdot 21 + 0.2 \cdot 18) = 25.52 \approx 26 \text{ pacientes.} \\
EV(15/01/2020) &= 0.8 \cdot (0.8 \cdot (0.8 \cdot 36 + 0.2 \cdot 28) + 0.2 \cdot (0.8 \cdot 25 + 0.2 \cdot 22)) \\
&\quad + 0.2 \cdot (0.8 \cdot (0.8 \cdot 25 + 0.2 \cdot 21)) + 0.2 \cdot (0.8 \cdot 21 + 0.2 \cdot 18)) \\
&= 33.87 \approx 34 \text{ pacientes.}
\end{aligned}
$$

Para este caso, se omitirá la definición del nuevo modelo por ser en esencia análoga a la del ejemplo anterior pero considerando en este caso un número superior de etapas.

Finalmente, en la Tabla 3.8 se presentan los resultados empleando el *solver* Gurobi para diez instancias del problema comparándolos con los obtenidos empleando el NL-PBA y el PBA-s.

	NL-PBA	**PBA-s**	**Expected value**
Variables	786	375843	2046
Restricciones	398	5323184	4398
Función objetivo	2240	2295	2265
Tiempo medio de resolución	1.18	906.36	6.78

Tabla 3.8: Comparativa del número de variables y de restricciones de tres modelos asociados a cada uno de los distintos enfoques, así como el valor de la función objetivo alcanzado y los tiempos medios de resolución empleando el *solver* Gurobi. Los valores de los parámetros del modelo empleados son: $b_D = 75$, $b_N = 10$, $b_P = 30$ y $\eta = 1$.

A la vista de la tabla parece claro que, si bien la solución obtenida empleando el *expected value* pierde precisión en el modelado de la incertidumbre en la demanda de pacientes, se logra un incremento en el valor de la función objetivo con respecto al enfoque diario y se mantienen tiempos de ejecución pequeños (de escasos segundos). Por ello, y a la vista de los dos ejemplos trabajados en este capítulo, el *expected value* se presenta como una solución alternativa al PBA que logra reunir en cierto modo las ventajas del PBA-s pero evitando un aumento demasiado elevado en el tamaño del problema.

Capítulo 4

Métodos heurísticos para el PBA

La dificultad en la resolución de problemas de programación matemática, bien por la complejidad del modelo o bien por el excesivo volumen de información involucrada, hace necesaria la utilización de procedimientos alternativos como los descritos en el Capítulo 1. Los métodos heurísticos tienen como propósito principal abordar los problemas de programación matemática de forma más simple y dejando a veces la resolución analítica del problema en un segundo plano. Estos métodos no garantizan alcanzar el óptimo exacto del problema, sino que proponen una solución, generalmente bastante próxima a la solución real del problema, en una cantidad de tiempo muy inferior a la requerida en la resolución del problema original.

La idea de este capítulo será presentar métodos heurísticos para resolver los problemas PBA y analizar sus ventajas y desventajas con respecto a los métodos clásicos de resolución presentados en los capítulos anteriores. El resto del capítulo se estructura de la siguiente manera. En la Sección 4.1 se introduce un primer método heurístico basado en el tiempo computacional de resolución de los problemas y se realiza una comparativa de casos en los que la heurística es eficaz y en los que no. Por otro lado, en la Sección 4.2 se describe un algoritmo que se aproxima bastante a la organización cotidiana de un hospital pero desde un punto de vista matemático. También se ilustrará la eficacia del algoritmo con ejemplos.

4.1. Restricción del tiempo computacional

A menudo, en problemas de optimización entera como es el caso del PBA, e incluso para situaciones en las que no es posible encuadrar al problema dentro de la programación lineal entera, los métodos de obtención de óptimos globales recurren internamente al uso de algoritmos de *branch and bound*. Como se puede ver en el primer capítulo de este documento, la idea detrás de estos algoritmos de ramificación y acotación es ir seleccionando y descartando ramas del árbol buscando una mejora progresiva en la función objetivo. Este descarte de las ramas que no ayudan a mejorar el valor de la función provoca que una gran parte del tiempo computacional de resolución de estos problemas empleando algoritmos de *branch and bound* se emplee en rechazar aquellas ramas que no interesan. Es por ello que, pese a que a menudo los algoritmos de resolución de problemas de programación entera logran obtener el valor óptimo para dicho problema en un tiempo no excesivamente elevado, la tarea de confirmar que el valor obtenido es, en efecto, óptimo, es tediosa y puede ser necesario una gran cantidad de tiempo para llevarla a cabo.

Por todo ello, un primer método heurístico muy sencillo pero que busca explotar las propiedades de los problemas de programación entera como el PBA reside en aplicar un método clásico como los estudiados en capítulos anteriores pero restringiendo el tiempo máximo de resolución permitido al *solver* correspondiente. La idea detrás de este método es suponer que el *solver* llegará en menos de ese tiempo al valor óptimo del problema (o al menos a un valor no demasiado alejado), esperando que

el *solver* emplee la gran mayoría del tiempo de resolución del problema en el proceso de descarte de ramas y no en la obtención del valor óptimo.

Sin embargo, más allá de las suposiciones anteriores sobre el comportamiento del *solver*, este método heurístico presenta algunos inconveniente que hacen que a menudo sea de poca utilidad. Uno de ellos, y quizás el más importante reside en que en situaciones en las que se tiene un número elevado de restricciones y variables y los conjuntos de datos tienen un tamaño considerable, el *solver* empleará gran parte del tiempo máximo de ejecución permitido en la propia lectura de datos, lo que puede provocar un grave empeoramiento en la calidad de la solución propuesta.

Resolviendo el PBA determinístico

Un caso extremo de estas situaciones que se menciona se corresponde con el ejemplo considerado en la Sección 2.3.3. Según la Tabla 4.1, que incluye información comparativa de la resolución del problema, el *solver* no lograba ni siquiera proporcionar una solución factible en una hora de tiempo límite para el modelo linealizado. Este empleaba todo el tiempo en la lectura de los datos. Esta lentitud del *solver* para resolver este problema hace que restringir el tiempo límite de cálculo del *solver* no produzca ninguna mejora en la resolución del problema, ya que seguiría sin obtenerse ni siquiera una solución factible.

	NL-PBA	**L-PBA**
Valor de la función objetivo	6210	————
Tiempo medio de resolución	14.22	————
Variables	32760	32760
Restricciones	813	55317412680
Iteraciones promedio	143.9	————
Iteraciones mínimas	143	————
Iteraciones máximas	144	————

Tabla 4.1: Información sobre la resolución del ejemplo de la Sección 2.3.3.

Resolviendo el problema PBA estocástico

Empleemos ahora este método sobre el problema de programación estocástica. En concreto, se estudiará qué ocurre con los valores de la función objetivo para el ejemplo de la Sección 3.2.4, asociado al área de Medicina Interna. En este ejemplo, y considerando el modelo estocástico, Gurobi necesitaba más de 15 minutos para obtener la solución óptima. En este caso se buscará limitar el tiempo máximo de ejecución a 5 y 10 minutos para comparar los resultados con los obtenidos sin límite de tiempo. Los resultados obtenidos están recogidos en la Tabla 4.2.

A la vista de la tabla se observa que en las diez instancias del problema para las que se ha limitado el tiempo máximo de ejecución del *solver* a 10 minutos se obtiene la solución óptima, por lo que el *solver* emplearía los 5 minutos posteriores en confirmar que esta solución es, en efecto, óptima. Sin embargo, para los casos en los que se considera un tiempo límite de 5 minutos el valor de la solución objetivo obtenido dista mucho del valor óptimo.

	Modelo estocástico	**10 minutos**	**5 minutos**
Variables	375843	375843	375843
Restricciones	5323184	5323184	5323184
Función objetivo promedio	2295	2295	1090.8
Tiempo de resolución promedio	906.36	600	300

Tabla 4.2: Comparativa de valores específicos del problema asociado en función del tiempo límite de ejecución permitido para el *solver* Gurobi. Para cada versión se han considerado 10 instancias distintas con diferentes instancias iniciales.

Este ejemplo ilustra perfectamente las ventajas y los inconvenientes de este algoritmo. En primer lugar, con una buena selección del tiempo límite proporciona soluciones muy cercanas a la óptima, o incluso la propia solución óptima. Sin embargo, a priori no se sabe qué tiempo límite será el adecuado, y para una elección deficiente del tiempo límite, la solución obtenida puede estar muy alejada del óptimo.

Otro de estos inconvenientes, que de hecho es común en muchos de los métodos heurísticos es que, si bien al aplicar el algoritmo a un caso concreto es difícil que se produzca un caso patológico en el que el valor propuesto por el método diste mucho del óptimo real del problema, a lo largo del tiempo, la probabilidad de que esta situación tenga lugar aumenta. A veces, dependiendo del contexto del problema de optimización correspondiente esta circunstancia puede no llegar a ser muy preocupante. Sin embargo, en el contexto hospitalario en el que nace el PBA, una asignación subóptima de pacientes a camas en un cierto día puede tener graves consecuencias en la calidad de los servicios sanitarios proporcionados, así como un posible incremento en las listas de espera.

4.2. Método basado en el árbol de decisión

Otro algoritmo heurístico que se plantea para resolución del PBA está basado en los árboles de decisión. La idea de un método de este estilo es diseñar, en función de las necesidades del hospital, una serie de preguntas encadenadas con respuesta dicotómica (principalmente “Sí” y “No”). En función de cada una de estas respuestas, se van recorriendo las distintas ramas del árbol para finalizar en un nodo terminal que determinará una cierta acción sobre cada paciente.

