Proyecto Fin de Carrera
Ingeniería de Tecnologías Industriales

Desarrollo de modelos de programación lineal para la gestión del flujo de pacientes en un servicio de urgencias hospitalario

Autor: Juan López Garcelán

Tutor: José Manuel Molina Pariente

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

Sevilla, 2023

Proyecto Fin de Carrera
Ingeniería de Tecnologías Industriales

Desarrollo de modelos de programación lineal para la gestión del flujo de pacientes en un servicio de urgencias hospitalario

Autor:
Juan López Garcelán

Tutor:
José Manuel Molina Pariente

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Proyecto Fin de Carrera: Desarrollo de modelos de programación lineal para la gestión del flujo de pacientes en un servicio de urgencias hospitalario

Autor: Juan López Garcelán

Tutor: José Manuel Molina Pariente

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, agradecer a mi familia y amigos, que han permanecido a mi lado a lo largo de toda mi etapa educativa, aportándome su apoyo, confianza, madurez, felicidad y amor, haciéndome crecer y madurar tanto en lo personal como en lo académico. Sin este pilar fundamental, estoy seguro de que no habría llegado hasta aquí y no sería quien soy ahora mismo.

Por otro lado, agradecer a todos los profesores del grado su tiempo invertido en mi aprendizaje y educación, así como al personal de movilidad, que me permitió vivir una experiencia única en el curso académico 2021-2022 de Erasmus. Por último, apreciar y agradecer el apoyo que José Manuel me ha dado durante la realización del presente TFG. Como él sabe, tenía pocos conocimientos sobre el tópico que concierne a este trabajo, pero me ha aceptado y me ha dedicado mucho tiempo en que yo aprenda y que hoy pueda entregar este proyecto. Gracias por tu cercanía y confianza depositada en mí.

Juan López Garcelán

Sevilla, 2023

Resumen

La saturación de los servicios de urgencia hospitalario es una realidad actual, y es un problema complejo que a día de hoy sigue sin tener solución. Con la reciente pandemia mundial que nos atacó a todos, este problema a cobrado más importancia aún si cabe, pues los niveles de saturación de todos los hospitales del mundo alcanzaron niveles de saturación insospechables.

En el presente TFG se propone el desarrollo de varios modelos de programación lineal para la asignación de pacientes a recursos a lo largo de todo el proceso de urgencias de cada paciente de forma óptima. Con ello, se podría incrementar la calidad de servicio del servicio de urgencias, así como mejorar la eficiencia operativa y la capacidad de respuesta de los mismos.

Los servicios de urgencia hospitalarios tienen que dar servicio a pacientes en muchas situaciones distintas, con lo que en este trabajo también se desarrollan modelos de programación lineal que atañen a distintas variantes reales que se podrían dar en el marco diario de cualquier servicio de urgencias hospitalario.

Abstract

The saturation of emergency departments is a current reality, and is a complex problem that still has no solution today. Due to the recent global pandemic that attacked us all, this problem has become even more important, if possible, since the saturation levels of all hospitals in the world reached unsuspected levels.

This Final Degree Project deals with the development of several linear programming models with the aim of assigning resources to patients throughout the entire emergency process of each patient in an optimal way. Using these models, the quality of the service could be increased, as well as the operational efficiency and response capacity of emergency departments would be improved.

Hospital emergency departments have to provide service to patients in many different situations, so that in this Final Degree Project linear programming models are also developed, concerning different real scenarios that could occur in the daily setting of any hospital emergency department.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

SUH	Servicio de Urgencia Hospitalario
CC.AA. [1]	Comunidades Autónomas
LOS	Length of Stay
TEPCOF	Tiempo de Espera hasta la Primera Consulta Facultativa
$<$	Menor
$>$	Mayor
$\leq$	Menor o igual
$\geq$	Mayor o igual
$\mid$	Tal que
$\in$	Pertenece
$\forall$	Para todo

1 INTRODUCCIÓN

La salud es la riqueza real, y no piezas de oro y plata.

- Mahatma Gandhi-

La optimización de recursos es un área cada vez más presente en todos los sectores de la industria. Así pues, los conceptos de eficiencia, eficacia o productividad conforman la base de cualquier industria, cuyo objetivo común es dar un servicio de la mayor calidad posible a coste mínimo. Para conseguir el mayor beneficio dentro de un proceso productivo, es fundamental realizar una buena gestión de los recursos que se tienen, de modo que no se produzcan despilfarros ni tiempos ociosos.

La gestión de los recursos es un desafío clave en cualquier ámbito, ya sea en el sector público o privado. En el contexto empresarial, una gestión eficiente de los recursos se vuelve aún más crucial debido a su impacto directo en la rentabilidad, productividad y competitividad de las organizaciones. La asignación adecuada y la optimización de los recursos, tanto tangibles como intangibles, son elementos fundamentales para lograr el éxito y la sostenibilidad a largo plazo.

En el sector sanitario es muy importante la optimización de los recursos, ya que aunque esta no va ligada a una rentabilidad cuantificable monetaria como la mayoría de las industrias, una buena gestión de los recursos que se tienen dentro del servicio de urgencias de un hospital puede llegar a salvar vidas, que aunque no cuantificable, es mucho más importante.

Se ha vivido de primera mano hace tres años que la vida puede cambiar en cuestión de segundos y que los hospitales pueden desbordarse hasta niveles insospechables. Ello debe servir de lección para intentar prepararnos para cualquier otra situación similar que se pueda presentar, en la que cada segundo cuenta. Para poder afrontar situaciones de esta escala, hay que formar una base consistente del sistema sanitario, donde no existan despilfarros y donde, en las peores situaciones, se pueda exprimir el uso de los recursos al cien por cien.

Este trabajo de fin de grado toma aún más valor después de que hace tres años la Organización Mundial de la Salud declarase una pandemia mundial odiada por todos, la infección por SARS-CoV-2, más conocida como la COVID-19. Ahora todos somos más conscientes del valor que tiene el trabajo que desarrollan los sanitarios, y de la importancia de tener un sistema sanitario de calidad en el país. La pandemia desestructuró todo el sistema sanitario de todos los países, donde los hospitales alcanzaron unos niveles de saturación nunca antes vistos.

1.1. Presentación del problema

En el presente trabajo se va a abordar la gestión del flujo de pacientes en el servicio de urgencias hospitalario (SUH). Según el Ministerio de Sanidad y Política social, el SUH puede definirse como una organización de profesionales sanitarios, ubicada en un centro hospitalario, que ofrece asistencia multidisciplinar, cumpliendo unos requisitos funcionales, estructurales y organizativos, de forma que garantiza unas condiciones adecuadas de seguridad, calidad y eficiencia para atender a las urgencias y emergencias. El SUH se configura como una unidad intermedia, que presta servicios (asistencia médica, cuidados de enfermería) hasta la estabilización del cuadro clínico a los pacientes que son finalmente ingresados en el hospital, y como un servicio final para aquellos pacientes que, habiendo acudido a la unidad, son finalmente dados de alta [2].

El problema que existe es común para cualquier SUH del país, y es que todos ellos están saturados, con una carga asistencial que supera incluso el 40% en algunas CC.AA [3]. Esto se debe fundamentalmente a que los SUH cada vez tienen que atender a un mayor volumen de pacientes. La saturación del SUH es un problema que afecta a la mayoría de los países, independientemente de su nivel socioeconómico. No obstante, la situación de sobrecarga en urgencias es un problema de todo el hospital, y no exclusivo del SUH, aunque es aquí donde se pretende hacer hincapié. Las causas más frecuentemente implicadas en la saturación de urgencias son las siguientes [2]:

- Incremento en la demanda de asistencia al SUH: una de las principales causas de la saturación es el incremento inesperado y de gran volumen en el número de pacientes que buscan atención médica en el SUH. Esta situación se da por ejemplo en epidemias, brotes de enfermedades (como la gripe), desastres naturales o cualquier otro evento que provoque un aumento de la necesidad de atención médica urgente en las personas.
- Atención primaria escasa e inadecuada: la escasez de un servicio de atención primaria de calidad y puntual hace que muchas personas acudan al SUH para poder recibir atención médica, una atención que bien podría haberse desarrollado en un ambulatorio u otro centro de atención primaria. La escasez de médicos y de clínicas de atención primaria en el país, así como la dificultad para obtener citas médicas en un horizonte temporal razonable, hace que muchos pacientes acudan al SUH para recibir atención médica cuanto antes.
- Incremento del número de pacientes con enfermedades crónicas: es un hecho que la población se ha envejecido en los últimos años, lo que junto a la prevalencia de enfermedades crónicas (por ejemplo, la diabetes) han contribuido a la saturación del SUH en esta tipología de pacientes. Los pacientes que presentan este marco crónico pueden precisar atención recurrente y su estado puede empeorar en cuestión de segundos, lo que conlleva que también colaboren en la saturación de los SUHs.
- Insuficiencia de personal y de recursos: la falta de personal médico, enfermeros, celadores y otro personal de apoyo dentro del SUH, junto con la falta de camas, equipos médicos y suministros, limitan la capacidad de un SUH para atender los pacientes de forma efectiva y eficiente. El personal del SUH suele estar sobrecargado de trabajo, lo que afecta de forma directa a la calidad de la atención, mientras que la falta de recursos provoca grandes tiempos de espera. La falta de camas de hospital es en muchos estudios la causa fundamental de la saturación de un SUH.
- Retrasos durante el proceso de atención al paciente: la demora en la evaluación, diagnóstico y posterior tratamiento de los pacientes que acuden al SUH generan una gran acumulación de pacientes en el SUH, que provoca a veces que algunos pacientes incluso abandonen el SUH antes de ser atendidos. Esto se puede deber a muchos factores entre los que se encuentran la falta de especialistas, procedimientos que implican otros departamentos del hospital o la demora en la obtención de resultados por el mal procesamiento de los mismos.
- Pacientes no urgentes: son muchos los pacientes que acuden al SUH para recibir atención médica que no es urgente, o que podrían haberse abordado desde otros entornos, como por ejemplo clínicas de atención primaria, o que ni siquiera precisaban de atención médica. Hay muchas personas que acuden al SUH con el mínimo síntoma, provocando que la carga de trabajo aumente innecesariamente y retrasando la atención de aquellos pacientes que realmente si precisan de atención médica urgente.

La frecuentación de la población al SUH (hospitales de agudos y de larga estancia) es de 0,50 veces por persona y año, lo que supone un total de 23.602.940 urgencias atendidas en 2019 [4]. La frecuentación urgente en el nivel de atención especializada muestra una clara tendencia al alza desde 2012, y entre 2010 y 2019 ha aumentado en todos los territorios, excepto en Navarra [4].

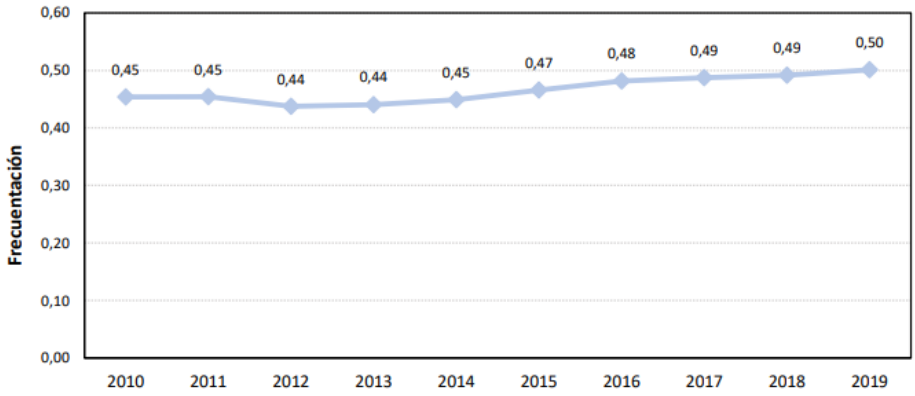

Figura 1-1. Frecuentación en urgencias hospitalarias en el SNS en España. [4]

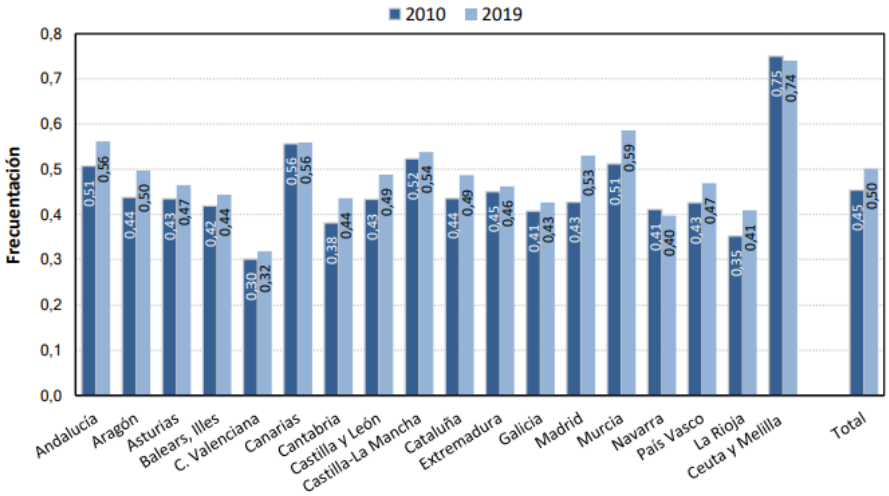

Figura 1-2. Frecuentación en urgencias hospitalarias en el SNS en España según CC.AA [4]

No obstante, la situación actual de saturación en los SUH, se puede abordar desde distintas soluciones que se citan a continuación [5]:

- Fortalecer el sistema de atención primaria: dar mayor accesibilidad a los centros de atención primaria, así como aumentar la calidad del servicio en dichos centros, pueden ayudar a reducir la alta demanda del SUH. Para ello, el aseguramiento de suficientes médicos de familias y centros de primaria para toda la población con el fin de aumentar la capacitación de todos los centros de atención primaria son factores claves en esta solución. Asimismo, se debe concienciar a la población de acudir al SUH solo y únicamente cuando sea necesario, y buscar sino la solución a su dolencia en otros medios.
- Implementación de un sistema de triaje eficiente: un sistema de triaje preciso y eficiente puede ayudar a identificar la gravedad del paciente; y, por tanto, el proceso de urgencia que seguirá dentro del SUH y los recursos que le serán asignados, de forma muy rápida y efectiva. Esto garantizaría además el hecho de que los pacientes más críticos reciban atención inmediata, mientras que los casos menos urgentes serían atendidos dentro de un tiempo estándar razonable. Aquí está la clave de muchos SUH, que ya tienen implementado un sistema de triaje, pero no está bien depurado o no se utiliza correctamente.
- Gestión eficiente de la demanda: implementar estrategias en la gestión de la demanda puede descongestionar el SUH. Para ello, soluciones tales como servicios de atención telefónica de asesoramiento de pacientes (que también determine si es oportuna la asistencia a urgencias del paciente), el establecimiento de citas programadas para pacientes no urgentes, o el desarrollo de programas de educación en salud para concienciar a la población, son claves para conseguir disminuir la saturación del SUH.
- Optimización de la gestión de recursos: muchas veces se dan tiempos ociosos en los recursos que

componen un SUH. Esta solución va de la mano con la gestión de la demanda, de modo que si se conocen los pacientes que hay en el SUH, hay que tratar de asignar el personal médico a los recursos de forma que en una situación ideal se produjera un flujo continuo de pacientes entre los recursos. Hay muchas investigaciones abiertas en este ámbito, pero la optimización de recursos y del flujo de pacientes del SUH es un problema de una complejidad muy elevada.

- ❖ Mejorar la coordinación entre departamentos: coordinar la conexión del SUH con otros servicios del hospital es difícil debido a la alta fluctuación de tipología y números de pacientes que presenta diariamente un SUH. Sin embargo, una buena coordinación interdepartamental puede agilizar los procesos de urgencia y reducir los tiempos de espera. Así pues, facilitar la accesibilidad a pruebas con especialistas o la transferencia de pacientes entre servicios puede suponer la reducción de demoras innecesarias.
- ❖ Promover la integración de tecnologías de la información: la implementación de sistemas electrónicos de registro de pacientes y la interoperabilidad de datos entre diferentes sistemas de salud puede mejorar la eficiencia y la comunicación en el SUH. Esto facilita el acceso a información clínica relevante, agiliza los procesos de atención y reduce los errores.
- ❖ Sistemas de integración de la información: la tecnología sigue creciendo y desarrollándose, y debe ser trasladada a todos los SUH del país. Implementar sistemas electrónicos de registro de pacientes que fomente la transmisión instantánea de datos entre todas las áreas del hospital conlleva mejorar la eficiencia y la comunicación de cualquier SUH. Además, se incrementa así el acceso a toda la información clínica de pacientes que muchas veces se pierde, agiliza el proceso de urgencia y reduce los errores humanos que se pueden dar dentro del SUH.

De todas las causas anteriormente citadas que derivan en la saturación de los SUH, en este proyecto se abordarán la sobrecarga de personal y de recursos, el aumento de la demanda y las demoras en el proceso de atención. Para ello, se han diseñado y desarrollado modelos matemáticos de programación lineal en el lenguaje de programación Python. De este modo, a través de modelos de programación lineal se intentará conseguir una mayor eficiencia en la utilización de los recursos disponibles, teniendo en cuenta que los pacientes más críticos tienen que recibir atención inmediata, y que se prima que un paciente permanezca dentro del hospital (en el sistema) lo menos posible.

1.2 Ámbito de aplicación

Este proyecto se desarrolla en el marco de un SUH de un hospital cualquiera con una configuración de recursos conocida. No obstante, la experimentación presentada en el trabajo resultará de los modelos propuestos en del SUH propuesto en el artículo "*Evaluating alternative resource allocation in an emergency department using discrete event simulation*" de Leonardo Bedoya-Valencia y Emre Kirac [6]. En este artículo, se propone un número de recursos según horarios del hospital, ruta de cada paciente según prioridad, datos de frecuentación al SUH de pacientes según prioridad, distribuciones de tiempos que toma cada actividad para cada paciente según prioridad, etc.

1.3 Objetivos

El objetivo principal del presente TFG es la creación de modelos de programación lineal para optimizar el SUH de un hospital a nivel operativo, es decir, en lo referente a la capacidad y eficiencia con la que se gestionan los recursos y se brinda atención médica inmediata y especializada a los pacientes en situaciones de emergencia, tal y como se explicará en la sección 2 del presente trabajo. El desarrollo de dichos modelos de programación lineal tiene varios objetivos específicos, que se pueden enumerar de la siguiente forma:

- ✓ Asignación eficiente de recursos: se busca utilizar de manera óptima los recursos disponibles, como personal médico, equipos y suministros, para garantizar una atención adecuada a los pacientes de urgencias. El modelo de programación lineal puede ayudar a determinar la asignación óptima de estos recursos, maximizando su utilización y minimizando el despilfarro.
- ✓ Mejora de la capacidad de respuesta: los modelos pueden ayudar a reducir los tiempos de espera y

mejorar la capacidad de respuesta del servicio de urgencia. Al optimizar la programación de los recursos, se pueden identificar cuellos de botella y asignar personal y recursos de manera eficiente para minimizar los tiempos de espera y maximizar la atención médica por prioridades.

- ✓ Gestión de la demanda conocida: conocidos los pacientes que se tienen en el SUH en un instante de tiempo determinado, se quiere gestionar la asignación de cada paciente a los recursos convenientes para que sean atendidos lo antes posible y en el menor tiempo posible a lo largo de todo su proceso de urgencias.
- ✓ Mejora de la eficiencia operativa: al optimizar la asignación de recursos, se puede mejorar la calidad de la atención médica proporcionada en términos de tiempos de espera y de estancia de cada paciente dentro del SUH.

Estos objetivos contribuyen a brindar una atención médica más efectiva y oportuna a los pacientes que requieren atención urgente.

1.4 Sumario

El presente TFG está formado por seis capítulos y un anexo, que serán brevemente introducidos en este apartado:

Capítulo 1. En este primer capítulo se realiza una breve introducción del problema que se desea resolver, así como los objetivos del presente trabajo para tratar de abordar la situación de saturación que viven los SUH de los centros hospitalarios actualmente. Asimismo, se expone el ámbito de aplicación del trabajo.

Capítulo 2. En este capítulo se expone el marco teórico que concierne a la gestión del SUH. Para ello, inicialmente se manifiestan los antecedentes históricos de los servicios de urgencias, para posteriormente poder explicar los dos niveles que existen actualmente en los SUHs (táctico y operativo). Por último, se explica el proceso y configuración del SUH, así como los indicadores de calidad que existen para medir la eficiencia del SUH.

Capítulo 3. Contiene una descripción detallada del problema que se va a abordar, con la correspondiente explicación de la función objetivo y de los recursos que intervendrán en el problema, aclarando también las limitaciones de los mismos. Para una mejor comprensión, se realiza una pequeña ejemplificación del problema en cuestión.

Capítulo 4. En este capítulo se exponen los modelos matemáticos de programación lineal que se han creado y desarrollado en el presente TFG para el problema de saturación del SUH. Además, se explican todas las restricciones que intervienen en el modelo.

Capítulo 5. Experimentación. Aplicación de los modelos a las configuraciones de recursos propuestas en el artículo [6]. Se resolverán tres casos de estudio para distintos niveles de saturación con el fin de ver como responden los modelos creados ante distintas configuraciones de recursos y niveles de saturación. Elección del mejor de los dos métodos de programación utilizados en el modelado del problema.

Capítulo 6. Conclusiones. A partir de los resultados de la experimentación, concluir que modelos responden mejor a las necesidades, cuales no consiguen dar una respuesta buena, y bajo que configuración de recursos y nivel de saturación los modelos tienen mejor rendimiento. Líneas de futuro.

2 GESTIÓN DEL SERVICIO DE URGENCIAS HOSPITALARIO

La gestión del SUH inherentemente compleja. La alta demanda, la variabilidad de los casos, los tiempos de espera impredecibles y la necesidad de asignar recursos de manera eficiente son solo algunos de los desafíos que enfrenta. Además, se requiere una coordinación precisa entre médicos, personal de enfermería, servicios de diagnóstico y otros equipos para garantizar una atención médica de calidad y oportuna. La complejidad radica en encontrar el equilibrio adecuado entre la capacidad operativa, la calidad del servicio y la satisfacción del paciente, todo ello en un entorno dinámico y en constante cambio. En este capítulo se describen los aspectos más relevantes sobre la gestión del SUH.

2.1 Antecedentes

En España, el primer SUH registrado con este nombre se creó en Barcelona en 1970 en el hospital de Santa Creu i Sant Pau. Desde este primer inicio, los servicios de urgencia hospitalario SUHs en el país han vivido muchos avances en todos los ámbitos, desde su organización, hasta los protocolos de atención, la tecnología disponible para atender a pacientes, o la forma de gestionar los recursos del SUH [7]. El momento de saturación que están viviendo los SUH hace que esta evolución se siga produciendo actualmente ante la necesidad de buscar solución a los problemas de incremento de demanda, la mejora de la eficiencia del SUH y la búsqueda del aumento de la calidad en las situaciones de emergencia [8].

Así pues, la gestión del SUH ha sido objeto de atención y estudio durante varias décadas debido a su importancia en la atención médica de emergencia [7]. Desde epidemias, enfermedades crónicas, el crecimiento demográfico y por ende el envejecimiento de la población. Son muchos los factores que han hecho que se invierta tiempo de investigación y desarrollo de los SUHs [7]. Los avances en la tecnología médica también han tenido un impacto significativo en la gestión de los SUH. La mejora en las técnicas de diagnóstico, los equipos médicos y las opciones de tratamiento ha permitido una atención más precisa y eficaz [7]. Sin embargo, la incorporación de estas tecnologías en los SUHs también ha jugado un papel inverso con efectos negativos ante la inexperiencia de muchos trabajadores en los nuevos medios que se han ido implementando a lo largo de la historia [9].

Por otro lado, también las políticas de salud y las regulaciones gubernamentales también han influido en la forma en que se gestionan los SUHs. Y es que los gobiernos y las autoridades sanitarias han implementado políticas que han supuesto grandes cambios en los enfoques de gestión, como la implementación de protocolos de triaje y la mejora de la coordinación entre diferentes profesionales de la salud. Asimismo, se ha observado un enfoque creciente en la mejora continua de los SUHs. La gestión de la calidad se ha convertido en un componente fundamental, con la implementación de programas como Lean Six Sigma y otras metodologías de mejora continua [3]. Estas iniciativas buscan identificar y abordar los puntos débiles en la gestión del SUH, con el objetivo de optimizar la eficiencia operativa, reducir los tiempos de espera, mejorar la satisfacción del paciente y garantizar una atención de alta calidad.

La base de la gestión del SUH que se tiene actualmente está sustentada por metodologías y prácticas que llevan en desarrollo desde muchos años atrás y que a día de hoy siguen siendo pulidas y mejoradas. A continuación, se presentan algunas de ellas [10] [11]:

- Servicio basado en roles: todo el personal del SUH tenía asignado un rol determinado que desempeñaba una serie de tareas conocidas. Así pues, el personal médico quedaba dividido en médicos, enfermeros, personal administrativo… y cada uno desempeñaba su función dentro del SUH. De esta forma, se simplifica el problema de organizar es flujo de trabajo y se consigue una mayor especialización del personal médico en las tareas que debe realizar.

- Modelo de triaje: el triaje es el primero de los procesos del SUH llevado a cabo por principalmente por el personal de enfermería, en el que el paciente es registrado dentro de la base de datos del SUH siendo aquí identificado y asignado una prioridad según la gravedad que presente. Los pacientes quedan así divididos en distintos niveles de prioridad en función de síntomas y gravedad que presentan, siendo mucho más fácil tratar de dar un buen nivel de servicio dependiendo de la tipología de pacientes que si hubiera que hacerlo para todos los pacientes que acuden al SUH por igual. Además, este modelo facilita la asignación de recursos y optimización de los mismos.
- Equipos multidisciplinarios: cuando ya existía la división de personal en roles, se avanzó creando equipos que combinaban los distintos roles, de forma que en un mismo equipo (compuesto por médicos, enfermeros, especialistas, celadores…), un paciente podría recibir atención completa pasando solo por un único equipo de trabajo. Estos equipos trabajan conjuntamente desde la evaluación y diagnóstico hasta el tratamiento y el alta del paciente, permitiendo dar un mejor servicio al paciente siendo atendido de forma más centrada y especializada dentro de un mismo equipo de trabajo.
- Gestión y planificación de los recursos: cuando se habla de gestionar un SUH, se habla de planificar rigurosa y meticulosamente todos los recursos disponibles en un intervalo de tiempo dentro del SUH. Dicha gestión se lleva haciendo desde hace muchos años, aunque siempre se encuentra en proceso de mejora ya que es un problema muy complejo, pero gracias a la tecnología cada vez se consigue avanzar más. Anteriormente, se trataba de asignar pacientes con determinadas actividades a recursos disponibles utilizando métodos muy rudos.
- Protocolos y directrices: sel establecimiento de protocolos y directrices clínicas para abordar situaciones comunes y emergencias médicas específicas siempre ha estado presente. Es la base de la gestión del SUH, ya que si todo está estandarizado, la gestión se simplifica significativamente. Estos protocolos proporcionaban pautas claras y consistentes para el manejo de casos, lo que permitía una atención estandarizada y de calidad. Además, se establecían flujos de trabajo y procedimientos específicos para mejorar la eficiencia y la seguridad en el SUH.

Estas metodologías, que son ideas que fueron surgiendo progresivamente, siguen vigentes en todos los SUH actuales, aunque siempre se trata de mejorarlas con el fin de algún día llegar a una gestión del SUH óptima, aunque la optimalidad es de gran complejidad. Los avances tecnologías y las investigaciones vinculadas a la gestión del SUH ligadas a las necesidades de atención médica han supuesto la creación de enfoques más sofisticados y concretos de la gestión del flujo de pacientes y de la gestión de recursos, tales como la creación de modelos de programación lineal para optimizar el SUH a nivel operativo, objetivo del presente TFG.

2.2 Gestión del SUH

El problema de la saturación del SUH es algo de actual preocupación para el ser humano, por lo que actualmente existen muchas investigaciones en desarrollo con el fin de tratar de eliminar o al menos disminuir esta cuestión. Dichas investigaciones pueden desarrollarse a dos niveles dentro del SUH que se exponen a continuación.