Una de las principales ventajas que presenta un algoritmo de este estilo es que permite resolver problemas de PBA online, es decir, problemas en los que el conjunto de pacientes que pretenden ingresar en el hospital es dinámico, donde estos llegan al hospital de manera progresiva (como en un servicio de urgencias). De hecho, un modelo genérico como el que se presenta en la Figura 4.1 no dista mucho del procedimiento llevado a cabo en los hospitales en la actualidad. Además, no sería difícil introducir el componente matemático en un árbol de decisión como el mostrado en la Figura 4.1, ya que bastaría con tener en cuenta los niveles de riesgo considerados en los modelos de los capítulos anteriores para cada paciente, y considerar como criterio la elección de los pacientes que menos penalizan el valor de la función objetivo en el modelo de programación matemática. Por otro lado, la implementación de un modelo de este estilo sería relativamente sencilla, ya que tan solo serían necesarias una secuencia de condiciones lógicas anidadas. En concreto, el código implementado en lenguaje R asociado al árbol de decisión de la figura se puede encontrar en el Apéndice C.

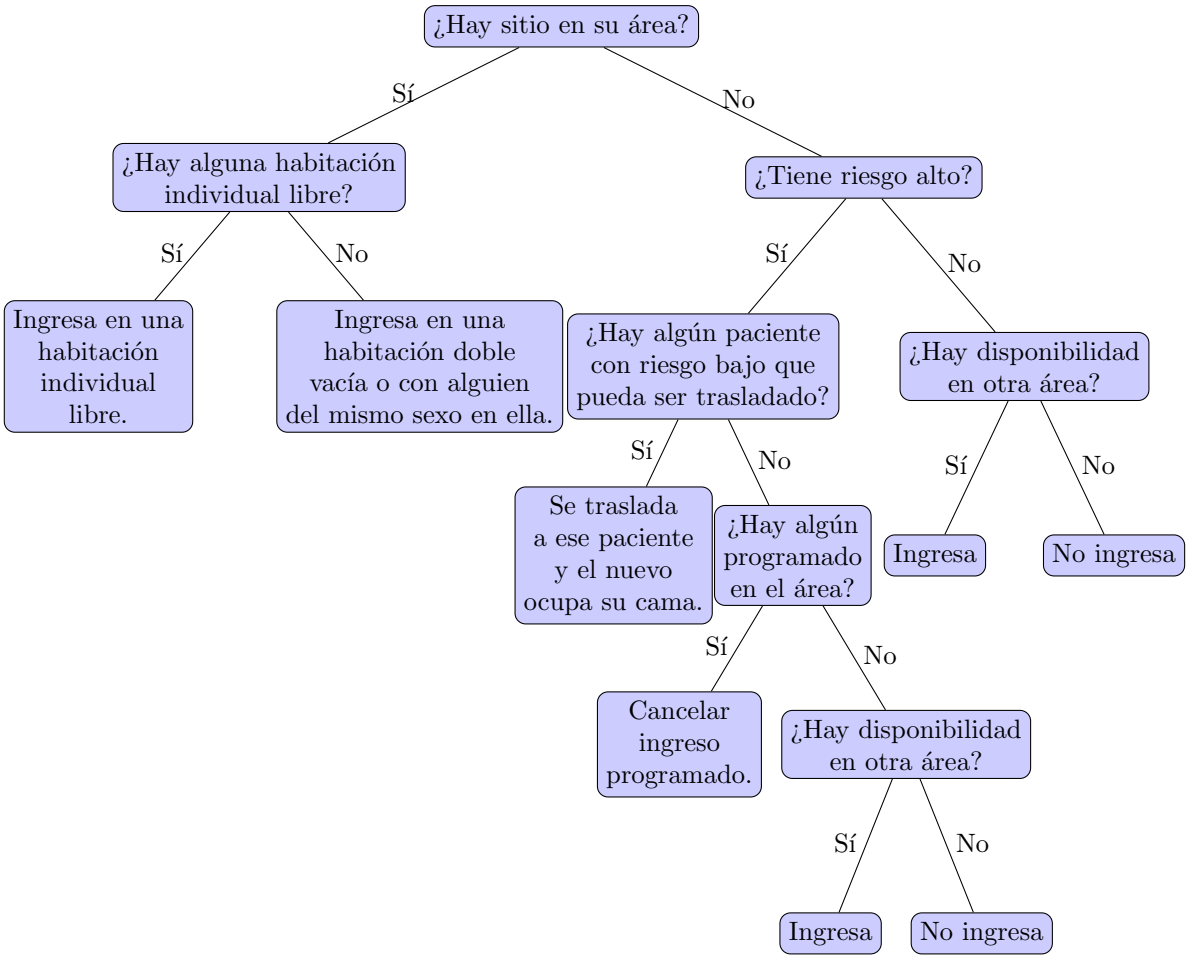


Figura 4.1: Ejemplo de un algoritmo de árbol de decisión para resolver el PBA.

4.2.1. Situación favorable para la heurística

Para evaluar el comportamiento del algoritmo, se aplicará al mismo conjunto de datos del ejemplo de la Sección 2.3.3 para la situación global del hospital, realizando en este caso 50 reordenaciones del vector de pacientes[1]. Como en este caso la comparativa entre número de variables y de restricciones no tiene mucho sentido, se compararán los valores obtenidos de la función objetivo y los tiempos promedio de resolución. Para el método heurístico se tendrán en cuenta las soluciones obtenidas con las 50 permutaciones del vector de pacientes y para el NL-PBA se recuperarán los datos de la Tabla 2.4.

	NL-PBA	**Árbol de Decisión**
Valor de la función objetivo	9335	9335
Tiempo medio de resolución	14.22	0.02

Tabla 4.3: Valor promedio de la función objetivo óptimo alcanzado resolviendo el NL-PBA y el árbol de decisión, así como los tiempos medios de resolución. Para ambos modelos se han considerado los parámetros $\eta = 0.85$, $BD = 75$, $BN = 10$ y $BP = 30$.

A la vista de la pequeña comparativa de la Tabla 4.3 se puede observar que el método heurístico logra obtener el valor óptimo de la solución independientemente de la ordenación del vector de pacientes considerada. Además lo logra en un tiempo notablemente inferior al modelo original. Este es un claro ejemplo donde el uso de un modelo heurístico logra reducir notablemente el cálculo computacional y no solo proporciona una solución buena, sino que además es la óptima.

4.2.2. Situación desfavorable para la heurística

Sin embargo, al igual que para el método heurístico de la sección anterior, también existen inconvenientes en el uso de un árbol de decisión para resolver el PBA. En primer lugar, la consideración de posibles tendencias en el número de ingresos en fechas posteriores no es tan "directa" como ocurría para los modelos generales, en los que se empleaba la programación estocástica. Esto hace que no sea posible tener en cuenta situaciones cambiantes a lo largo del tiempo, como un brote de una enfermedad infecciosa, al menos de una forma directa. Por otro lado, si bien un algoritmo como el descrito en 4.1 parece regular bien la situación hospitalaria, tampoco tiene en cuenta factores claves como el número esperado de pacientes a ingresar en un mismo día, provocando que días en los que se espere una gran afluencia de pacientes, pueda llegar a colapsarse el hospital incluso sin llegar a ingresar a los pacientes más graves y provocando una asignación de pacientes subóptima. Para la situación global del hospital considerada esto no ocurría, pues el porcentaje global de ocupación distaba mucho del umbral prefijado del 85 %. Veamos qué ocurre en un caso en el que la ocupación hospitalaria sea más elevada. En concreto, elegiremos el día 14 de enero de 2020, en el cual el porcentaje de ocupación del hospital está cercano al 85 %. Además, ese mismo día pretenden ingresar 149 pacientes, por lo que se prevé que se pueda alcanzar el umbral de saturación del hospital prefijado. En la Tabla 4.4 se recogen los valores óptimos obtenidos tanto por el NL-PBA como por el método heurístico, así como los tiempos medios de ejecución obtenidos de igual manera al ejemplo anterior.

En este caso, si bien nuevamente el tiempo de resolución empleando el método heurístico es bastante inferior al asociado al NL-PBA, la diferencia entre las soluciones proporcionadas por ambos métodos es notable. Esto se debe a que, ante una gran presión hospitalaria, el método heurístico no logra tener

[1] Es importante notar que la solución obtenida por este método heurístico puede depender del orden de los pacientes que ingresan en el hospital. Como los datos de las horas de llegada de los pacientes son desconocidas, se realizarán permutaciones del conjunto de pacientes y se analizarán las soluciones obtenidas.

	NL-PBA	**Árbol de Decisión**
Valor de la función objetivo	16905	16820.8
Tiempo medio de resolución	17.94	0.23

Tabla 4.4: Valor promedio de la función objetivo óptimo alcanzado resolviendo el NL-PBA y el árbol de decisión, así como los tiempos medios de resolución. Para ambos modelos se han considerado los parámetros $\eta = 0.85$, $BD = 75$, $BN = 10$ y $BP = 30$.

una visión más global que evite que entren pacientes que realmente no tienen una patología tan grave como los que llegarán posteriormente. Este es un claro ejemplo en el cual el uso de un método heurístico puede producir una asignación subóptima, provocando un empeoramiento en la calidad de la atención médica.

Nuevamente, al igual que ocurría para el método de la sección previa, este método heurístico presenta unas ciertas propiedades positivas realmente interesantes, pero también posee desventajas derivadas de la posibilidad de obtener un valor lejano al óptimo, así como de la dificultad para modelar la incertidumbre en el número de pacientes.