2.2.1 Niveles de decisión en la gestión del SUH

2.2.1.1 Nivel táctico

En el nivel táctico de gestión de un SUH, se llevan a cabo actividades de planificación y supervisión a medio-plazo para mejorar la calidad y eficiencia de la atención. Este nivel implica el establecimiento de políticas, protocolos y estrategias que guían la gestión operativa y aseguran una atención de calidad [12]. Así pues, la gestión del SUH a nivel táctico abarca desde la planificación de la capacidad del mismo, la implementación de programas de mejora continua dentro del SUH, la estandarización de procesos y guías de actuación, la formación y entrenamiento del personal médico, y el cálculo y análisis de indicadores de gestión, todos ellos con el fin de mejorar la calidad, la eficiencia, la efectividad y la seguridad del servicio de urgencias.

En este nivel, se realiza la planificación de la capacidad del SUH, lo que depende de factores tales como la cantidad de recursos de los que se dispone para cada actividad, el personal médico y de enfermería disponible, y los equipos, maquinaria e instalaciones necesarios para brindar la atención necesaria a todos los pacientes que acuden al SUH. Para ello, analizar la demanda del SUH y determinar la capacidad necesaria de todos los recursos anteriormente mencionados son dos aspectos cruciales para optimizar el SUH a nivel táctico.

Otra actividad importante en el nivel táctico es la implementación de programas de mejora continua. Se busca identificar problemas recurrentes, optimizar los procesos y reducir los tiempos de espera en el servicio de urgencias [12]. Para conseguirlo se realizan análisis detallados de los flujos de trabajo, con los que se obtienen datos tales como el cuello de botella que está congestionando el SUH. Además se busca mejorar el nivel de satisfacción del paciente mediante la implementación de medidas orientadas a la atención veloz, eficiente y focalizada en el paciente [11].

En el nivel táctico también se desarrollan protocolos y guías de actuación basados en evidencia científica. Estos protocolos establecen los pasos a seguir en el manejo de diferentes condiciones médicas y situaciones de urgencia. La implementación de protocolos estandarizados asegura una atención coherente y de calidad, minimizando la variabilidad en la práctica clínica y garantizando la seguridad del paciente. La formación y capacitación del personal del servicio de urgencias también es un aspecto fundamental en el nivel táctico. Se promueve la actualización de conocimientos y habilidades necesarios para brindar una atención eficaz y segura. Esto puede incluir programas de formación continua, cursos especializados y talleres prácticos para mejorar las competencias clínicas y de liderazgo del personal [13]. Además, en el nivel táctico se lleva a cabo el análisis de indicadores de gestión, los cuales permiten evaluar el desempeño del servicio de urgencias, identificar áreas de mejora y realizar ajustes o modificaciones en los procesos de atención según sea necesario.

2.2.1.2 Nivel operativo

En el nivel operativo de gestión de un SUH, se llevan a cabo actividades diarias y a corto plazo para garantizar un flujo eficiente de pacientes y una atención de calidad. Este nivel se enfoca en la ejecución y coordinación de las tareas y procesos operativos necesarios para brindar una atención oportuna y adecuada a los pacientes que acuden al servicio de urgencias [10].

En dicho nivel, se realiza la gestión del flujo de pacientes en el SUH. Esto implica coordinar y organizar el proceso de triaje, asignando la prioridad adecuada a cada paciente en función de la gravedad de su condición médica. Se realiza el registro de los pacientes, se les asigna una cama o una sala de espera según su clasificación de urgencia, y se los deriva a las áreas correspondientes para su evaluación y tratamiento [13].

Hay muchos métodos para la evaluación y determinación de la prioridad asistencial de cada paciente. En España se utiliza el Sistema Español de Traje, SET, el cual se compone de un sistema informático de ayuda a la toma de decisión de asignación de una prioridad a un paciente. Se establecen 5 niveles fundamentales de prioridad asignables a cada paciente que entre al registro, y según el nivel que tenga tendrán que ser atendidos con mayor o menor brevedad. No obstante, el nivel de prioridad puede cambiar a lo largo del proceso de urgencias del paciente, en caso de que algún médico así lo estime oportuno, y con ello cambiaría el proceso de urgencia de dicho paciente. Así pues, los pacientes son clasificados en 5 niveles de prioridad, siendo los pacientes de prioridad 1 los que presentan mayor gravedad (deben ser atendidos de forma inmediata), mientras que los pacientes de prioridad 5 no presentan carácter urgente.

En este nivel se lleva a cabo la gestión de los recursos disponibles en el SUH. Por un lado, la gestión de los recursos humanos, asegurando la disponibilidad y asignación adecuada del personal médico, de enfermería y de apoyo para garantizar una atención continua y eficiente. Además, se fomenta la comunicación y coordinación entre los diferentes profesionales de la salud para asegurar una atención integral y multidisciplinaria [13].

Por otro lado, se lleva a cabo la gestión de los recursos materiales y equipos necesarios para el funcionamiento del SUH, lo que implica mantener un inventario adecuado de suministros médicos y equipos, así como gestionar su correcta utilización y mantenimiento. También se realizan actividades de control de calidad y aseguramiento de la seguridad en el uso de los equipos y dispositivos médicos [13].

La monitorización de los tiempos de espera y los tiempos de atención es fundamental en el nivel operativo. Se registran y analizan los tiempos que los pacientes pasan en el servicio de urgencias, desde su llegada hasta su atención médica y posterior disposición (alta hospitalaria, ingreso o derivación a otro servicio). Esto permite

identificar posibles cuellos de botella y realizar ajustes para mejorar la eficiencia y reducir los tiempos de espera [14].

Además, en el nivel operativo se implementan medidas de control de infecciones y de seguridad del paciente. Se promueve el cumplimiento de las normas y protocolos de higiene, se monitorea la limpieza y desinfección de las áreas y se promueve la seguridad en la administración de medicamentos y en la realización de procedimientos médicos [14]. De este modo, el nivel operativo de gestión en un servicio de urgencias hospitalario implica la gestión del flujo de pacientes, la coordinación de los recursos humanos y materiales, la monitorización de los tiempos de espera y de atención, y la implementación de medidas de control de infecciones y seguridad del paciente [13].

Muchos son los indicadores para medir la eficiencia del SUH a nivel operativo que se pueden encontrar en [2], no obstante, los más importantes son los siguientes [6]:

- ✓ *Length of Stay* (LOS): tiempo total que un paciente pasa dentro del SUH, es decir, desde que es atendido en triaje hasta que recibe el alta o pasa a ser ingresado en el hospital.
- ✓ *Time to Be Seen By a Physician or Physician Assistant* (TBSPPA): tiempo que pasa desde que el paciente es registrado con una prioridad en triaje, hasta que es atendido por primera vez en 1ª Consulta. En España, este indicador se denomina TEPCOF.
- ✓ *Left Without Been Seen* (LWBS): número de pacientes que abandonan el SUH sin ser atendidos.

Dada la complejidad del problema de secuenciación de pacientes, actualmente existen numerosos estudios con distintas metodologías para abordarlo tales como modelos de programación lineal, programación dinámica, heurísticas constructivas o metaheurísticas [14].

Este es el nivel que se abordará en este TFG, donde se analizará la situación del SUH en un instante de tiempo determinado y se intentará dar una solución óptima al mismo en el menor plazo posible. En este trabajo, se pretende crear modelos de programación lineal en los que el objetivo sea cumplir con los valores establecidos por la dirección del hospital de algunos de los indicadores anteriormente mencionados.

2.2.2 Configuración del SUH

Todo hospital cuenta con un SUH, al que los pacientes pueden llegar a pie, en ambulancia, trasladado por la policía como emergencia… La tasa de llegada de pacientes es una demanda plenamente estocástica, que varía diariamente y a cada hora. Es cierto que hay franjas horarias en las que se puede afirmar una mayor afluencia de pacientes en el SUH, pero cuantificar de manera certera resulta algo imposible, por lo que es común recurrir a distribuciones estadísticas (por ejemplo, en [6] se utiliza una distribución de Poisson $\lambda(t)$ con la que se puede aproximar la tasa de llegada al SUH). En este mismo artículo, se estudió en que porcentaje los pacientes llegaban de una forma u otra al SUH (ver Figura 2-1).

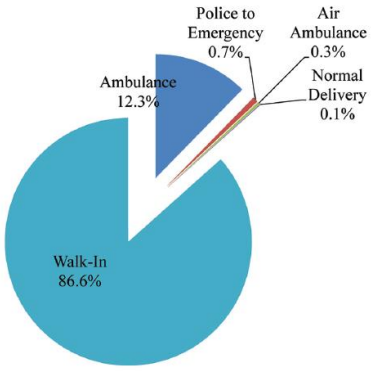

Figura 2-1. Tipología de llegada al SUH [6]

A continuación, se muestra una ilustración esquemática del proceso que un paciente sigue a lo largo de toda su estancia dentro del SUH:

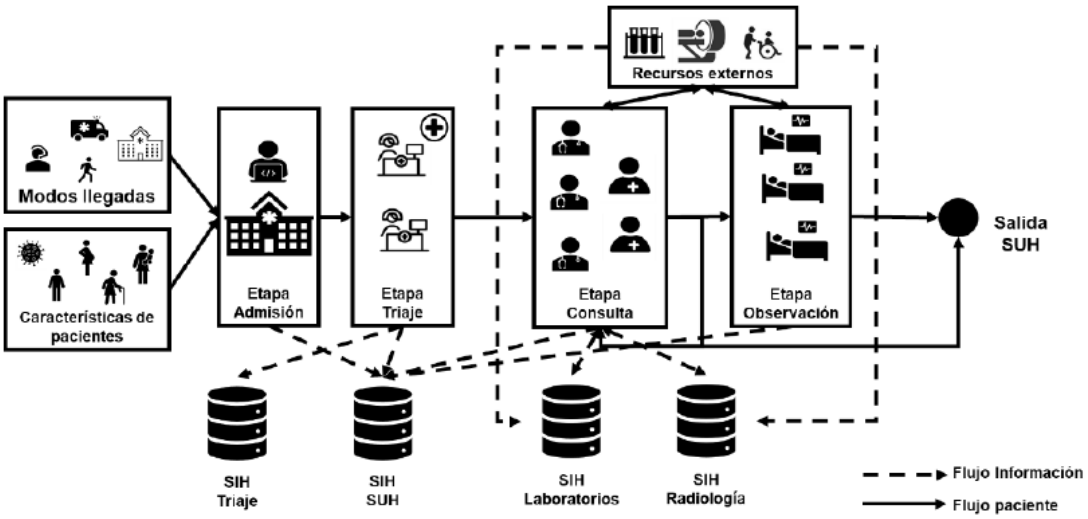

Figura 2-2. Flujo de información y movimiento del paciente dentro del SUH [21]

En primera instancia, todos los pacientes pasan un proceso de admisión en el que el personal de la recepción del SUH recoge todos los datos del paciente en cuestión necesario para su registro en el sistema. Posteriormente, el personal de triaje, que habitualmente es personal de enfermería, realizan una primera evaluación del paciente, sin ni siquiera ver al mismo. Esta evaluación se basa en distintos parámetros tales como el motivo de ingreso en el SUH, estado del paciente, forma de llegada, entre otros. El objetivo principal de esta evaluación es poder elaborar una jerarquía de pacientes, en la que en base a parámetros estandarizados se pueda generar una estructura de los pacientes por prioridad asistencial, ya que no tiene la misma urgencia a ser atendido una persona resfriada que una con dolores crónicos.

No se puede hablar de prioridad asistencial sin hacer referencia al TEPCOF (Tiempo de Espera de Primera Consulta Facultativa), parámetro estandarizado en el que se establece un tiempo máximo según la prioridad del paciente en que este debe ser atendido en primera consulta por el médico correspondiente. Así pues, para pacientes de alta gravedad, dicho tiempo es nulo. De este modo, tomando los datos de TEPCOF oficiales publicados por la Junta de Andalucía [15], se muestran a continuación el tiempo en que los pacientes deben ser atendidos como máximo para una primera evaluación por un profesional:

Prioridad	Nivel de Urgencia	TEPCOF
P1	Emergencia	0
P2	Muy urgente	15
P3	Urgente	60
P4	Poco urgente	100
P5	No urgente	120

Tabla 2-1. TEPCOF según prioridad del paciente [15] [14]

Una vez priorizado, el paciente es evaluado en una 1ª Consulta Facultativa, que debe de producirse en un tiempo menor al TEPCOF asignado a esa prioridad. A partir de aquí, el paciente puede ser sometido a distintas pruebas como radiología, extracción de sangre, TAC, o cualquier otra prueba que el médico estime oportuno para poder diagnosticar al paciente. Finalmente, el paciente será reevaluado de nuevo por un médico, que determinará si el paciente debe ingresar en planta en el hospital, o si por el contrario el paciente recibe el alta y abandona el hospital. En ambos casos, el paciente sale del sistema del servicio de urgencias, ya sea por irse a casa o por pasar al sistema de ingresos del propio hospital.

3 DESCRIPCIÓN DEL PROBLEMA

En este apartado se procede a poner en contexto del problema concreto que se pretende abordar. Así pues, es necesario conocer en que entorno se va a desarrollar el problema, qué recursos se deben considerar, las limitaciones de dichos recursos; y, qué función objetivo se tiene con el fin de optimizar la secuenciación del proceso de urgencia de los pacientes del SUH.

3.1 Descripción del entorno

El presente trabajo que aborda la secuenciación del proceso de urgencia es equiparable al entorno de trabajo Jobshop [14]. En este entorno se tienen *m* máquinas y *n* trabajos, con la peculiaridad de que cada trabajo puede seguir una ruta distinta, y que una máquina puede desarrollar más de una tarea (son máquinas multidisciplinares). Esta flexibilidad sin apenas restricciones hace que la programación óptima de las n tareas en las m máquinas sea habitualmente imposible.

Estas máquinas y trabajos mencionados, se pueden trasladar al ámbito sanitario, de modo que las máquinas serían las consultas (cada consulta tiene siempre asignado un médico o médico asistente), el personal de enfermería, salas de rayos, entre otros, y los trabajos serían los pacientes, que según la prioridad seguirán una ruta por unas máquinas u otra. De este modo, un mismo recurso, por ejemplo la consulta, puede llevar a cabo dos actividades, la 1ª consulta y la posterior reevaluación.

Siguiendo con la equivalencia entre el Jobshop y el problema planteado, se mencionan a continuación los recursos (máquinas) de los que se dispone en un SUH [14]:

- ✓ Consulta
- ✓ Enfermería
- ✓ Sala de rayos
- ✓ Laboratorio
- ✓ Sala TAC
- ✓ Celadores
- ✓ Camas de observación

Asimismo, las principales actividades que se desarrollan a lo largo del proceso de urgencia de cualquier paciente son:

- ✓ 1ª Consulta facultativa
- ✓ 1ª Enferemería
- ✓ Pruebas radiodiagnósticas (TAC, Rayos X…)
- ✓ Traslado del paciente por medio de celadores
- ✓ Analíticas
- ✓ Reevaluación
- ✓ Alta

Dichas actividades se pueden subdividir en más, por ejemplo, en enfermería normalmente hay dos vistitas, una al comienzo y otra justo antes de abandonar el SUH [6]. O en el caso de Rayos X o el TAC, muchas veces las salas o recursos donde se llevan a cabo son distintas áreas del SUH. En la experimentación de este TFG que se expondrá más adelante, se tomarán datos y recursos concretos apoyados en el artículo [6]. Así pues, para cada paciente se tiene un proceso de urgencia dentro del SUH según la prioridad asignada al mismo.

3.2 Limitaciones de los recursos

Cada hospital tiene una organización distinta, pero lo más frecuente es que todos los médicos (doctores, enfermeros, celadores…) tengan una jornada laboral de 8 horas/día durante 5 días a la semana, a lo que habría que sumar días de guardia, vacaciones… Aunque esto es algo que no concierne al problema que se quiere abordar en este proyecto. Lo relevante es que aunque se tengan determinados recursos, hay que tener en cuenta los horarios del personal y en que momentos se puede contar con ese recurso para llevar a cabo un trabajo, porque en caso contrario no se podrá asignar a un paciente a ese recurso.

Por un lado, se puede dar el caso de que los pacientes más graves (menor prioridad) tengan que ser atendidos por los médicos de más experiencia, mientras que los de menor gravedad pueden ser atendidos por médicos asistentes o en prácticas. De esta forma, el SUH queda sectorizado según las prioridades que presenten los pacientes.

Por otro lado, con la reciente pandemia vivida más aún, también hay que tener en cuenta posibles situaciones que se pueden dar en las que debido a un alto riesgo de contagio, hay salas que quedan totalmente inhabilitadas para pacientes no contagiados (o contagiados), por lo que tampoco se puede asignar pacientes a recursos si no están o si están contagiados. Estas variantes serán explicadas más adelante.

Asimismo, también se puede dar el caso de que la política del hospital sea que los pacientes de gravedad elevada, necesiten doble asistencia simultáneamente, o que sean atendidos por los médicos más expertos y preparados para ello, mientras que los menos grave pueden ser atendidos por médicos en prácticas o con menos experiencia. Ello también limita la asignación de pacientes a recursos, al igual que la disponibilidad o la tipología de pacientes [6].

Por último, otro aspecto a tener en cuenta es puede darse el caso de que los médicos más preparados o con más experiencia sean los encargados de realizar la primera consulta, mientras que los que tienen menos experiencia realicen la reevaluación o el alta del paciente. Esta sectorización limita los recursos de forma que haya consultas en las que solo se puede llevar a cabo las tareas de primera consulta facultativa, mientras que otras solo realizan altas o reevaluaciones.

3.3 Función objetivo

Se pueden distinguir dos términos fundamentales que se pretenden minimizar con el fin de optimizar el flujo de pacientes dentro del SUH. Para conseguir la mayor eficiencia posible, se debe minimizar el tiempo que un paciente pasa dentro del SUH, de modo que cuanto menos tiempo pase quiere decir que menos tiempo ocupará los recursos disponibles y por tanto más disponibilidad habrá para el resto de pacientes.

No obstante, la minimización del tiempo de estancia dentro del SUH debe venir acompañada de cumplimiento del llamado TEPCOF, de modo que según la prioridad del paciente, este haya sido atendido dentro del tiempo máximo prefijado por el parámetro estándar. A modo de resumen, los dos indicadores sobre los que se va a trabajar y que se desean minimizar son:

- Cumplimiento del TEPCOF
- Length Of Stay (LOS)

3.4 Ejemplificación del problema

A modo de explicación gráfica, y con la ayuda del lenguaje de programación Python, se ha tomado una instancia pequeña con datos cualesquiera, con el fin de mostrar visualmente la secuenciación del proceso de urgencia de cada paciente en el SUH. Veamos los datos del problema para una mejor comprensión:

Actividades	Tipos de recurso	Recursos
(1) 1ª Consulta	Consulta	Consulta 1
(2) Enfermería	Enfermería	Consulta 2
(3) Rayos	Rayos	Consulta 3
(4) Reevaluación		Enfermería 1
(5) Alta		Enfermería 2
		Rayos

Tabla 3-1. Datos para el problema de ejemplificación (1)

Paciente	**Prioridad**	**Ruta**	**Fecha de llegada**
1	1	1 – 2 – 3 – 4 – 5	6
2	3	1 – 2 – 5	10
3	3	1 – 2 – 5	1
4	4	1 – 2 – 5	20

Tabla 3-2. Datos para el problema de ejemplificación (2)

Minimizando las variables LOS y TEPCOF, la ruta óptima para estos cuatro pacientes dentro del SUH queda de la siguiente forma:

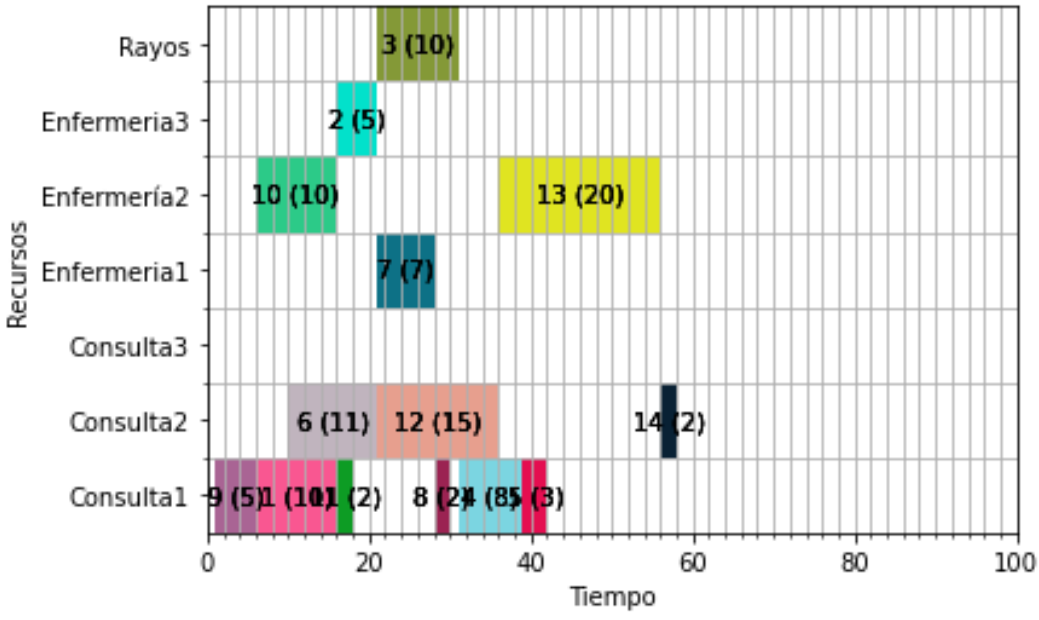

Figura 3-1. Diagrama de Gantt I del problema de ejemplificación. Fuente: propia

Las actividades correspondientes al paciente 1 van desde la actividad 1 a la 5, para el paciente 2 son las actividades 6, 7 y 8; para el paciente 3 son las actividades 9, 10 y 11; y, por último, para el paciente 4 las actividades son la 12, 13 y 14.

Como se puede observar, las fechas de llegada de cada uno de los pacientes se respetan, y el paciente 1 es atendido nada más llegar (instante de llegada al SUH t=6), debido a su prioridad asistencial. Además, para este ejemplo el único paciente que visita la sala de rayos es el paciente uno. Supongamos ahora que los pacientes 2 y 4 presentan COVID, de modo que solo pueden ser atendidos en las consultas 2 y 3, y en la enfermería 2, entonces el nuevo diagrama es el que se muestra en la Figura 3-2.

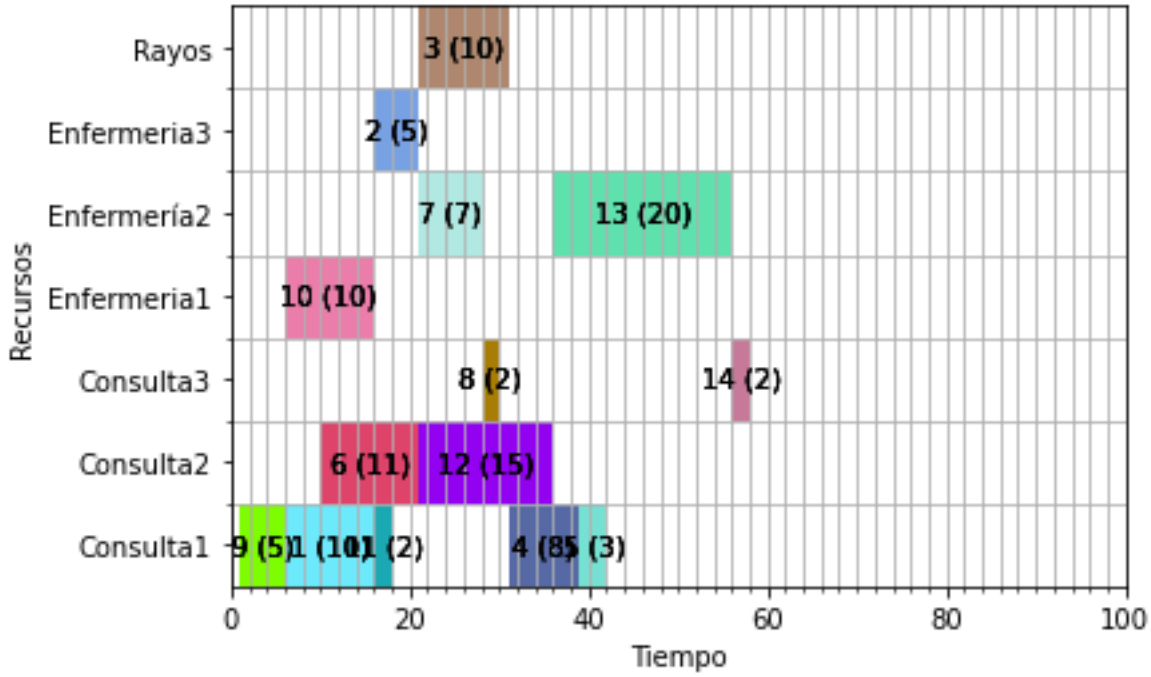

Figura 3-2. Diagrama de Gantt II del problema de ejemplificación. Fuente: propia

Como se puede apreciar, las actividades 6,7 y 8, y las actividades 12, 13 y 14, pertenecientes a los pacientes que presentan COVID respectivamente, se realizan en los correspondientes recursos destinados para ello. Supongamos ahora que debido a la gravedad que presenta el paciente 1, debe ser atendido por dos enfermeros en vez de uno:

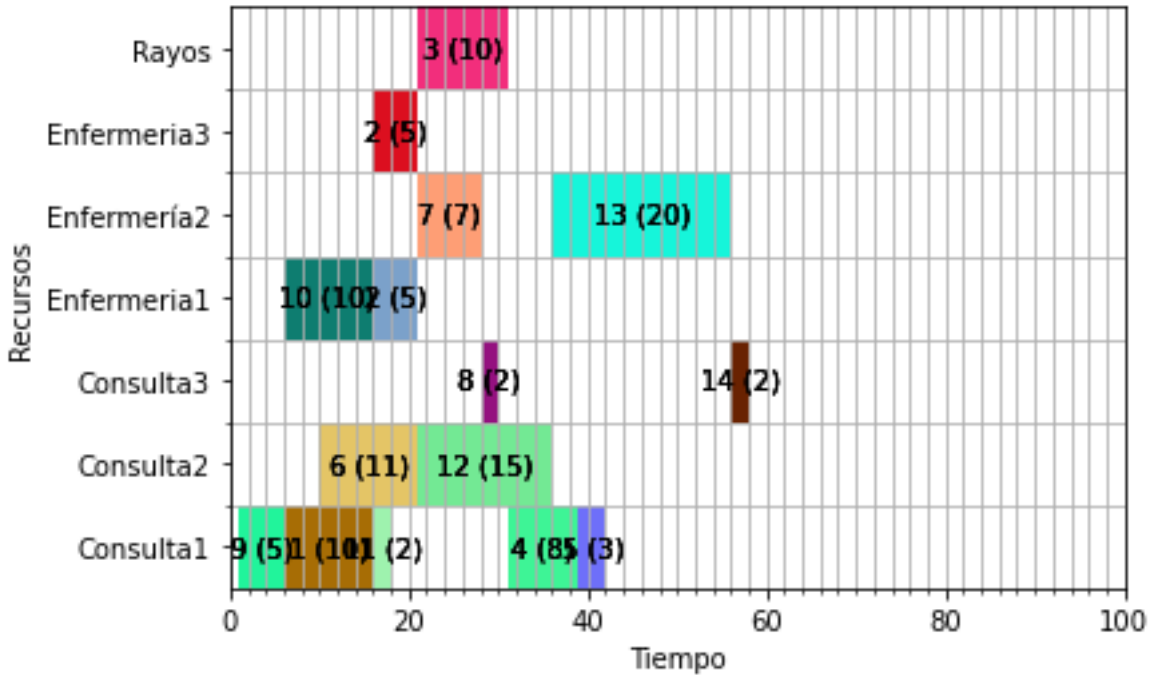

Figura 3-3. Diagrama de Gantt III del problema de ejemplificación. Fuente: propia