Capítulo 5

Conclusiones

La asignación eficiente de camas hospitalarias se ha convertido en un problema usual en el día a día de la gestión hospitalaria. Este problema puede ser abordado desde diferentes perspectivas, como la económica, minimizando el gasto hospitalario, la social, garantizando que todos sus usuarios tengan la posibilidad de ser hospitalizados, y desde el punto de vista sanitario, que garantice la hospitalización en el servicio que más se adecúe a la patología de cada uno de los pacientes.

A lo largo de este documento se han presentado distintas formas de resolver el PBA desde un enfoque matemático y más concretamente, de la investigación operativa. En concreto, hemos considerado una base de datos real, que contiene información de 215171 pacientes hospitalizados en el Complejo Hospitalario Universitario de Santiago de Compostela (CHUS), para la validación y evaluación de las propuestas descritas. De forma genérica, en este trabajo hemos alcanzando, entre otros, los siguientes objetivos:

- Hemos analizado desde un punto de vista descriptivo la base de datos proporcionada con el objetivo de identificar las áreas de hospitalización que presentan un índice de ocupación más crítico.
- Esa información ha sido utilizada con el objetivo de validar la primera propuesta de problema de programación matemática que optimice la eficiencia del sistema hospitalario. Aquí se han desarrollado varias tareas, desde la modelización del mismo, su implementación en código AMPL que permitió su resolución, y la validación de los resultados sobre la base de datos real originalmente proporcionada.
- El modelo original ha permitido considerar metodologías específicas para la planificación y la gestión hospitalaria en escenarios a largo plazo. Esta visión global proporciona mucha más información que la primera opción, aunque la complejidad computacional asociada se presenta como su principal hándicap.
- Por último, hemos presentado alternativas que proporcionen soluciones inmediatas aunque sin garantías de optimalidad.

A continuación, se detallan los objetivos alcanzados. Pese a ser un problema difícil de tratar, se ha propuesto un modelo inicial conceptualmente sencillo de programación entera para asignar de manera diaria pacientes a las camas del hospital. También se presentó una versión del modelo que se encuadra dentro de la programación lineal entera, pero que presentaba la desventaja de incrementar en exceso el tamaño del problema y provocando que su resolución fuese inviable para conjuntos de pacientes relativamente grandes. Por ello, el modelo original de programación entera no lineal se concluye que es un buen método para intentar resolver el PBA desde un punto de vista diario.

Sin embargo, las situaciones en un hospital tienden a ser más complejas y surge la necesidad de considerar una visión más a largo plazo y que permita tener en cuenta posibles tendencias en el número

de ingresos diarios. En ese contexto se define el modelo estocástico que, apoyado en una predicción del número de ingresos, busca modelar la incertidumbre asociada al mismo. Las conclusiones extraídas de las comparativas realizadas en esta sección confirman que una visión más global tiende a favorecer una mejor asignación de pacientes a las camas. Sin embargo, esta mejoría en el valor de la función objetivo se ve enturbiada por el aumento significativo del tamaño del problema cuando el número de etapas y de escenarios considerados aumenta. Para intentar abordar esta desventaja del modelo estocástico se define una nueva solución, el *expected value*, que busca limitar las desventajas computacionales que surgen bajo este enfoque. Además, en los casos reales considerados se puede ver que la mejora en términos de dificultad computacional es notable, lo que probablemente compense un ligero empeoramiento en la calidad del valor de la función objetivo en el óptimo. Por todo ello, se puede concluir que el *expected value*, que no es más que una ligera modificación del modelo estocástico, es un método muy interesante para dar solución al PBA.

Con el objetivo de reducir los esfuerzos computacionales en la práctica, donde los servicios de hospitalización correspondientes buscan herramientas rápidas y eficientes en la toma de decisiones, se introducen dos métodos heurísticos. Estos buscan proporcionar soluciones al problema original aunque sin garantías de optimalidad. En general, si bien presentan buenas propiedades en términos de la complejidad computacional medida en términos del tiempo en proporcionar soluciones, la primera propuesta está algo más restringida y su rendimiento puede verse muy limitado ante ligeras modificaciones en los conjuntos de datos. Sin embargo, el método basado en el árbol de decisión parece que proporciona resultados positivos.

A modo de resumen, el mejor modelo que se presenta en este trabajo para dar solución al PBA es el modelo de programación estocástica introduciendo la versión que incluye la solución del *expected value*, ya que logra reunir las ventajas de los modelos de programación matemática generales (tamaño del problema relativamente pequeño) y las del modelo estocástico (incorporación de la incertidumbre en el número de pacientes). Además, para casos en los que los tamaños de estos problemas sean muy elevados, los métodos heurísticos pueden suponer una herramienta muy útil para proporcionar soluciones suficientemente buenas que permitan ofrecer una asignación eficiente de pacientes a las camas del hospital.

Como trabajo futuro, debe abordarse el empleo de técnicas específicas de resolución de problemas de programación matemática que reduzcan la complejidad de resolución aparente asociada al tamaño del problema que resulta. En ese contexto, las técnicas de descomposición se presentan como herramientas útiles para reducir esa dificultad, y concretamente las descomposiciones Lagrangiana (Ver Geoffrion (1974) y Fisher (1981)) y la generalizada de Benders (ver Geoffrion (1972) y Rahmaniani et al. (2017)).

Apéndice A

Descripción de la base de datos

Este apéndice proporciona una descripción detallada de la base de datos de pacientes proporcionada por el FIDIS y que ha sido empleada tanto en la motivación como en los ejemplos de este trabajo. Además, esta base de datos es de especial importancia ya que la información que contiene ha sido la base fundamental en la elaboración de modelos de optimización cuyo objetivo es la asignación óptima de pacientes a camas del hospital.

La base de datos contiene una gran cantidad de información relevante acerca de los pacientes, no solo en relación a las patologías de cada paciente y a sus características físicas, sino también a datos demográficos, fechas de ingreso y de alta, tipo de ingreso, etc. Esta cantidad de información permite tener una visión global del conjunto de pacientes y facilita el diseño de herramientas de toma de decisiones en la asignación de camas. A continuación, se describirán las variables recogidas en la base de datos, haciendo especial énfasis en aquellas de especial relevancia para la elaboración de los modelos.

A.1. Descripción global de la base de datos

La base de datos recoge información de 215171 pacientes ingresados en el Complejo Hospitalario Universitario de Santiago de Compostela (CHUS) entre el día 01/01/2016 y el día 18/11/2021. Si bien la base de datos proporcionada por el hospital se ocupa de recoger las características de cada paciente, es importante subrayar el hecho de que los datos se presentan de manera anónima, sin distintivos que permitan identificar al paciente, con el propósito de cumplir con las leyes de protección de datos.

Cabe destacar también que la recopilación de los datos a lo largo del tiempo permiten el estudio de tendencias en la ocupación del hospital o del número de ingresos. Este hecho es de gran valor de cara a posibles predicciones de escenarios futuros.

A.2. Análisis de las variables

En esta sección se introducirá brevemente cada una de las variables y se realizará un breve análisis descriptivo de las variables que conforman la base de datos, en especial de las que son de utilidad para la construcción de los modelos.

- **Ano**: Variable cuantitativa discreta que representa el año de ingreso del paciente. Como los datos están tomados entre los años 2016 y 2020, esta variable tomará los mismos valores.
- **Cód. diag. ppal.**: Variable cuantitativa que representa el código alfanumérico asociado al diagnóstico principal del paciente. Su utilidad reside en el sistema de organización interna del hospital, pero no será empleado en la elaboración de los modelos del PBA de este trabajo.

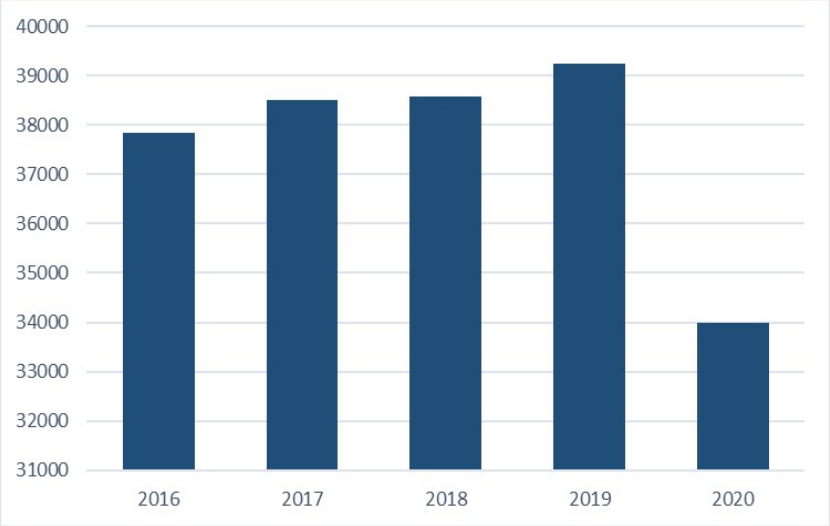


Figura A.1: Número total de ingresos anuales en el Hospital Clínico Universitario de Santiago de Compostela entre los años 2016 y 2020.

- **Diagnóstico principal**: Variable cualitativa que describe el diagnóstico principal de cada paciente ingresado. Es de especial utilidad para identificar la gravedad en la condición de un paciente.
- **Tipo ingreso**: Variable cualitativa que recoge el tipo de ingreso de cada paciente. Los más comunes son los ingresos de urgencias, ya sean procedentes del servicio de urgencias o de consultas externas, y programados. Es de especial utilidad de cara a la elaboración de los modelos pues permite identificar a los pacientes programados.