El cambio introducido hace que las enfermerías 1 y 3 estén ocupadas al mismo tiempo y por la misma actividad (actividad 2, enfermería del paciente 1; ver Figura 3-3). Finalmente, supongamos que ninguna de las tres consultas está disponible hasta el minuto 30, de modo que las fechas de llegada se convierten en irrelevantes:

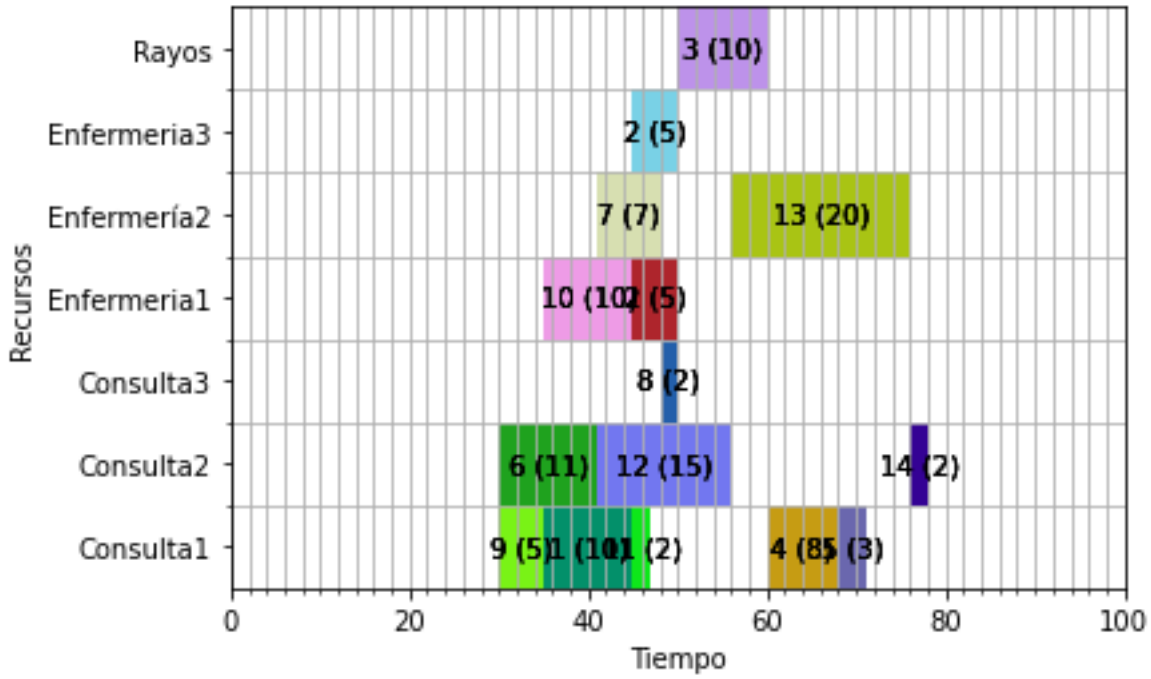

Figura 3-4. Diagrama de Gantt IV del problema de ejemplificación. Fuente: propia

Se puede ver (Figura 3-4) que ningún paciente recibe la diagnosis de un médico hasta al menos transcurridos los 30 primeros minutos del marco temporal. Muchas son las variantes que se pueden presentar a lo largo de una jornada laboral en un hospital, por lo que encontrar un modelo único que resuelva el problema existente con la saturación en los SUH es muy complejo debido a la fluctuación en la demanda y casuística presente en los hospitales diariamente.

4 MODELADO DEL PROBLEMA

En este capítulo se procede a la exposición y respectiva explicación de los modelos desarrollados de programación lineal entera para la secuenciación de pacientes del SUH. Para ello, en primer lugar, se describen los conjuntos, parámetros y variables decisión necesarios para la gestión del SUH. En segundo lugar, se explican todos los modelos diseñados y desarrollados en el TFG. Cabe destacar que se han considerado dos formas de modelar diferentes, relativas a la discretización o no del dominio de las variables de decisión. Para ambas metodologías, en primer lugar se expone un modelo básico, al cual se van sumando variantes reales que se pueden encontrar en el problema de gestión del SUH. Asimismo, para la programación e implementación de estos modelos se ha utilizado el lenguaje de programación Python.

4.1 Datos

4.1.1 Conjuntos

En primer lugar, se describen los conjuntos que definen el modelo de programación lineal:

Nomenclatura	Descripción
$i \in I$	Conjunto de pacientes
$j \in J$	Conjunto de recursos
$a \in A$	Conjunto de actividades
$p \in P$	Conjunto de posiciones
$w \in W$	Conjunto tipo de recurso
$k \in K$	Conjunto prioridad de pacientes
$c \in C$	Conjunto tipo de paciente
$o \in O$	Conjunto de actividades maestras

Tabla 4–1 Conjuntos e índices

El conjunto posiciones solo estará presente en una de los dos métodos de programación.

4.1.2 Parámetros

A continuación, se muestran todos los parámetros que componen el problema, es decir, los datos conocidos a partir de los cuales hay que optimizar los procesos de urgencia de cada uno de los pacientes:

Nomenclatura	Descripción
I	Número total de pacientes
$pu_{i,a}$	Actividad a del proceso de urgencia del paciente i
pp_i	Prioridad del paciente i
$first_i$	Primera actividad en el proceso de urgencia del paciente i
$last_i$	Última actividad en el proceso de urgencia del paciente i
r_tepcof_i	Fecha de llegada al SUH del paciente i
r_i	Tiempo que le queda al paciente i de la actividad que esté en curso en el instante en que se quiere resolver el problema
A	Número total de actividades
$maxA$	Número máximo de actividades que puede tener un paciente

	dentro del proceso de urgencia
pt_a	Tiempo de proceso de la actividad a
pa_a	Prioridad de la actividad a según paciente al que pertenezca
s_a	Tipo de recurso con el que se desarrolla la actividad a
am	Actividades maestras de las que se compone el proceso de urgencia
m_a	Actividad maestra a la que pertenece la actividad a
cv_a	Tipo de paciente al que pertenece la actividad a
J	Número total de recursos
t_j	Tipo de recurso al que pertenece el recurso j
$d_{j,2}$	Disponibilidad del recurso j (instante inicial / final)
$pr_{k,j}$	1 si el paciente de prioridad k se puede asignar al recurso j; 0 e.c.c.
$b_{o,j}$	1 si la actividad maestra o puede ser asignada al recurso j; 0 e.c.c.
$k_{c,j}$	1 si el tipo de paciente c puede ser asignado al recurso j; 0 e.c.c.
$n_{k,o}$	Número de recursos que el paciente de prioridad k necesita para ser atendido en la actividad maestra o
M	Número muy grande
$tepcof_k$	Tiempo máximo en que el paciente de prioridad k tiene que ser atendido en 1ª consulta desde su ingreso en el SUH
p	Número de posiciones máximo en un recurso para el segundo método de programación

Tabla 4–2 Parámetros

4.2 Modelos de programación lineal

Como se ha introducido anteriormente, el modelo y todas las variantes del mismo, van a ser implementadas desde dos formas de modelado distintas que son comúnmente observadas en la literatura. La clave de estas dos formas reside en la secuenciación de actividades dentro de los procesos de urgencias de todos los pacientes. En otras palabras, que en un recurso no pueda empezar una actividad hasta que la anterior no haya finalizado. Se presentan a continuación las diferencias:

- Modelo GAMMA (G): en este método hay una variable de decisión que indica si una actividad va antes que otra en un mismo recurso o no. En caso afirmativo, se modelará con restricciones para que el instante de inicio de la segunda actividad sea igual o superior al instante de finalización de la actividad predecesora en ese mismo recurso.
- Modelo POSICIONES (D): en este método, la variable de decisión anterior desaparece, pero la variable X depende de un nuevo parámetro p. De esta forma, como solo puede haber una única actividad asignada a una posición en un recurso, el solapamiento de dos actividades en un mismo recurso nunca se producirá. El valor de p se determina atendiendo a las actividades vinculadas a según que tipología de recursos, de modo que el valor de p es igual al mayor número de actividades asignadas a un mismo tipo de recurso.

El objetivo de programar de dos formas distintas es demostrar cuál de estas dos formas es más eficiente, atendiendo a cuantas variables hay en cada modelo, cuantas restricciones se generan, cuanto tiempo tarda cada modelo en encontrar el óptimo, o en caso de no encontrarlo, cuál de los dos se queda más cerca de dicho óptimo… Muchos son los datos en los que se puede basar el estudio para comprobar la eficiencia de ambos modelos, que será visto en más profundidad más adelante en este proyecto durante la sección 5.

4.2.1 Variables de decision

Conocidas ambas formas de modelar, se procede a exponer las variables de decisión que estarán presentes para dada uno de dichos métodos. Para el modelo G las variables son:

Nomenclatura	Descripción
$X_{a,j}$	1 si la actividad a está asignada al recurso j; 0 e.c.c.
$\gamma_{a,a',j}$	1 si la actividad a va delante de la actividad a' en el recurso j
$CT_{a,j}$	Instante de finalización de la actividad a en el recurso j
T_i	Tiempo que el paciente i ha tardado en ser atendido en 1ª consulta desde su ingreso en el SUH

Tabla 4–3 Variables de decisión para el primer método de programación

Se muestran ahora las variables de decisión de la segunda forma de programación (discretización del dominio de las variables de decisión):

Nomenclatura	Descripción
$X_{a,j,p}$	1 si la actividad a está asignada al recurso j; 0 e.c.c.
$CT_{a,j}$	Instante de finalización de la actividad a en el recurso j
T_i	Tiempo que el paciente i ha tardado en ser atendido en 1ª consulta desde su ingreso en el SUH

Tabla 4–4 Variables de decisión para el segundo método de programación

4.2.2 Modelo G-básico

Se muestra a continuación el modelo de programación lineal entera del problema planteado:

$$Min. \sum_{i=1}^{I} T_i + \sum_{i=1}^{I} \sum_{j=1}^{J} CT_{last_i,j} \qquad (1)$$

$$s.a: \sum_{j \in J \mid s_a = t_j} X_{a,j} = 1 \quad \forall \quad a \in A \qquad (2)$$

$$\sum_{j \in J \mid s_a \neq t_j} X_{a,j} = 0 \quad \forall \quad a \in A \qquad (3)$$

$$CT_{first_i,j} \geq \left(r_i + pt_{first_i}\right) \cdot X_{first_i,j} \quad \forall\, i \in I, \forall j \in J \mid s_{first_i} = t_j \qquad (4)$$

$$CT_{pu_{i,a},j} \geq CT_{pu_{i,a-1},j'} + pt_{pu_{i,a}} + M \cdot \left(X_{pu_{i,a},j} + X_{pu_{i,a-1},j'} - 2\right)$$

$$\forall\, i \in I, \forall\, a \in A, \forall\, j \in J, \forall\, j' \in J \mid pu_{i,a} \neq 0, t_j = s_{pu_{i,a}}, t_{j'} = s_{pu_{i,a-1}} \qquad (5)$$

$$CT_{a',j} + M \cdot \left(2 - X_{a,j} - X_{a',j}\right) \geq CT_{a,j} + pt_{a'} - M \cdot \left(1 - \gamma_{a,a',j}\right)$$
$$\forall a \in A, \forall\, a' \in A, \forall j \in J \mid a < a', t_j = s_a, t_j = s_{a'} \qquad (6)$$

$$CT_{a,j} + M \cdot \left(2 - X_{a,j} - X_{a',j}\right) \geq CT_{a',j} + pt_a - M \cdot \gamma_{a,a',j}$$
$$\forall a \in A, \forall\, a' \in A, \forall j \in J \mid a < a', t_j = s_a, t_j = s_{a'} \qquad (7)$$

$$CT_{a,j} \geq \left(pt_a + d_{j,1}\right) \cdot X_{a,j} \qquad \forall\, a \in A, \forall j \in J \mid s_a = t_j \qquad (8)$$

$$CT_{a,j} \leq d_{j,2} \cdot X_{a,j} \qquad \forall\, a \in A, \forall j \in J \mid s_a = t_j \qquad (9)$$

$$T_i \geq CT_{first_i,j} - pt_{first_i} - r_{tepcof_i} - tepcof_{pp_i}$$
$$\forall\, i \in I, \forall\, j \in J \mid m_{first_i} = 1, t_j = s_{first_i} \qquad (10)$$

$$X_{a,j} \in \{0,1\} \qquad \forall\, a \in A, \forall j \in J \qquad (11)$$

$$CT_{a,j} \geq 0 \qquad \forall\, a \in A, \forall j \in J \qquad (12)$$

$$\gamma_{a,a',j} \in \{0,1\} \qquad \forall\, a \in A, \forall\, a' \in A, \forall j \in J \qquad (13)$$

$$T_i \geq 0 \qquad \forall\, i \in I \qquad (14)$$

Tal y como se menciona anteriormente, la función objetivo consta de dos partes fundamentales. El primer término hace referencia al cumplimiento del TEPCOF, de modo que se minimiza el tiempo que pasa desde que el paciente llega a SUH hasta que es atendido en la primera consulta facultativa. Se explicará con mayor atención más adelante con la explicación de la restricción 9. Por otro lado, se quiere minimizar la estancia de todos los pacientes dentro del SUH, con lo que el segundo término de la función objetivo implementa la minimización del valor del instante de tiempo en que la última actividad (que es el alta para todos los pacientes, ya sea para proceder a ingreso o para irse a casa) de cada paciente finaliza.

El primer conjunto de restricciones (2) modela el hecho de que una actividad *a* siempre sea asignada a un único recurso *j*. Para ello, el recurso al que se asigne dicha actividad debe ser del mismo tipo de recurso que la propia actividad. Por ejemplo, para la actividad de reevaluación (tipo de recurso consulta) de un paciente cualquiera, si se disponen de tres consultas, la actividad solo puede ser asignada a una de estas tres consultas, y no a dos, tres o ninguna consulta.

El segundo conjunto de restricciones (3) va de la mano con la primera, modelando la asignación de las actividades a los recursos atendiendo a la tipología de recurso. Esta restricción tiene el objetivo opuesto a la primera, de modo que si la tipología de recurso en la que se debe realizar una actividad *a* no es igual a la tipología de recurso de un recurso cualquiera *j*, la actividad no sea asignada. En otras palabras, la anterior actividad de reevaluación de tipo consulta, nunca puede ser asignada para realizarse en la sala de enfermería o

en la de rayos X.

El modelo atiende a la optimización del flujo de pacientes en un instante determinado del SUH, de modo que en ese instante no todos los pacientes van a estar en la sala de espera esperando a ser atendidos por primera vez, sino que pueden estar en curso dentro de una actividad. Esto se modela en el conjunto de restricciones (4), de modo que si en el instante t=0 donde se inicia el programa, hay un paciente siendo atendido en algún recurso (dicho recurso no estará disponible), la siguiente actividad del proceso de urgencia de dicho paciente, que será la primera del mismo en el instante inicial, no podrá comenzar hasta que no termine la actividad en la que ese mismo paciente estaba siendo atendido.

El conjunto de restricciones (5) sirve para modelar el proceso de urgencia de un paciente *i*, asegurando el hecho de que para un paciente cualquiera con un proceso de urgencias predeterminada, las actividades del proceso de urgencia se irán llevando a cabo sin solapamientos. Es decir, si un paciente tiene que ir a 1ª consulta, enfermería y alta, la actividad de enfermería no comenzará hasta que no finalice la 1ª consulta, y el alta no comenzará hasta que el paciente no termine de ser atendido en el recurso de enfermería.

El conjunto de restricciones (6) y (7) se complementan, y modelan la secuenciación de actividades en un mismo recurso, de modo que en un recurso una actividad no puede comenzar en caso de que haya una actividad en curso llevándose a cabo en este mismo recurso. Esto es, si un paciente *i* está siendo atendido en la consulta 2, y hay otro paciente *i* que también tiene que ser atendido en dicha consulta, este segundo no podrá ser atendido hasta que el primero no abandone la consulta 2. Para ello se utiliza la variable gamma, de modo que si toma valor 1 (actividad a delante de *a*'), la restricción 6 no aplica, mientras que la restricción 5 si lo hará; y si toma valor 0 se producirá el caso contrario, pero las actividades en un mismo recurso no se solaparán en ninguno de los casos.

Siguiendo con los conjuntos de restricciones (8) y (9), estos conciernen a la disponibilidad de los recursos. En la forma en que se ha orientado el problema, el primero de ellos (8) modela el hecho de que partiendo de un instante determinado t=0 en el que puede haber recursos ocupados por pacientes siendo atendidos, ningún paciente pueda ser atendido en esos recursos hasta que no estén disponibles. Dando un horizonte temporal de programación alto, las restricciones (9) se convierte algo irrelevante, ya que nunca se llegará a ese instante en que el recurso vuelva a no estar disponible. No obstante, siempre se puede dar el caso de que un médico abandone su turno y su consulta tuviera que quedar inhabilitada, en ese caso, las restricciones 8 asegura que una actividad que no puede ser finalizada antes del instante de finalización de la disponibilidad de un recurso *j*, nunca sea asignada a ese recurso *j*, sino a otro cualquiera.

Sin embargo, para casos como suponer que todos los pacientes están en sala de espera en el instante inicial, o si se quiere dar disponibilidad de recursos por horarios de trabajo, las restricciones (8) modela que una actividad a sea asignada a un recurso *j* a partir de que este recurso esté disponible, y nunca antes. Mientras que la restricción número (9) asegura que una actividad nunca sea asignada a un recurso *j* si va a finalizar más tarde del instante final de disponibilidad de dicho recurso *j*.

El conjunto de restricciones (10) atiende al concepto del TEPCOF. Siendo r_tepcof el dato que indica cuanto tiempo llevaba el paciente *i* antes del instante incial t=0 esperando a ser atendido en la 1ª consulta facultativa, el valor del completion time menos el tiempo de proceso y el valor del r_tepcof de un paciente *i* es el tiempo total que dicho paciente ha tardado en ser atendido por primera vez desde su llegada al SUH. Si la resta de este valor menos el TEPCOF estandarizado según la prioridad de ese paciente es positivo, quiere decir que el paciente ha sido atendido en más tiempo del que debería según el valor establecido en el TEPCOF. En cambio, si esa resta conlleva un valor negativo, quiere decir que el paciente ha sido atendido a tiempo. La variable de decisión T es definida positiva (restricción 13), con lo que para pacientes que cumplan el TEPCOF, la variable tomará valor 0. En caso contrario, tomará un valor positivo que se trata de minimizar en la función objetivo.

El último grupo de restricciones 10, 11, 12 y 13, definen el tipo de variables de las que se compone el problema. Las variables de decisión X y γ son definidas binarias, mientras que las variables T y CT son variables definidas positivas.

4.2.3 Modelo D-básico

Se expone el mismo modelo que el del apartado inmediatamente anterior, con el método de modelado D:

$$Min. \sum_{i=1}^{I} T_i + \sum_{i=1}^{I} \sum_{j=1}^{J} CT_{last_i,j} \qquad (15)$$

$$s.a: \sum_{j \in J \mid s_a = t_j} \sum_{p \in P} X_{a,j,p} = 1 \quad \forall \quad a \in A \qquad (16)$$

$$\sum_{j \in J \mid s_a \neq t_j} \sum_{p \in P} X_{a,j,p} = 0 \quad \forall \quad a \in A \qquad (17)$$

$$CT_{first_i,j} \geq \left(r_i + pt_{first_i}\right) \cdot \sum_{p \in P} X_{first_i,j,p} \qquad \forall\, i \in I, \forall j \in J \mid s_{first_i} = t_j \qquad (18)$$

$$CT_{pu_{i,a},j} \geq CT_{pu_{i,a-1},j'} + pt_{pu_{i,a}} + M \cdot \left(\sum_{p \in P} X_{pu_{i,a},j,p} + \sum_{p \in P} X_{pu_{i,a-1},j',p} - 2 \right)$$
$$\forall\, i \in I, \forall\, a \in A, \forall\, j \in J, \forall\, j' \in J \mid pu_{i,a} \neq 0, t_j = s_{pu_{i,a}}, t_{j'} = s_{pu_{i,a-1}} \qquad (19)$$

$$\sum_{a \in A} X_{a,j,p} \leq 1 \qquad \forall\, j \in J, \forall\, p \in P \qquad (20)$$

$$CT_{a,j} \geq CT_{a',j} + pt_a - M \cdot (2 - X_{a,j,p} - \sum_{p' \in P} X_{a',j,p'})$$
$$\forall\, a \in A, \forall\, a' \in A, \forall\, j \in J, \forall\, p \in P \mid a \neq a', p > 1, t_j = s_a, t_j = s'_a \qquad (21)$$

$$CT_{a,j} \geq \left(pt_a + d_{j,1}\right) \cdot \sum_{p \in P} X_{a,j,p} \qquad \forall\, a \in A, \forall j \in J \mid s_a = t_j \qquad (22)$$

$$CT_{a,j} \leq d_{j,2} \cdot \sum_{p \in P} X_{a,j,p} \qquad \forall\, a \in A, \forall j \in J \mid s_a = t_j \qquad (23)$$

$$T_i \geq CT_{first_i,j} - pt_{first_i} - r_{tepcof_i} - tepcof_{pp_i}$$
$$\forall\, i \in I, \forall\, j \in J \mid m_{first_i} = 1, t_j = s_{first_i} \qquad (24)$$

$$X_{a,j,p} \in \{0,1\} \quad \forall\, a \in A, \forall j \in J, \forall\, p \in P \qquad (25)$$

$$CT_{a,j} \geq 0 \quad \forall\, a \in A, \forall j \in J \qquad (26)$$

$$T_i \geq 0 \quad \forall\, i \in I \qquad (27)$$

El modelo es muy parecido al anterior, así que solo se comentarán las diferencias entre ambos. En los conjuntos de restricciones número 16, 17, 18, 19, 22 y 23 lo único que cambia es la variable X, la cual ahora también depende de la posición *p* a la que la actividad a esté asignada en el recurso *j*. Se añade el sumatorio, ya que para que la restricción aplique o no, la actividad a debe estar asignada al recurso *j* en al menos una de las *p* posiciones en las que se divide dicho recurso, en caso contrario, la variable de decisión X tomará valor nulo.

En la restricción 5 se modela el hecho de que en una posición determinada *p* de un recurso *j* solo puede haber asignada como mucho una única actividad. De lo contrario, el modelo asignaría más de una actividad a una misma posición y se producirían infactibilidades y solapamientos de actividades en un recurso.

El otro conjunto de restricciones nuevo es (21), equivalente a las restricciones 6 y 7 del anterior modelo básico gamma. Al no tener ahora la variable de decisión gamma, hay que construir una restricción que asegure que las actividades no se solapen en un mismo recurso, y esto es lo que se modela en el conjunto de restricciones (21). Simplemente se trata de que una actividad *a* que va antes que la actividad *a'* en un recurso, ocupe una posición anterior a la que ocupará *a'* al ser asignada. Las posiciones no tienen por qué ser contiguas, basta con que la actividad que vaya antes tenga una posición anterior a su sucesora.

Por último, cabe destacar que la función objetivo es la misma que en el modelo anterior.

4.3 Variantes del modelo

A continuación, se presentan distintas variantes reales que se pueden producir en el SUH de cualquier hospital y que son modelables simplemente modificando los modelos base anteriormente presentados. Antes de comenzar, se añadirá de ahora en adelante para el resto de las variantes, una variante de complejidad al modelo básico. Hay pacientes críticos que a menudo precisan de la atención de más de un enfermero debido al alcance de la lesión, es decir, que sean atendidos por más de un recurso al mismo tiempo. Para modelarlo, solo hay que variar la restricción número 1 del modelo básico, que en vez de igualar el sumatorio a la unidad, se iguala al número de recursos que necesite el paciente *i* en la actividad *a:*

- ✓ Modelo G-1:

$$\sum_{j \in J \mid s_a = t_j} X_{a,j} = n_{pa_a, m_a} \quad \forall \quad a \in A \qquad (1)$$

- ✓ Modelo D-1:

$$\sum_{j \in J \mid s_a = t_j} \sum_{p \in P} X_{a,j,p} = n_{pa_a, m_a} \quad \forall \quad a \in A \qquad (1)$$

4.3.1 Asignación de prioridades a recursos

Una posibilidad es que en un SUH, según el grado de especialización del enfermero, o del médico, haya algunos de estos que estén asignados solo a atender a los pacientes más críticos, mientras que otros de menor experiencia están asignados a atender únicamente a pacientes que no presentan gravedad. A través de la asignación de prioridad a los pacientes, simplemente hay que modelar que determinados recursos solo puedan atender a según que tipos de pacientes atendiendo a la prioridad de los mismos. En este apartado se muestra esta variante al modelo básico desde ambos métodos de programación.

Para un mayor contraste visual, solo se van a escribir los cambios que se producen en el modelo básico del punto 4.2. para que el modelo responda ante estos cambios. De este modo, la función objetivo, que permanecerá igual para el resto de las variables, no se volverá a escribir, entre otras restricciones.

4.3.1.1 Asignación de prioridades a recursos: "Modelo G-2"

Se divide la restricción número 2 del modelo básico en dos restricciones:

$$\sum_{j \in J \mid s_a = t_j, pr_{pa_{a,j}}=0} X_{a,j} = 0 \quad \forall \quad a \in A \qquad (2.1)$$

$$\sum_{j \in J \mid s_a \neq t_j} X_{a,j} = 0 \quad \forall \quad a \in A \qquad (2.2)$$

La restricción 2.2 sigue siendo la misma que la restricción 2 del modelo básico, es decir, que si la tipología de recurso que necesita la actividad *a* para ser desarrollada no es la misma que la del recurso *j*, la actividad *a* no podrá ser asignada a dicho recurso. Con la restricción 2.1 además se añade, que si la tipología de recurso de la actividad *a* y la del recurso *j* coinciden, pero sin embargo el recurso *j* no puede atender a pacientes de la prioridad asignada a dicho paciente y por tanto a la actividad *a*, dicha actividad *a* tampoco podrá ser asignada al recurso *j*.

El parámetro *pr* es una matriz *5 (nº de prioridades) x nº de recursos*, de modo que si el valor es la unidad en una prioridad *i* y un recurso *j,* quiere decir que esa actividad *a* de prioridad *i* podría ser asignada en el recurso *j.*

4.3.1.2 Asignación de prioridades a recursos: "Modelo D-2"

Se realiza el mismo cambio que el propuesto en el apartado 4.3.1.1. del modelo gamma:

$$\sum_{j \in J \mid s_a = t_j, pr_{pa_{a,j}}=0} \sum_{p \in P} X_{a,j,p} = 0 \quad \forall \quad a \in A \qquad (2.1)$$

$$\sum_{j \in J \mid s_a \neq t_j} \sum_{p \in P} X_{a,j,p} = 0 \quad \forall \quad a \in A \qquad (2.2)$$

De nuevo, en la restricción 2.1 se añade que aunque la tipología de recursos coincida en actividad-recurso, si el recurso no puede atender actividades de la prioridad asignada a una actividad *a*, ésta no podrá ser asignado a dicho recurso *j*.

4.3.2 Asignación de actividades maestras a recursos

Puede darse la casuística de que en un SUH se quiera que solo determinados recursos realicen determinadas actividades. Por ejemplo, si hay 4 consultas, que solo dos de ellas realicen la 1ª consulta del paciente, y las otras dos se encarguen de realizar la reevaluación del paciente y el posterior alta del mismo. Esto se puede modelar también haciendo pequeños cambios en las restricciones planteadas. Para ello, es necesario conocer cuales son las actividades maestras de las que se compone el SUH, independientemente de que haya pacientes que realicen o no todas ellas. Conocidas las actividades, se pueden asignar a recursos de la misma forma que se ha realizado la asignación de prioridades a recursos.