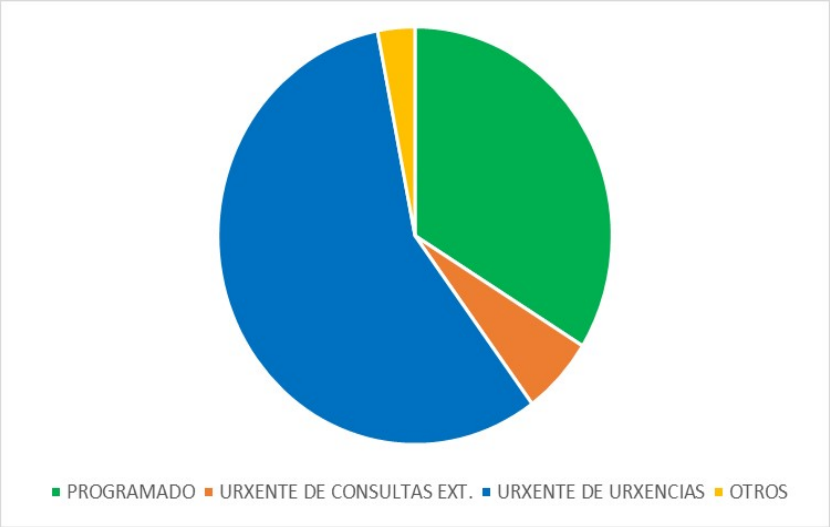


Figura A.2: Proporción de pacientes según el tipo de ingreso.

- **Cód. tipo ing.**: Variable cuantitativa que asocia un código a cada paciente en función del tipo de ingreso. Al igual que el código del diagnóstico principal, no será de mucha utilidad en la

elaboración de los modelos.

- **Data ingreso**: Variable cuantitativa discreta que recoge la fecha en la que ingresa cada paciente. Permite estudiar el número de ingresos diarios y, combinada con la variable *Data de alta* permite conocer el número de pacientes que hay ingresados en el hospital en un determinado día.
- **Motivo alta**: Variable cualitativa que recoge el motivo por el cual se da de alta a un paciente. El motivo de alta más común está asociado a que el paciente pueda regresar a su domicilio, seguido por los fallecimientos, exitus, con una proporción mucho menor. No se ha tenido en cuenta esta variable para la elaboración de los modelos, pero contiene información relevante que podría ser de utilidad para predecir la duración de estancia esperada de un paciente.

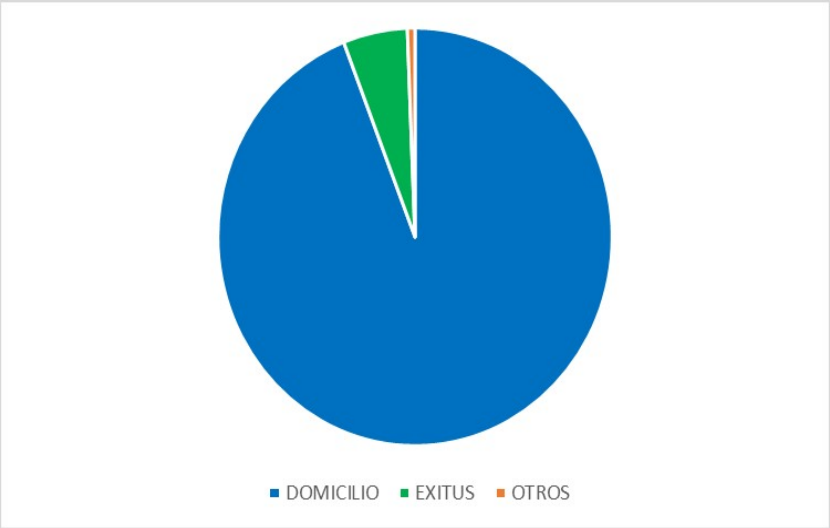


Figura A.3: Proporción de pacientes en función del motivo de alta.

- **Cód. mot. alta**: Al igual que para las variables *Cód. diag. ppal.* y *Cód. tipo ing.* es una variable cuantitativa que asocia cada motivo de alta con un código interno del hospital. De nuevo no será de utilidad en la elaboración de los modelos.
- **Data de alta**: Variable cuantitativa discreta que recoge la fecha en la que recibe el alta cada paciente. Junto con la variable *Data de alta* permite conocer el número de pacientes que hay ingresados en el hospital en un determinado día.
- **GFH alta**: Variable cualitativa que representa el subárea hospitalaria a la que fue asignado cada paciente. Se consideran 58 subáreas que se agrupan en el trabajo en 32 áreas más globales.
- **Id. hospitalización**: Variable cuantitativa discreta que identifica de manera única a cada paciente que ingresa en el hospital. Es la manera de identificar al paciente sin incurrir en violaciones de la ley de protección de datos.
- **Altas cód. vál.**: Variable cuantitativa que representa un código interno del hospital asociado al alta del paciente. No será de utilidad para la creación de modelos ni para la realización de ejemplos.
- **Cód. conc. res.**: Variable cuantitativa discreta que recoge el código postal del concello de residencia del paciente. Como para ninguno de los modelos empleados se han hecho distinciones según el lugar de procedencia del paciente, no se tendrá en cuenta esta variable.

- **Concello residencia**: Variable cualitativa que contiene el nombre del concello de residencia del paciente. Esta variable y la variable *Cód. conc. res.* están directamente relacionadas.
- **Sexo**: El sexo (Mujer u Hombre) del paciente. Es una variable cualitativa y su uso es indispensable en la formulación de los modelos, pues se ha restringido a lo largo del trabajo que no pueda haber en la misma habitación pacientes de diferente sexo.

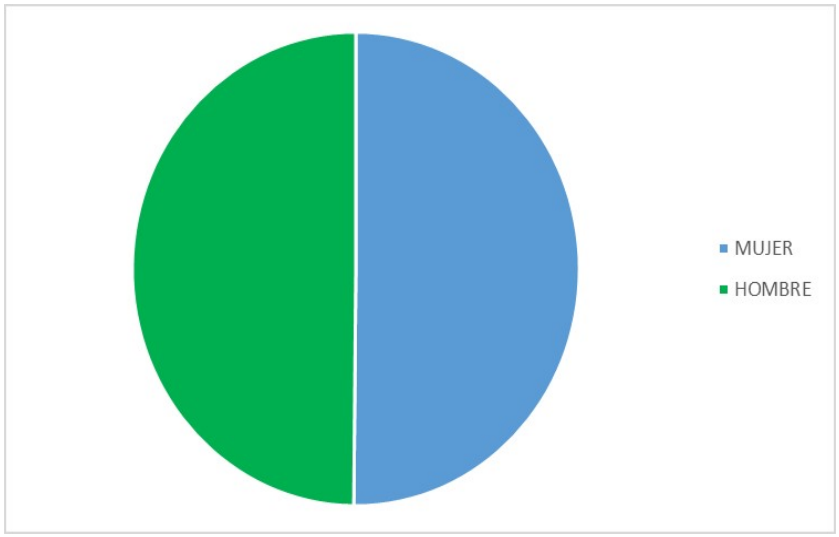
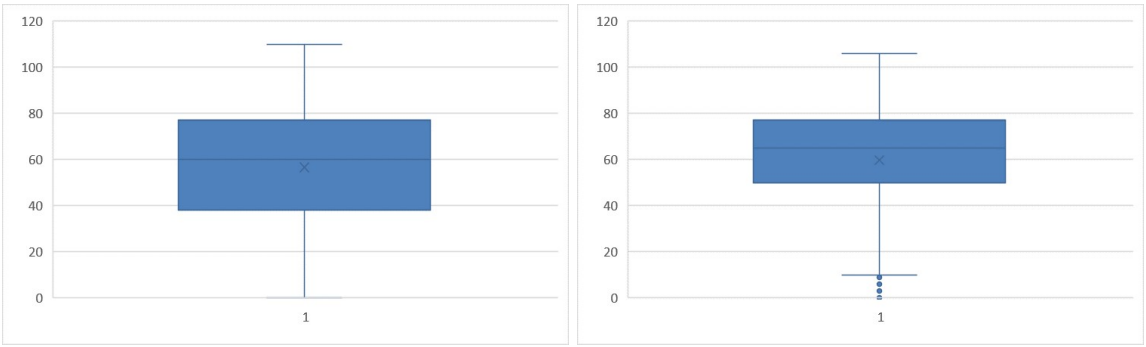


Figura A.4: Distribución de mujeres y hombres entre los pacientes del hospital.

- **Idade anos**: Recoge la edad en años de cada paciente. Es una variable cuantitativa discreta que es muy útil para identificar el nivel de gravedad de cada paciente. Además, permite hacer estudios de todo tipo, como la edad promedio de cada tipo de patología o de cada área del hospital.



(a) Edad de las pacientes de sexo femenino.

(b) Edad de los pacientes de sexo masculino.

Figura A.5: BoxPlot que representan la distribución de la edad según el sexo de los pacientes.

Apéndice B

Descripción de los *solvers* empleados

En este apéndice se presentarán brevemente los cuatro *solvers* para la resolución de problemas de optimización que han sido empleados a lo largo de este documento. Es decir, se presentará *BARON*, *Couenne*, *OCTERACT* y *Gurobi*.

B.1. BARON

BARON es un *solver* de optimización global diseñado hace casi treinta años y que es uno de las herramientas más potentes para resolver problemas de optimización de muy distintos tipos. De entre estas clases de problemas destacan:

- problemas de programación lineal,
- problemas de programación no lineal,
- problemas de programación entera,
- problemas de programación entera no lineal.

Pese a la amplia variedad de problemas que es capaz de resolver, el uso más destacado de BARON es la resolución problemas no lineales, tanto con variables enteras como sin ellas. De hecho, está considerado el primer *solver* comercial con capacidad para resolver problemas de programación no lineal y de programación entera no lineal. Es en este tipo de problemas donde emplea algoritmos de *branch and reduce*, similares a los de ramificación y acotación, combinados con el uso de heurísticas, para hallar la solución óptima del problema. Además, BARON no precisa un punto de inicio proporcionado por el usuario para inicializar el algoritmo. Más allá de las características técnicas del *solver*, BARON posee otra ventaja importante con respecto a algunos de sus competidores y es que se adapta a varios lenguajes de programación: MATLAB, JuMP, Pyomo, AMPL, GAMS, ...

Para más información acerca del *solver*, se puede consultar el link https://minlp.com/baron-solver.

B.2. Couenne

Al igual que BARON, Couenne es otro *solver* de optimización global con bastantes años uso. Sin embargo, su características son más limitadas, ya que está diseñado específicamente para la resolución de problemas de programación entera no lineal. En este tipo de problemas, el procedimiento del *solver*

es similar al de BARON, buscando reformularlos para convertirlos en problemas de programación lineal aproximados y sobre los que aplicar algoritmos de ramificación y acotación. Además, Couenne admite menos lenguajes de programación que BARON (AMPL y GAMS).

Más información acerca de Couenne se puede consultar en https://www.coin-or.org/Couenne/.

B.3. OCTERACT

OCTERACT es un *solver* mucho más reciente (2017) que BARON y Couenne, con un enfoque completamente distinto. En primer lugar, OCTERACT está especializado en problemas no lineales, tanto enteros como no, y produce resultados significativamente peores que los demás *solvers* si se emplea para resolver problemas lineales. Sin embargo, el enfoque para abordar problemas no lineales es completamente distinto al empleado por los *solvers* anteriores en los que se utilizaban técnicas similares al algoritmo de *branch and bound*, y es que OCTERACT emplea técnicas de Machine Learning innovadoras y que buscan mejorar los métodos clásicos de resolución para este tipo de problemas.

De nuevo se puede consultar más información acerca del *solver* en https://octeract.gg/.

B.4. Gurobi

Finalmente, Gurobi es un *solver* creado en 2008 y cuyo objetivo principal es bastante diferente al de los tres anteriores, ya que está especializado en la resolución de problemas lineales. Es un optimizador global muy potente con muy buen comportamiento para resolver problemas de programación lineal. Además, posee una versión que permite introducir restricciones cuadráticas empleando el atributo *nonconvex=2*. Como las restricciones no lineales que se tratan en este documento son de esta forma, Gurobi es también un *solver* adecuado para resolver los problemas que se formulan.

Más imformación acerca de Gurobi se puede consultar en https://www.gurobi.com/.

Apéndice C

Códigos AMPL y R para el PBA

En este apéndice se detallan los códigos empleados para la resolución de los diferentes problemas descritos a lo largo de esta memoria. Cabe destacar que todos ellos son de creación propia y específicos para la estructura de la base de datos proporcionada, aunque su uso sería fácilmente extensible a cualquier otro esquema de información. Se trata de códigos R y AMPL.

C.1. Código AMPL para el NL-PBA

A continuación se presenta el código, en lenguaje AMPL asociado al problema de optimización no lineal NL-PBA presentado en la Sección 2.1. Se definen los conjuntos de datos, los parámetros y las variables del modelo para posteriormente definir el problema de optimización.

```
#Conjuntos de datos del problema

set pacientes ordered; #Conjunto de pacientes
set camas ordered; #Conjunto de camas
set departamentos ordered; #Conjunto de departamentos
set habitaciones ordered; #Conjunto de habitaciones
set habdobles ordered; #Conjunto de habitaciones dobles
set hombres ordered; #Conjunto de hombres
set mujeres ordered; #Conjunto de mujeres
set pacprog ordered; #Conjunto de pacientes programados

#Parametros del problema

param P:=card{pacientes}; #Numero de pacientes
param C:=card{camas}; #Numero de camas
param PP:=card{pacprog}; #Numero de programados
param BD>=0; #Beneficio por departamento adecuado
param BN>=0; #Beneficio por nivel alto de riesgo asignado
param BP>=0; #Beneficio por tratar a un paciente programado
param umbraltotal>=0; #Umbral de ocupacion maxima en el hospital
param O default 0; #Numero de camas ocupadas antes del estudio
param G default 0; #Numero total de camas
param RiesgoPac {p in pacientes}; #Nivel de riesgo de cada paciente
param Ubi {c in camas, h in habitaciones} binary; #Ubicacion de cada cama
param DepHab {h in habitaciones, d in departamentos} binary; #Ubicacion de cada habitacion
param DepOpt {p in pacientes, d in departamentos} binary; #Departamento optimo para cada
    paciente
```

```
param suma default 0;

#Variables del problema

var y {p in pacientes} binary;
var x {p in pacientes, c in camas} binary;

#PROBLEMA DE OPTIMIZACION

##Funcion objetivo a maximizar

maximize Total_Beneficio: sum{p in pacientes} y[p]*BD + sum{p in pacientes, c in camas}
    x[p,c]*RiesgoPac[p]*BN + sum {p in pacprog, c in camas}x[p,c]*BP;

##Restricciones

#No puede haber un paciente con mas de una cama asignada
subj to UnicaCama{p in pacientes}:
   sum{c in camas} x[p,c]<=1;

#No puede haber una cama con mas de un paciente asignado
subj to UnicoPaciente{c in camas}:
   sum{p in pacientes} x[p,c]<=1;

#El numero total de camas usadas no puede superar un umbral
subj to UmbralOcupacion:
   (O+sum{p in pacientes, c in camas}x[p,c])/G <=umbraltotal;

#Relacion entre las variables
subj to RelacionVar{p in pacientes}:
   y[p]=sum{c in camas,h in habitaciones,d in
       departamentos}x[p,c]*Ubi[c,h]*DepHab[h,d]*DepOpt[p,d];

#Habitaciones mismo sexo
subj to MismoSexo{h in habdobles}:
   (sum{i in hombres,c in camas}x[i,c]*Ubi[c,h])*(sum{j in mujeres, c in
       camas}x[j,c]*Ubi[c,h])=0;

#Ajuste programados
subj to AjusteProgramados:
   sum{p in pacprog, c in camas} x[p,c]<=PP;
```

C.2. Código AMPL para el L-PBA

Para evitar la repetición de prácticamente la totalidad del código anterior, se presenta el único bloque de restricciones que diferencia la formulación del L-PBA de la Sección 2.2 con la del NL-PBA.

```
    #Habitaciones mismo sexo
subj to MismoSexoHombres{c in camas, b in camas, h in habdobles, i in hombres, j in mujeres}:
   (1-x[i,c]-x[j,b])*Ubi[c,h]*Ubi[b,h]>=0;
```

C.3. Código R para el modelo predictivo

En esta sección se presentará el código en lenguaje R del modelo predictivo considerado en el trabajo empleando la herramienta *Prophet*. Antes de presentar el código, se detallan algunos elementos empleados en él extraídos de la base de datos de pacientes.

- ***historia*** hace referencia a la secuencia temporal que se ha considerado para diseñar el modelo predictivo.
- ***Ingreso*** es el vector compuesto por las fechas de ingreso de cada paciente de la base de datos.
- ***Alta*** es el vector compuesto por las fechas de alta de cada paciente de la base de datos.
- ***numcamas*** es el número total de camas del hospital.

```
install.packages('prophet')
library(prophet)

capacidad<-c()
ingresos<-capacidad
k<-1
for (i in historia){
  capacidad[k]<-(sum(Ingreso<i & Alta>=i))/numcamas
  ingresos[k]<-sum(Ingreso==i)
  k<-k+1
}

df_ing<-data.frame("ds"=historia,"y"=ingresos)
m_ing <- prophet(df_ing,daily.seasonality=TRUE)
future_ing <- make_future_dataframe(m_ing, periods = 365)
tail(future_ing)
forecast_ing <- predict(m_ing, future_ing)
tail(forecast_ing[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
plot(m_ing, forecast_ing,xlab="",ylab="")
prophet_plot_components(m_ing, forecast_ing)
dyplot.prophet(m_ing, forecast_ing)

df_cap<-data.frame("ds"=historia,"y"=capacidad)
m_cap <- prophet(df_cap,daily.seasonality=TRUE)
future_cap <- make_future_dataframe(m_cap, periods = 365)
tail(future_cap)
forecast_cap <- predict(m_cap, future_cap)
tail(forecast_cap[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
plot(m_cap, forecast_cap,xlab="",ylab="")
prophet_plot_components(m_cap, forecast_cap)
dyplot.prophet(m_cap, forecast_cap)
```