4.3.2.1 Asignación de actividades maestras a recursos: "Modelo G-3"

A continuación, se muestran la variación que hay que hacer en la restricción número 1 con respecto al modelo básico con el fin de poder asignar determinadas actividades a recursos:

$$\sum_{j \in J \mid s_a = t_j, b_{m_a,j}=1} X_{a,j} = n_{pa_a,m_a} \quad \forall \quad a \in A \qquad (1)$$

Anteriormente solo se exigía que el tipo de recurso de la actividad a coincidiese con el del recurso *j*. Ahora se añade la restricción de que la actividad maestra a la que se corresponde la actividad *a* en el recurso *j* se puedan asignar, es decir, que el parámetro *b* tome valor 1. Dicho parámetro es una matriz de *nº de actividades maestras x nº de recurso*s, de modo que toma valor 1 si la actividad maestra a puede ser realizada en el recurso *j,* y 0 en caso contrario, caso en el que no aplicaría la restricción número 1.

4.3.2.2 Asignación de actividades maestras a recursos: "Modelo D-3"

Al igual que para el modelo gamma, solo hay que añadir la restricción de que el parámetro *b* tome valor unidad para que aplique en la restricción 1, de modo que dicha restricción queda de la siguiente forma:

$$\sum_{j \in J \mid s_a = t_j, b_{m_a,j}=1} \sum_{p \in P} X_{a,j,p} = n_{pa_a,m_a} \quad \forall \quad a \in A \qquad (1)$$

4.3.3 Distribución de recursos por tipología de pacientes

En un SUH se pueden dar situaciones en las que sea necesario sectorizar el SUH para atender a distintos tipos de pacientes. Sin ir más lejos, durante la pandemia del COVID, en muchos servicios de urgencias del país se crearon salas de espera a parte del resto de pacientes para aquellos que presentaran síntomas y poder así evitar contagios. También era efectivo utilizar solo determinados recursos para pacientes contagiados y otros para los no contagiados, con el fin de ahorrar tiempo en desinfectar salas y utillaje.

Esto también se puede modelar para que según el tipo de paciente, este solo pueda ser atendido en determinados recursos. Por ejemplo, si se tienen de nuevo 4 consultas, sería inteligente, ahora que la pandemia está mucho más relajada, destinar solo una de esas 4 consultas a pacientes COVID, mientras que las otras tres pueden atender a pacientes ordinarios. Así, se evitarían los famosos tiempos de set-up para tener que limpiar, y aumentaría la seguridad del paciente de contraer algún virus.

4.3.3.1 Distribución de recurso por tipología de pacientes: "Modelo G-4"

De nuevo, para el caso que se plantea, solo es necesario hacer una pequeña modificación en la restricción número 1:

$$\sum_{j \in J \mid s_a = t_j, b_{m_a,j}=1, k_{cv_a,j}=1} X_{a,j} = n_{pa_a,m_a} \quad \forall \quad a \in A \qquad (1)$$

Se tiene un parámetro parecido al parámetro *b* introducido en la variante anterior. El parámetro *k* es una matriz *c* x *nº de recursos,* de modo que para un paciente del tipo *c*, si en el recurso *j* el parámetro *k* toma valor 1, quiere decir que puede ser asignado a ese recurso, mientras que en el caso de que el parámetro *k* tome valor nulo, quiere decir que ese tipo de paciente no podrá ser atendido en el recurso correspondiente. Para que la restricción 1 aplique y la actividad a sea asignada *a* un recurso *j*, la actividad a debe pertenecer a una tipología de paciente admisible en el recurso *j*.

4.3.3.2 Distribución por tipología de pacientes: "Modelo D-4"

Nuevamente, hay que añadir exactamente lo mismo que lo que se ha añadido en el apartado 4.3.3.1 del modelo gamma para esta nueva variante, de modo que la restricción 1 aplique solamente cuando el parámetro *k* tome

valor unitario:

$$\sum_{j \in J \mid s_a = t_j, b_{m_a,j}=1, k_{cv_a,j}=1} \sum_{p \in P} X_{a,j,p} = n_{pa_a,m_a} \quad \forall \quad a \in A \qquad (1)$$

4.4 Modelos definitivos

En este apartado, a modo de resumen, se vuelven a presentar los modelos, pero ya con todas las variantes activas al mismo tiempo. De este modo, para que una actividad *a* sea asignada a un recurso *j*, ambos tienen que tener la misma tipología de recurso asignado, el recurso *j* tiene que poder atender a pacientes de prioridad *k*, el recurso *j* tiene que poder realizar la actividad maestra "*o*" a la que se corresponda la actividad *a*, y además el recurso *j* tiene que poder recibir a pacientes de una determinada tipología. En caso contrario, la actividad *a* no podrá ser asignada al recurso *j*.

4.4.1 Modelo G

Se muestra el modelo de programación lineal que aúna todas las variantes en un único modelo, atendiendo al método de programación que incluye la variable de decisión gamma:

$$Min. \sum_{i=1}^{I} T_i + \sum_{i=1}^{I} \sum_{j=1}^{J} CT_{last_i,j}$$

$$s.a: \sum_{j \in J \mid s_a = t_j, b_{m_a,j}=1, k_{cv_a,j}=1} X_{a,j} = n_{pa_a,m_a} \quad \forall \quad a \in A \qquad (1)$$

$$\sum_{j \in J \mid s_a = t_j, pr_{pa_{a,j}}=0} X_{a,j} = 0 \quad \forall \quad a \in A \qquad (2.1)$$

$$\sum_{j \in J \mid s_a \neq t_j} X_{a,j} = 0 \quad \forall \quad a \in A \qquad (2.2)$$

$$CT_{first_i,j} \geq \left(r_i + pt_{first_i}\right) \cdot X_{first_i,j} \quad \forall\, i \in I, \forall j \in J \mid s_{first_i} = t_j \qquad (3)$$

$$CT_{pu_{i,a},j} \geq CT_{pu_{i,a-1},j'} + pt_{pu_{i,a}} + M \cdot \left(X_{pu_{i,a},j} + X_{pu_{i,a-1},j'} - 2\right)$$
$$\forall\, i \in I, \forall\, a \in A, \forall\, j \in J, \forall\, j' \in J \mid pu_{i,a} \neq 0, t_j = s_{pu_{i,a}}, t_{j'} = s_{pu_{i,a-1}} \qquad (4)$$

$$CT_{a',j} + M \cdot \left(2 - X_{a,j} - X_{a',j}\right) \geq CT_{a,j} + pt_{a'} - M \cdot \left(1 - \gamma_{a,a',j}\right)$$
$$\forall a \in A, \forall\, a' \in A, \forall j \in J \mid a < a', t_j = s_a, t_j = s_{a'} \qquad (5)$$

$$CT_{a,j} + M \cdot \left(2 - X_{a,j} - X_{a',j}\right) \geq CT_{a',j} + pt_a - M \cdot \gamma_{a,a',j}$$

$$\forall a \in A, \forall\, a' \in A, \forall j \in J \mid a < a', t_j = s_a, t_j = s_{a'} \qquad (6)$$

$$CT_{a,j} \geq \left(pt_a + d_{j,1}\right) \cdot X_{a,j} \qquad \forall\, a \in A, \forall j \in J \mid s_a = t_j \qquad (7)$$

$$CT_{a,j} \leq d_{j,2} \cdot X_{a,j} \qquad \forall\, a \in A, \forall j \in J \mid s_a = t_j \qquad (8)$$

$$T_i \geq CT_{first_i,j} - pt_{first_i} - r_{tepcof_i} - tepcof_{pp_i}$$
$$\forall\, i \in I, \forall\, j \in J \mid m_{first_i} = 1, t_j = s_{first_i} \qquad (9)$$

$$X_{a,j} \in \{0,1\} \qquad \forall\, a \in A, \forall j \in J \qquad (10)$$

$$CT_{a,j} \geq 0 \qquad \forall\, a \in A, \forall j \in J \qquad (11)$$

$$\gamma_{a,a',j} \in \{0,1\} \qquad \forall\, a \in A, \forall\, a' \in A, \forall j \in J \qquad (12)$$

$$T_i \geq 0 \qquad \forall\, i \in I \qquad (13)$$

4.4.2 Modelo D

Por último, se expone el modelo de programación lineal que aúna todas las variantes en un único modelo, atendiendo al método de programación que divide los recursos en P posiciones:

$$Min. \sum_{i=1}^{I} T_i + \sum_{i=1}^{I} \sum_{j=1}^{J} CT_{last_i,j}$$

$$s.a: \sum_{j \in J \mid s_a = t_j, b_{m_a,j}=1, k_{cv_a,j}=1} \sum_{p \in P} X_{a,j,p} = n_{pa_a,m_a} \quad \forall \quad a \in A \qquad (1)$$

$$\sum_{j \in J \mid s_a = t_j, pr_{pa_{a,j}}=0} \sum_{p \in P} X_{a,j,p} = 0 \quad \forall \quad a \in A \qquad (2.1)$$

$$\sum_{j \in J \mid s_a \neq t_j} \sum_{p \in P} X_{a,j,p} = 0 \quad \forall \quad a \in A \qquad (2.2)$$

$$CT_{first_i,j} \geq \left(r_i + pt_{first_i}\right) \cdot \sum_{p \in P} X_{first_i,j,p} \qquad \forall\, i \in I, \forall j \in J \mid s_{first_i} = t_j \qquad (3)$$

$$CT_{pu_{i,a},j} \geq CT_{pu_{i,a-1},j'} + pt_{pu_{i,a}} + M \cdot \left(\sum_{p \in P} X_{pu_{i,a},j,p} + \sum_{p \in P} X_{pu_{i,a-1},j',p} - 2 \right)$$
$$\forall\, i \in I, \forall\, a \in A, \forall\, j \in J, \forall\, j' \in J \mid pu_{i,a} \neq 0, t_j = s_{pu_{i,a}}, t_{j'} = s_{pu_{i,a-1}} \quad (4)$$

$$\sum_{a \in A} X_{a,j,p} \leq 1 \quad \forall\, j \in J, \forall\, p \in P \quad (5)$$

$$CT_{a,j} \geq CT_{a',j} + pt_a - M \cdot (2 - X_{a,j,p} - \sum_{p' \in P} X_{a',j,p'})$$
$$\forall\, a \in A, \forall\, a' \in A, \forall\, j \in J, \forall\, p \in P \mid a \neq a', p > 1\ t_j = s_a, t_j = s'_a \quad (6)$$

$$CT_{a,j} \geq (pt_a + d_{j,1}) \cdot \sum_{p \in P} X_{a,j,p} \quad \forall\, a \in A, \forall j \in J \mid s_a = t_j \quad (7)$$

$$CT_{a,j} \leq d_{j,2} \cdot \sum_{p \in P} X_{a,j,p} \quad \forall\, a \in A, \forall j \in J \mid s_a = t_j \quad (8)$$

$$T_i \geq CT_{first_i,j} - pt_{first_i} - r_{tepcof_i} - tepcof_{pp_i}$$
$$\forall\, i \in I, \forall\, j \in J \mid m_{first_i} = 1, t_j = s_{first_i} \quad (9)$$

$$X_{a,j,p} \in \{0,1\} \quad \forall\, a \in A, \forall j \in J, \forall\, p \in P \quad (10)$$

$$CT_{a,j} \geq 0 \quad \forall\, a \in A, \forall j \in J \quad (11)$$

$$T_i \geq 0 \quad \forall\, i \in I \quad (12)$$

5 EXPERIMENTACIÓN

En este apartado, se muestran los resultados de los diferentes modelos de decisión propuestos en el TFG. Para ello, se considera el problema de gestión del SUH de un hospital clásico americano, cuyas características se muestran en el artículo [6]. El hospital donde se desarrolla dicho artículo cuenta con una serie de recursos que permanecerán fijos durante toda la experimentación, siendo variables otros parámetros como el número de pacientes, y los procesos de urgencia de dichos pacientes.

En primer lugar, se citan las asunciones del marco en que se desarrolla la experimentación [6]:

- ✓ Ningún paciente abandona el SUH antes de recibir el alta. Es decir, todos los pacientes realizan su proceso de urgencias de principio a fin.
- ✓ Todos los equipos de procesamiento de pruebas y datos permanecen disponible y operativo las 24 horas del día y los 7 días de la semana.
- ✓ El índice de desempeño de todo el personal médico permanece constante durante todo el horizonte temporal de resolución del problema, así como la disponibilidad de dicho personal.
- ✓ El tiempo que tarda el personal del hospital en trasladarse de su ubicación al recurso donde tiene que atender a un paciente es muy pequeño y se considera despreciable.
- ✓ Todos los pacientes permanecen en el mismo grado de prioridad que el asignado en triaje durante todo su proceso de urgencias.

Durante la experimentación, el objetivo es ver como responden los modelos ante distintos niveles de saturación del SUH. Para ello, se generarán 10 instancias para tres niveles disitntos de saturación, de modo que para el primero de los niveles el SUH esté muy poco saturado (25%); en el segundo se dará un nivel intermedio de saturación (50%); y por último un tercer nivel que implica una alta saturación del SUH (75%).

5.1 Datos de la experimentación

La generación de instancias mencionadas es aleatoria, atendiendo únicamente a los distintos grados de saturación. Es decir, los pacientes y sus respectivos procesos de urgencias se generan aleatoriamente (datos dinámicos). No obstante, también hay datos estáticos que no cambiarán y que son independientes del nivel de saturación que se tenga en el SUH, como lo son aquellos datos que conciernen a los recursos y la tipología de los mismos.

5.1.1 Datos estáticos

En [6] se presenta un calendario diario que muestra los recursos que se tienen disponibles por rangos horarios en lo que a médicos, médicos asistentes y enfermeros concierne. Como solo se expone el horario de dichos recursos, hacemos tres nuevas asunciones para abordar el problema:

- ✓ Se tiene un laboratorio para las pruebas de extracción de sangre siempre disponible.
- ✓ Se tiene una sala de radiodiagnóstico (rayos X) siempre disponible.
- ✓ Se tiene una sala para pruebas TAC (CT-Scan, Computed-Tomography Scan) siempre disponible.
- ✓ Para poder trabajar con la variante de tipología de pacientes, se va a incrementar el número de dichos tres recursos a dos unidades para poder diferenciar un tipo de pacientes de otro. No obstante, también se podría abordar la variante aumentando los tiempos de procesos de las actividades para considerar hipotéticos tiempos de set-up en desinfección de recursos y limpieza.