C.4. Código AMPL para el PBA-s

En esta sección se presenta el código AMPL asociado al PBA-s. Por simplicidad, solo se presentará el código asociado al ejemplo de la Sección 3.2.2, pero para el caso con cuatro etapas sería análogo.

```
    #Conjuntos de datos del problema
```

```
set pacientes0 ordered;
set pacientes01 ordered;
set pacientes1 ordered;
set pacientes11 ordered;
set pacientes12 ordered;
set pacientes13 ordered;
set camastotales ordered;
set departamentos ordered;
set habitaciones ordered;
set habdobles ordered;
set hombres0 ordered;
set hombres01 ordered;
set hombres1 ordered;
set hombres11 ordered;
set hombres12 ordered;
set hombres13 ordered;
set mujeres0 ordered;
set mujeres01 ordered;
set mujeres1 ordered;
set mujeres11 ordered;
set mujeres12 ordered;
set mujeres13 ordered;
set pacprog01 ordered;
set pacprog11 ordered;
set pacprog12 ordered;
set pacprog13 ordered;

#Parametros del problema

param T:=card{camastotales}; #Numero de camas
param PP01:=card{pacprog01}; #Numero de programados
param PP11:=card{pacprog11};
param PP12:=card{pacprog12};
param PP13:=card{pacprog13};
param BD>=0; #Beneficio por departamento adecuado
param BN>=0; #Beneficio por nivel alto de riesgo asignado
param BP>=0; #Beneficio por tratar a un paciente programado
param umbraltotal>=0; #Umbral de ocupacion maxima en el hospital
param RiesgoPac01 {p in pacientes01}; #Nivel de riesgo de cada paciente
param RiesgoPac11 {p in pacientes11};
param RiesgoPac12 {p in pacientes12};
param RiesgoPac13 {p in pacientes13};
param Ubi {t in camastotales, h in habitaciones} binary; #Ubicacion de cada cama
param DepHab {h in habitaciones, d in departamentos} binary; #Ubicacion de cada habitacion
param DepOpt01 {p in pacientes01, d in departamentos} binary; #Departamento optimo para cada
    paciente
param DepOpt11 {p in pacientes11, d in departamentos} binary;
param DepOpt12 {p in pacientes12, d in departamentos} binary;
param DepOpt13 {p in pacientes13, d in departamentos} binary;
param X0 {p in pacientes0, t in camastotales}; #Ubicacion previa de los pacientes
param F0 {p in pacientes0}; #Tiempo de estancia esperado de cada paciente
param F01 {p in pacientes01};
param F11 {p in pacientes11};
param F12 {p in pacientes12};
param F13 {p in pacientes13};
param suma default 0;
```

```
param q11 default 0; #Probabilidad del escenario 1
param q12 default 0; #Probabilidad del escenario 2

#Variables del problema

var y01 {p in pacientes01} binary;
var y11 {p in pacientes11} binary;
var y12 {p in pacientes12} binary;
var y13 {p in pacientes13} binary;
var x01 {p in pacientes01, t in camastotales} binary;
var x11 {p in pacientes11, t in camastotales} binary;
var x12 {p in pacientes12, t in camastotales} binary;
var x13 {p in pacientes13, t in camastotales} binary;
var z0 {p in pacientes0, t in camastotales} binary;
var z1 {p in pacientes1, t in camastotales} binary;

#PROBLEMA DE OPTIMIZACION

##Funcion objetivo a maximizar

maximize Total_Beneficio: sum{p in pacientes01} y01[p]*BD + sum{p in pacientes01, t in
    camastotales} x01[p,t]*RiesgoPac01[p]*BN + sum {p in pacprog01, t in
    camastotales}x01[p,t]*BP+q11*(sum{p in pacientes11} y11[p]*BD + sum{p in pacientes11, t
    in camastotales} x11[p,t]*RiesgoPac11[p]*BN + sum {p in pacprog11, t in
    camastotales}x11[p,t]*BP)+q12*(sum{p in pacientes12} y12[p]*BD + sum{p in pacientes12, t
    in camastotales} x12[p,t]*RiesgoPac12[p]*BN + sum {p in pacprog12, t in
    camastotales}x12[p,t]*BP)+(1-q11-q12)*(sum{p in pacientes13} y13[p]*BD + sum{p in
    pacientes13, t in camastotales} x13[p,t]*RiesgoPac13[p]*BN + sum {p in pacprog13, t in
    camastotales}x13[p,t]*BP);

##Restricciones

#No puede haber un paciente con mas de una cama asignada
subj to UnicaCama01{p in pacientes01}:
   sum{t in camastotales} x01[p,t]<=1;
subj to UnicaCama11{p in pacientes11}:
   sum{t in camastotales} x11[p,t]<=1;
subj to UnicaCama12{p in pacientes12}:
   sum{t in camastotales} x12[p,t]<=1;
subj to UnicaCama13{p in pacientes13}:
   sum{t in camastotales} x13[p,t]<=1;

#No puede haber una cama con mas de un paciente asignado
subj to UnicoPaciente01{t in camastotales}:
   sum{p in pacientes01} x01[p,t]<=1;
subj to UnicoPaciente11{t in camastotales}:
   sum{p in pacientes11} x11[p,t]<=1;
subj to UnicoPaciente12{t in camastotales}:
   sum{p in pacientes12} x12[p,t]<=1;
subj to UnicoPaciente13{t in camastotales}:
   sum{p in pacientes13} x13[p,t]<=1;

#El numero total de camas usadas no puede superar un umbral
subj to UmbralOcupacion01:
   (sum{p in pacientes01, t in camastotales}x01[p,t]+sum{r in pacientes0,t in
       camastotales}z0[r,t])/T <=umbraltotal;
```

```
subj to UmbralOcupacion11:
   (sum{p in pacientes11, t in camastotales}x11[p,t]+sum{r in pacientes1,t in
       camastotales}z1[r,t])/T <=umbraltotal;
subj to UmbralOcupacion12:
   (sum{p in pacientes12, t in camastotales}x12[p,t]+sum{r in pacientes1,t in
       camastotales}z1[r,t])/T <=umbraltotal;
subj to UmbralOcupacion13:
   (sum{p in pacientes13, t in camastotales}x13[p,t]+sum{r in pacientes1,t in
       camastotales}z1[r,t])/T <=umbraltotal;


#Relacion entre las variables
subj to RelacionVar01{p in pacientes01}:
   y01[p]=sum{t in camastotales,h in habitaciones,d in
       departamentos}x01[p,t]*Ubi[t,h]*DepHab[h,d]*DepOpt01[p,d];
subj to RelacionVar11{p in pacientes11}:
   y11[p]=sum{t in camastotales,h in habitaciones,d in
       departamentos}x11[p,t]*Ubi[t,h]*DepHab[h,d]*DepOpt11[p,d];
subj to RelacionVar12{p in pacientes12}:
   y12[p]=sum{t in camastotales,h in habitaciones,d in
       departamentos}x12[p,t]*Ubi[t,h]*DepHab[h,d]*DepOpt12[p,d];
subj to RelacionVar13{p in pacientes13}:
   y13[p]=sum{t in camastotales,h in habitaciones,d in
       departamentos}x13[p,t]*Ubi[t,h]*DepHab[h,d]*DepOpt13[p,d];

#Ajuste programados
subj to AjusteProgramados01:
   sum{p in pacprog01, t in camastotales} x01[p,t]<=PP01;
subj to AjusteProgramados11:
   sum{p in pacprog11, t in camastotales} x11[p,t]<=PP11;
subj to AjusteProgramados12:
   sum{p in pacprog12, t in camastotales} x12[p,t]<=PP12;
subj to AjusteProgramados13:
   sum{p in pacprog13, t in camastotales} x13[p,t]<=PP13;


#Habitaciones mismo sexo (a)
subj to MismoSexoa01{h in habdobles}:
   (sum{i in hombres01,t in camastotales}x01[i,t]*Ubi[t,h])*(sum{j in mujeres01, t in
       camastotales}x01[j,t]*Ubi[t,h])=0;
subj to MismoSexoa11{h in habdobles}:
   (sum{i in hombres11,t in camastotales}x11[i,t]*Ubi[t,h])*(sum{j in mujeres11, t in
       camastotales}x11[j,t]*Ubi[t,h])=0;
subj to MismoSexoa12{h in habdobles}:
   (sum{i in hombres12,t in camastotales}x12[i,t]*Ubi[t,h])*(sum{j in mujeres12, t in
       camastotales}x12[j,t]*Ubi[t,h])=0;
subj to MismoSexoa13{h in habdobles}:
   (sum{i in hombres13,t in camastotales}x13[i,t]*Ubi[t,h])*(sum{j in mujeres13, t in
       camastotales}x13[j,t]*Ubi[t,h])=0;

#Habitaciones mismo sexo (b)
subj to MismoSexob01{h in habdobles}:
   (sum{i in mujeres0,t in camastotales}z0[i,t]*Ubi[t,h])*(sum{j in hombres01, t in
       camastotales}x01[j,t]*Ubi[t,h])=0;
subj to MismoSexob11{h in habdobles}:
   (sum{i in mujeres1,t in camastotales}z1[i,t]*Ubi[t,h])*(sum{j in hombres11, t in
```

```
       camastotales}x11[j,t]*Ubi[t,h])=0;
subj to MismoSexob12{h in habdobles}:
   (sum{i in mujeres1,t in camastotales}z1[i,t]*Ubi[t,h])*(sum{j in hombres12, t in
       camastotales}x12[j,t]*Ubi[t,h])=0;
subj to MismoSexob13{h in habdobles}:
   (sum{i in mujeres1,t in camastotales}z0[i,t]*Ubi[t,h])*(sum{j in hombres13, t in
       camastotales}x13[j,t]*Ubi[t,h])=0;

#Habitaciones mismo sexo (c)
subj to MismoSexoc01{h in habdobles}:
   (sum{i in mujeres01,t in camastotales}x01[i,t]*Ubi[t,h])*(sum{j in hombres0, t in
       camastotales}z0[j,t]*Ubi[t,h])=0;
subj to MismoSexoc11{h in habdobles}:
   (sum{i in mujeres11,t in camastotales}x11[i,t]*Ubi[t,h])*(sum{j in hombres1, t in
       camastotales}z1[j,t]*Ubi[t,h])=0;
subj to MismoSexoc12{h in habdobles}:
   (sum{i in mujeres12,t in camastotales}x12[i,t]*Ubi[t,h])*(sum{j in hombres1, t in
       camastotales}z1[j,t]*Ubi[t,h])=0;
subj to MismoSexoc13{h in habdobles}:
   (sum{i in mujeres13,t in camastotales}x13[i,t]*Ubi[t,h])*(sum{j in hombres1, t in
       camastotales}z1[j,t]*Ubi[t,h])=0;

#Inicializacion z's
subj to Inicializacion{p in pacientes0, t in camastotales}:
   z0[p,t]+z1[p,t]=X0[p,t]*min(F0[p],2);

#Actualizacion z's
subj to Actualizacion01{p in pacientes01, t in camastotales}:
   z1[p,t]=x01[p,t]*min(F01[p]-1,1);

#Restriccion z's
subj to Restriccion{p in pacientes0,t in camastotales}:
   z0[p,t]>=z1[p,t];

#Relacion z's y x's
subj to Relacionzx{p in pacientes01,t in camastotales}:
   z1[p,t]-x01[p,t]<=0;

#Restriccion ocupacion
subj to RestOcup01{t in camastotales}:
   sum{p in pacientes0}z0[p,t]*sum{r in pacientes01}x01[r,t]=0;
subj to RestOcup11{t in camastotales}:
   sum{p in pacientes1}z1[p,t]*sum{r in pacientes11}x11[r,t]=0;
subj to RestOcup12{t in camastotales}:
   sum{p in pacientes1}z1[p,t]*sum{r in pacientes12}x12[r,t]=0;
subj to RestOcup13{t in camastotales}:
   sum{p in pacientes1}z1[p,t]*sum{r in pacientes13}x13[r,t]=0;
```

C.5. Código R para el árbol de decisión

Antes de presentar el código en lenguaje R asociado a la heurística del árbol de decisión de la Sección 4.2 se describen a continuación los conjuntos empleados extraídos directamente de la base de datos de pacientes.

- ***pacientes*** es el conjunto de pacientes que buscan ingresar en el hospital.

- ***programados*** es un subconjunto de *pacientes* que contiene a los de tipo de ingreso programado.
- ***riesgopac*** es un vector que almacena el nivel de riesgo de cada paciente.
- ***corresp*** es un data frame que relaciona cada subárea del hospital con su correspondiente área asociada mediante un número.
- ***Sexo*** recoge el sexo de cada paciente.
- ***dobleslib*** es un vector que almacena el número de camas disponibles en cada área que se encuentran en habitaciones dobles.

```
progs<-numeric(length(pacientes))
riesgos<-numeric(length(pacientes))
areas<-numeric(length(pacientes))
num_pac<-numeric(length(pacientes))
sexos<-c()
u<-1
for (i in pacientes){
  if(i %in% programados) progs[u]<-"Si"
  else progs[u]<-"No"
  riesgos[u]<-riesgopac[i]
  areas[u]<-as.numeric(subset(corresp, area == AreaDiag[i], select = num))
  sexos[u]<-Sexo[i]
  num_pac[u]<-u
  u<-u+1
}

set.seed(123)

#Se crea un data frame que se actualizara en cada iteracion del algoritmo
df <- data.frame(num_pac,pacientes,riesgos,areas,sexos,
  camas_asignadas = rep(NA, length(pacientes)),
  progs)

#Se crea un vector de listas, donde cada entrada se corresponde con la capacidad de las
    camas libres segun el sexo
dobleslib_sexos <- vector("list", 32)
for(i in 1:32){
  dobleslib_sexos[[i]] <- data.frame(
    cama1 = rep(NA, dobleslib[i] %/% 2),
    cama2 = rep(NA, dobleslib[i] %/% 2)
  )
}

# Funcion para trasladar un paciente de bajo riesgo a otra area
trasladar_paciente <- function(df, camasdep, camaslibres, camasocupadas, libresind,
    dobleslib, dobleslib_sexos, area){
  # Identificar pacientes de bajo riesgo en el area que pueden ser trasladados
  pacientes_traslado <- which(df$areas == area & df$riesgos < 5 & !is.na(df$camas_asignadas))
  # Si no hay pacientes que puedan ser trasladados, devolver NULL
  if(length(pacientes_traslado) == 0) return(NULL)
  # Elegir un paciente para trasladar (el primero en la lista)
  pac_traslado <- pacientes_traslado[1]
  # Buscar el area con el mayor numero de camas libres
  area_alternativa <- which.max(camaslibres)
```

```
  # Actualizar el departamento del paciente y la informacion de la cama
  df[pac_traslado, 'areas'] <- area_alternativa
  return(list(df, area_alternativa, pac_traslado))
}

# Funcion para buscar pacientes programados que hayan ingresado previamente
buscar_paciente_programado <- function(pacientes, pac) {
  indice_pac <- match(pac, pacientes)
  # Verificar si el paciente actual esta en la lista de pacientes
  if (!is.na(indice_pac)) {
    # Buscar pacientes previos al paciente actual en la lista
    pacientes_previos <- pacientes[1:(indice_pac - 1)]
    # Buscar el primer paciente programado en los pacientes previos
    indice_paciente_programado <- match("Si", progs[pacientes_previos])
    # Si se encuentra un paciente programado, devolverlo
    if (!is.na(indice_paciente_programado)) {
      paciente_programado <- pacientes_previos[indice_paciente_programado]
      return(paciente_programado)
    }
  }
  # Si no se encuentra un paciente programado, devolver NULL
  return(NULL)
}

# Funcion para asignar cama a un paciente
asignar_cama <- function(df, camasdep, camaslibres, camasocupadas, libresind, dobleslib,
    dobleslib_sexos, BN, BD, pac){
  area <- df[df$pacientes == pac, "areas"]
  riesgo <- df[df$pacientes == pac, 'riesgos']
  sexo <- df[df$pacientes == pac, 'sexos']

  # Si hay camas disponibles en el area
  if(camaslibres[area] > 0){
    # Si hay camas individuales libres, asignar una
    if(libresind[area] > 0){
      df[df$pacientes == pac, 'camas_asignadas'] <- 1
      camaslibres[area] <- camaslibres[area] - 1
      camasocupadas[area] <- camasocupadas[area] + 1
      libresind[area] <- libresind[area] - 1
      if (df[df$pacientes == pac, 'progs']=="Si"){
      return(list(BN * riesgo + BD + BP, df, dobleslib_sexos))
      }
      else return(list(BN * riesgo + BD , df, dobleslib_sexos))
    }
    # Si no hay camas individuales libres, verificar camas dobles
    else if(dobleslib[area] > 0){
      # Identificar habitaciones dobles con al menos una cama libre
      habitaciones_disponibles <- which((is.na(dobleslib_sexos[[area]]$cama1) &
          is.na(dobleslib_sexos[[area]]$cama2)) |
                                        (is.na(dobleslib_sexos[[area]]$cama1) &
                                            dobleslib_sexos[[area]]$cama2 == sexo) |
                                        (dobleslib_sexos[[area]]$cama1 == sexo &
                                            is.na(dobleslib_sexos[[area]]$cama2)))

      # Si hay una habitacion doble compatible, asignar la cama y actualizar dobleslib_sexos
      if(length(habitaciones_disponibles) > 0){
```

```
      df[df$pacientes == pac, 'camas_asignadas'] <- 2
      camaslibres[area] <- camaslibres[area] - 1
      camasocupadas[area] <- camasocupadas[area] + 1
      dobleslib[area] <- dobleslib[area] - 1
      if(is.na(dobleslib_sexos[[area]][habitaciones_disponibles[1], 'cama1']){
        dobleslib_sexos[[area]][habitaciones_disponibles[1], 'cama1'] <- sexo
      } else{
        dobleslib_sexos[[area]][habitaciones_disponibles[1], 'cama2'] <- sexo
      }
      if (df[df$pacientes == pac, 'progs']=="Si"){
        return(list(BN * riesgo + BD + BP, df, dobleslib_sexos))
      }
      else return(list(BN * riesgo + BD , df, dobleslib_sexos))
    }
    else return(list(0,df,dobleslib_sexos))
  }
}

else if(riesgo <= 8) {
  # Buscar el area con el mayor numero de camas libres
  area_alternativa <- which.max(camaslibres)

  # Si hay un area alternativa con camas libres, asignar una cama
  if(camaslibres[area_alternativa] > 0){
    # Si hay camas individuales libres, asignar una
    if(libresind[area_alternativa] > 0){
      df[df$pacientes == pac, 'camas_asignadas'] <- 1
      camaslibres[area_alternativa] <- camaslibres[area_alternativa] - 1
      camasocupadas[area_alternativa] <- camasocupadas[area_alternativa] + 1
      libresind[area_alternativa] <- libresind[area_alternativa] - 1
      if (df[df$pacientes == pac, 'progs']=="Si"){
        return(list(BN * riesgo + BP, df, dobleslib_sexos))
      }
      else return(list(BN * riesgo , df, dobleslib_sexos))
    }
    # Si no hay camas individuales libres, verificar camas dobles
    else if(dobleslib[area_alternativa] > 0){
      # Identificar habitaciones dobles con al menos una cama libre y del mismo sexo
      habitaciones_disponibles <- which((is.na(dobleslib_sexos[[area]]$cama1) &
          is.na(dobleslib_sexos[[area]]$cama2)) |
                                      (is.na(dobleslib_sexos[[area]]$cama1) &
                                          dobleslib_sexos[[area]]$cama2 == sexo) |
                                      (dobleslib_sexos[[area]]$cama1 == sexo &
                                          is.na(dobleslib_sexos[[area]]$cama2)))

      # Si hay una habitacion doble compatible, asignar la cama y actualizar dobleslib_sexos
      if(length(habitaciones_disponibles) > 0){
        df[df$pacientes == pac, 'camas_asignadas'] <- 2
        camaslibres[area_alternativa] <- camaslibres[area_alternativa] - 1
        camasocupadas[area_alternativa] <- camasocupadas[area_alternativa] + 1
        dobleslib[area_alternativa] <- dobleslib[area_alternativa] - 1
        if(is.na(dobleslib_sexos[[area_alternativa]][habitaciones_disponibles[1], 'cama1']){
          dobleslib_sexos[[area_alternativa]][habitaciones_disponibles[1], 'cama1'] <- sexo
        } else{
          dobleslib_sexos[[area_alternativa]][habitaciones_disponibles[1], 'cama2'] <- sexo
        }
```

```
        if (df[df$pacientes == pac, 'progs']=="Si"){
          return(list(BN * riesgo + BP, df, dobleslib_sexos))
        }
        else{
        return(list(BN * riesgo, df, dobleslib_sexos))
        }
      }
    }
  }
}
else{
  # Intentar trasladar a un paciente de bajo riesgo
  traslado <- trasladar_paciente(df, camasdep, camaslibres, camasocupadas, libresind,
      dobleslib, dobleslib_sexos, area)

  # Si se puede trasladar a un paciente, hacerlo y asignar la cama al paciente actual
  if(!is.null(traslado)){
    df <- traslado[[1]]
    area_alternativa <- traslado[[2]]
    pac_traslado <- traslado[[3]]

    # Actualizar la informacion de la cama para el paciente trasladado
    df <- asignar_cama(df, camasdep, camaslibres, camasocupadas, libresind, dobleslib,
        dobleslib_sexos, BN, BD, pac_traslado)[[2]]

    # Asignar la cama al paciente actual
    df[df$pacientes == pac, 'camas_asignadas'] <- df[pac_traslado, 'camas_asignadas']
    df[df$pacientes == pac_traslado, 'camas_asignadas'] <- NA
    camaslibres[area] <- camaslibres[area] + 1
    camasocupadas[area] <- camasocupadas[area] - 1
    if (df[df$pacientes == pac, 'progs']=="Si"){
      return(list(BN * riesgo + BD + BP, df, dobleslib_sexos))
    }
    else return(list(BN * riesgo + BD , df, dobleslib_sexos))
  }
  else{
    paciente_programado <- buscar_paciente_programado(pacientes, pac)

    if (!is.null(paciente_programado)) {

      # Actualizar la contribucion del nuevo paciente y restar la contribucion del paciente
          programado cancelado
      if (df[df$pacientes == pac, 'progs'] == "Si") {
        return(list(BN * (riesgo - df[df$pacientes == paciente_programado, 'riesgos']), df,
            dobleslib_sexos))
      } else {
        return(list(BN * (riesgo - df[df$pacientes == paciente_programado, 'riesgos']) -
            BP, df, dobleslib_sexos))
      }
    }
    else{
      # Si hay un area alternativa con camas libres, asignar una cama
      if(camaslibres[area_alternativa] > 0){
        # Si hay camas individuales libres, asignar una
        if(libresind[area_alternativa] > 0){
          df[df$pacientes == pac, 'camas_asignadas'] <- 1
```

```
          camaslibres[area_alternativa] <- camaslibres[area_alternativa] - 1
          camasocupadas[area_alternativa] <- camasocupadas[area_alternativa] + 1
          libresind[area_alternativa] <- libresind[area_alternativa] - 1
          if (df[df$pacientes == pac, 'progs']=="Si"){
            return(list(BN * riesgo + BP, df, dobleslib_sexos))
          }
          else return(list(BN * riesgo , df, dobleslib_sexos))
        }
        # Si no hay camas individuales libres, verificar camas dobles
        else if(dobleslib[area_alternativa] > 0){
          # Identificar habitaciones dobles con al menos una cama libre y del mismo sexo
          habitaciones_disponibles <- which((is.na(dobleslib_sexos[[area]]$cama1) &
              is.na(dobleslib_sexos[[area]]$cama2)) |
                                            (is.na(dobleslib_sexos[[area]]$cama1) &
                                                dobleslib_sexos[[area]]$cama2 == sexo) |
                                            (dobleslib_sexos[[area]]$cama1 == sexo &
                                                is.na(dobleslib_sexos[[area]]$cama2)))

          # Si hay una habitacion doble compatible, asignar la cama y actualizar
              dobleslib_sexos
          if(length(habitaciones_disponibles) > 0){
            df[df$pacientes == pac, 'camas_asignadas'] <- 2
            camaslibres[area_alternativa] <- camaslibres[area_alternativa] - 1
            camasocupadas[area_alternativa] <- camasocupadas[area_alternativa] + 1
            dobleslib[area_alternativa] <- dobleslib[area_alternativa] - 1
            if(is.na(dobleslib_sexos[[area_alternativa]][habitaciones_disponibles[1],
                'cama1']){
              dobleslib_sexos[[area_alternativa]][habitaciones_disponibles[1], 'cama1'] <-
                  sexo
            } else{
              dobleslib_sexos[[area_alternativa]][habitaciones_disponibles[1], 'cama2'] <-
                  sexo
            }
            if (df[df$pacientes == pac, 'progs']=="Si"){
              return(list(BN * riesgo + BP, df, dobleslib_sexos))
            }
            else return(list(BN * riesgo , df, dobleslib_sexos))
            return(list(BN * riesgo, df, dobleslib_sexos))
          }
        }
      }
    }
  }
}

# Bucle principal
beneficio_total <- 0
contribuciones<-numeric(length(pacientes))
m<-1
tiempo<-system.time(

for(pac in pacientes){
  if (sum(camasocupadas)/sum(camasdep)>=eta) break
  resultado <- asignar_cama(df, camasdep, camaslibres, camasocupadas, libresind, dobleslib,
      dobleslib_sexos, BN, BD, pac)
```

```
  contribuciones[m]<-resultado[[1]]
  beneficio_total <- beneficio_total + resultado[[1]]
  df <- resultado[[2]]
  dobleslib_sexos <- resultado[[3]]
  m<-m+1
}
)
```

Referencias

Antelo, M., Santias, F. R., y Calvo, A. M. (2015). Bed capacity and surgical waiting lists: a simulation analysis. *European Journal of Government and Economics*, *4*(2), 118–133.

Birge, J. R., y Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Ceschia, S., y Schaerf, A. (2011). Local search and lower bounds for the patient admission scheduling problem. *Computers & Operations Research*, *38*(10), 1452–1463.

Demeester, P., Souffriau, W., De Causmaecker, P., y Berghe, G. V. (2010). A hybrid tabu search algorithm for automatically assigning patients to beds. *Artificial Intelligence in Medicine*, *48*(1), 61–70.

Díaz, A., Glover, F., Ghaziri, H., González, J., Laguna, M., Moscato, P., y Tseng, F. T. (2000). *Optimización heurística y redes neuronales*. Paraninfo.

Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management science*, *27*(1), 1–18.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications*, *10*, 237–260.

Geoffrion, A. M. (1974). Lagrangean relaxation and its uses in integer programming. *Mathematical Programming Study*, *82*(2).

Guido, R., y Conforti, D. (2017). A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. *Computers & Operations Research*, *87*, 270–282.

Guido, R., Groccia, M. C., y Conforti, D. (2018). An efficient matheuristic for offline patient-to-bed assignment problems. *European Journal of Operational Research*, *268*(2), 486–503.

Kroneman, M., y Siegers, J. J. (2004). The effect of hospital bed reduction on the use of beds: a comparative study of 10 European countries. *Social science & medicine*, *59*(8), 1731–1740.

Mısır, M., Verbeeck, K., De Causmaecker, P., y Berghe, G. V. (2013). An investigation on the generality level of selection hyper-heuristics under different empirical conditions. *Applied Soft Computing*, *13*(7), 3335–3353.

Noonan, F., O'Brien, J., Broderick, E., Richardson, I., y Walsh, J. (2019). Hospital bed management practices: A review. *HEALTHINF*, 326–331.

Rahmaniani, R., Crainic, T. G., Gendreau, M., y Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*(3), 801–817.

Range, T. M., Lusby, R. M., y Larsen, J. (2014). A column generation approach for solving the patient admission scheduling problem. *European Journal of Operational Research*, *235*(1), 252–264.

Taylor, S. J., y Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45.

Vancroonenburg, W., De Causmaecker, P., Spieksma, F., y Berghe, G. V. (2013). Scheduling elective patient admissions considering room assignment and operating theatre capacity constraints. En *Proceedings of the 5th international conference on applied operational research, lecture notes in management science* (Vol. 5, pp. 153–158).

Vancroonenburg, W., Goossens, D., y Spieksma, F. (2011). On the complexity of the patient assignment problem. *Tech. rep., KAHO Sint-Lieven, Gebroeders De Smetstraat 1, Gent, Belgium*.