El horario que se propone en el artículo es el siguiente:

Shift	Schedule	Type and number of resources		
		Nurses	PAs	Physicians
1	7 a.m.–7 p.m.	6	0	1
2	9 a.m.–5 p.m.	0	1	1
3	9 a.m.–9 p.m.	1	0	0
4	11 a.m.–11 p.m.	1	1	0
5	5 p.m.–1 a.m.	0	1	1
6	7 p.m.–7 a.m.	6	0	1
7	12 a.m.–12 p.m.	1	0	0
8	3 p.m.–3 a.m.	1	0	0

PAs: physician assistants.

Figura 5-1. Calendario de médicos, médicos asistentes y enfermeros [6]

En la experimentación, al ejecutar el modelo nos situamos en un instante t=0 en el que ya existe un número de pacientes en el SUH, de forma que algunos pueden estar en mitad de su proceso de urgencia, otros ni haberlo empezado, y otros estar terminándolo. Para obtener un mayor número de soluciones en distintas situaciones y rangos, en la experimentación nos vamos a situar en dos horarios de los planteados en [6] , las 6:00 a.m. y las 12:00 p.m. Cabe destacar que durante dichos horarios los recursos disponibles son constantes, de modo que los recursos que se tienen por tipología en ambos horarios son los siguientes (unificando médicos y médicos asistentes):

6:00 a.m.	**Consulta**	**Enfermería**	**Laboratorio**	**Sala rayos**	**CT-Scan**
Número de recursos	1	6	1	1	1

Tabla 5-1. Número de recursos por tipología a las 6:00 a.m. [6]

12:00 a.m.	**Consulta**	**Enfermería**	**Laboratorio**	**Sala rayos**	**CT-Scan**
Número de recursos	4	8	1	1	1

Tabla 5-2. Número de recursos por tipología a las 12:00 a.m. [6]

Para abordar la cuarta variante en la que se consideran distintos tipos de pacientes, el desglose de recursos queda de la siguiente forma:

12:00 a.m.	**Consulta**	**Enfermería**	**Laboratorio**	**Sala rayos**	**CT-Scan**
Número de recursos	4	8	2	2	2

Tabla 5-3. Número de recursos por tipología a las 12:00 a.m. para la variante de tipología de pacientes [6]

Todos ellos hacen un total de 10, 15 y 18 recursos disponibles respectivamente en el horizonte temporal en que se desarrolla la experimentación. Cabe destacar que, en el instante inicial, puede ser que un recurso esté ocupado atendiendo a un paciente, por lo que no estaría disponible dicho instante inicial, sino una vez haya terminado de atender al paciente en curso.

Asimismo, las actividades maestras presentes en la experimentación, así como el tipo de recurso que dicha actividad maestra necesita para ser desarrollada, se muestran en la siguiente tabla:

Actividad maestra	Tipo de recurso
1ª Consulta	Consulta
1ª Enfermería	Enfermería
Extracción de sangre	Laboratorio
Rayos X	Sala rayos
TAC	CT-Scan
Reevaluación	Consulta
2ª Enfermería	Enfermería
Alta	Consulta

Tabla 5-4. Relación actividades maestras y tipología de recursos. [6]

5.1.2 Datos dinámicos

Como se ha explicado anteriormente, se generarán instancias con datos de pacientes aleatorios según el nivel de saturación del SUH. De [6] se pueden extraer datos que hacen referencia al cálculo del indicador Length of Stay real de cada paciente atendiendo a la prioridad del mismo:

	ESI level				
	1	2	3	4	5
Simulation average LOS (hrs.)	2.53	3.15	2.81	1.54	1.30
Real average LOS (hrs.)	3.15[a]	3.14	2.86	1.49	1.31
Real maximum LOS (hrs.)	3.15[a]	13.95	26.87	9.40	4.00
Simulation average time to be seen by a P/PA (hrs.)	0.05	1.10	1.21	1.00	1.23
Real average time to be seen by a P/PA (hrs.)	0.03[a]	1.14	1.27	1.03	1.25

[a]Corresponding to one patient during the period of time analyzed.
ESI: Emergency Severity Index; LOS: length-of-stay; P: physician; PA: physician assistant.

Figura 5-2. Resultados del estudio realizado en el artículo [6]

Lo que se busca es llegar a un nivel de saturación determinado dentro del periodo de tipo LOS_{medio}, es por ello tomando los datos de la segunda fila, se puede obtener un tiempo medio del parámetro LOS:

$$LOS_{medio} = \frac{3.15 + 3.14 + 2.86 + 1.49 + 1.31}{5} = 2.39\ horas = 143.4\ minutos$$

Con este dato, la generación de pacientes aleatoria se detendrá cuando en cualquiera de los tipos de recursos de los que se dispone, se alcance el nivel de saturación establecido en el periodo de tipo LOS_{medio}. Por ejemplo, si se estima un nivel de saturación del 75%, cuando un tipo de recurso acumule $0.75 \cdot 143.4 = 107.55$ minutos, la generación de pacientes se detendrá, y se obtiene una primera instancia.

En [6], también se realiza un estudio de en qué porcentaje acude un paciente de una determinada prioridad a urgencias, dato con el que también se produce la asignación aleatoria con dicha probabilidad de prioridad a un paciente:

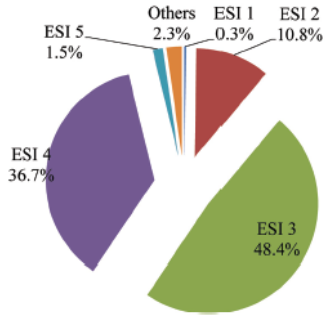

Figura 5-3. Porcentajes de asistencia de pacientes al SUH según prioridad [6]

Para los procesos de urgencia de cada paciente, en [6] se propone una utilización de recursos para cada prioridad de paciente, las cuales se muestran en la siguiente ilustración:

Patient ESI level	Resources				Required tests		
	Physician (P)	Physician assistant (PA)	Nurse	EMT	Blood	X-Ray	CT-Scan
1	1		2	1	√	√	√ 70%
2	1		2	1	√	√	√ 70%
3	1		1		√	√	√ 30%
4		1	1		X	√	X
5		1	1		X	X	X

Note. Patients rated as ESI levels 1 and 2 are considered high risk. Patients rated as ESI levels 3–5 are considered non-high risk.
EMT: Emergency medical technician.

Figura 5-4. Utilización de recursos en el proceso de urgencia por tipología de paciente [6]

Como se puede observar, los pacientes de prioridad 1 y 2, necesitan ser atendidos por dos enfermeros, lo que se modela en la variante de complejidad mencionada anteriormente. Asimismo, no se va a distinguir entre consulta de médico y consulta de médicos asistentes, sino que en a través de la variante de asignación de prioridades a recursos, se destinarán algunas consultas para los pacientes de prioridad 1,2 y 3 (médicos), y otras para los pacientes de prioridad 4 y 5 (médicos asistentes). Además, no siempre está asegurada la presencia del CT-Scan para ninguno de los pacientes, solo se da en los procesos de urgencia de pacientes de prioridad 1, 2 y 3 en según que porcentaje.

Otro factor a tener en cuenta es que en la experimentación planteada se excluye el servicio EMT (Equipos de Emergencia Médica), dado que no se asocian a ninguna tipología de recurso concreto, sino que son equipos de médicos que actúan en situaciones de extrema urgencia, como recibir a pacientes que llegan críticos en ambulancia (que quizás ni entren en el SUH y vayan directamente a cuidados intensivos) o proporcionan servicio clínico directo a poblaciones afectadas por emergencias y desastres. [16] Los procesos de urgencia que se plantean se simplifican al excluir los casos en los que hace falta este servicio de urgencia, teniendo en cuenta que los pacientes más críticos deben recibir atención inmediata (TEPCOF nulo).

Por otro lado, la experimentación también está apoyada en los datos que el artículo [6] proporciona del estudio de los tiempos de proceso de todas las actividades del proceso de urgencia de todos los pacientes según a prioridad. Estos tiempos presentan variabilidad, con lo que el artículo los plantea como una distribución triangular, y para la generación de instancias se utilizará un tiempo rándom dentro de dicha distribución. La tabla con los tiempos de proceso es la siguiente:

Process	Type of patient				
	ESI 1	ESI 2	ESI 3	ESI 4	ESI 5
Sign-in	Triangular (3, 5, 10)	Triangular (3, 5, 10)	Triangular (3, 5, 10)	Triangular (3, 5, 10)	Triangular (3, 5, 10)
Triage	Triangular (0.5, 1, 1.5)	Triangular (2, 3, 5)	Triangular (5, 7, 10)	Triangular (5, 7, 10)	Triangular (5, 7, 10)
Registration	Triangular (3, 8, 12)	Triangular (3, 8, 12)	Triangular (3, 8, 12)	Triangular (3, 8, 12)	Triangular (3, 8, 12)
First visit (nurse)	Triangular (20, 30, 75)	Triangular (18, 28, 45)	Triangular (15, 22, 30)	Triangular (5, 10, 20)	Triangular (2, 4, 8)
First visit (P/PA)	Triangular (10, 20, 25)	Triangular (10, 20, 25)	Triangular (5, 15, 20)	Triangular (2, 3, 5)	Triangular (2, 3, 5)
Second visit (nurse)	Triangular (20, 30, 40)	Triangular (15, 20, 30)	Triangular (10, 15, 20)	Triangular (5, 8, 12)	-
Second visit (P/PA)	Triangular (20, 30, 40)	Triangular (15, 20, 25)	Triangular (8, 15, 18)	Triangular (5, 10, 15)	-
X-Ray	Triangular (3, 5, 15)	Triangular (5, 10, 20)	Triangular (5, 10, 20)	Triangular (2, 5, 10)	-
Blood	Triangular (3, 5, 15)	Triangular (3, 5, 15)	Triangular (3, 5, 15)	-	-
CT-Scan	Constant 20	Constant 15	Constant 15	-	-

P: physician; PA: physician assistant.

Figura 5-5. Tiempos de proceso de las actividades del proceso de urgencia por prioridades [6]

Para el problema planteado, las tres primeras filas (registro, triaje y admisión) son redundantes, y los datos que sirven para el estudio empiezan en la 1ª Consulta. Así pues, se resume el proceso de urgencia de cada paciente por prioridades en la siguiente tabla:

Número de actividad	**Prioridad 1**	**Prioridad 2**	**Prioridad 3**	**Prioridad 4**	**Prioridad 5**
1	1ª Consulta	1ª Consulta	1ª Consulta	1ª Consulta	1ª Consulta
2	1ª Enfermería	1ª Enfermería	1ª Enfermería	1ª Enfermería	1ª Enfermería
3	Extracción	Extracción	Extracción	Rayos X	Alta
4	Rayos X	Rayos X	Rayos X	Reevaluación	
5	CT-Scan (70%)	CT-Scan (70%)	CT-Scan (30%)	2ª Enfermería	
6	Reevaluación	Reevaluación	Reevaluación	Alta	
7	2ª Enfermería	2ª Enfermería	2ª Enfermería		
8	Alta	Alta	Alta		

Tabla 5-5. Proceso de urgencia por prioridades de pacientes [6]

Dichos procesos se generarán de forma aleatoria, de la mano con la prioridad asignada al paciente. Además, como se ha comentado anteriormente, en caso de que el paciente ya esté en mitad de su proceso de urgencia, puede ser que la primera actividad de dicho paciente sea cualquier otra distinta a la primera consulta, lo que también será un dato dinámico. Para modelar esta casuística, se utiliza el parámetro de la disponibilidad de los recursos, de forma que en caso de que haya algún paciente que ya haya iniciado su proceso de urgencia, se asignará un instante inicial de disponibilidad al recurso en que el paciente se encuentre distinto de 0, que será un valor rándom elegido de la distribución triangular del tiempo de proceso asignado a la actividad que esté en curso, y que equivaldrá al tiempo que le quede al paciente para terminar de ser atendido en esa actividad, y poder pasar a la siguiente de su proceso de urgencia.

Por último, destacar que para estos pacientes no será posible saber si se ha cumplido el TEPCOF establecido o no, ya que para el instante de inicio de la experimentación, estos pacientes ya han sido atendidos en 1ª consulta, por lo que no se conoce el dato del inicio de esta actividad.

5.2 Resultados de la experimentación

Para la experimentación, se han considerado tres casos distintos comentados anteriormente:

- Basado en el número de recursos propuesto en horario de 6 de la mañana en el artículo [6], y suponiendo un laboratorio, una sala de radiodiagnóstico y una sala TAC.
- Basado en el número de recursos propuesto en horario de 12 del mediodía en el artículo [6], y suponiendo un laboratorio, una sala de radiodiagnóstico y una sala TAC.
- Basado en el número de recursos propuesto en horario de 12 del mediodía en el artículo [6], y suponiendo dos laboratorios, dos salas de radiodiagnóstico y dos salas TAC.

Para cada una de estas distribuciones de recursos, se van a considerar tres niveles de saturación distintos del SUH:

- Saturación del 25% en alguna de las tipologías de recurso.
- Saturación del 50% en alguna de las tipologías de recurso.
- Saturación del 75% en alguna de las tipologías de recurso.

Para cada configuración de recursos y nivel de saturación se generarán un total de diez instancias que contienen los datos de los pacientes con sus respectivos procesos de urgencia en un instante t=0, a partir de las cuales se podrán sacar conclusiones de como responden los modelos. Durante la exposición de los resultados obtenidos durante la exposición, se denominarán a los distintos modelos propuestos siguiendo la siguiente nomenclatura:

Nomenclatura	Descripción
$V1\_G$	Modelo básico con método de programación que utiliza la variable γ
$V2\_G$	Variante del modelo de asignación de prioridades a recursos con método de programación que utiliza la variable γ
$V3\_G$	Variante del modelo de asignación de actividades maestras a recursos con método de programación que utiliza la variable γ
$V4\_G$	Variante del modelo de tipología de pacientes con método de programación que utiliza la variable γ
$V1\_P$	Modelo básico con método de programación que divide la disponibilidad de los recursos en posiciones
$V2\_P$	Variante del modelo de asignación de prioridades a recursos con método de programación que divide la disponibilidad de los recursos en posiciones
$V3\_P$	Variante del modelo de asignación de actividades maestras a recursos con método de programación que divide la disponibilidad de los recursos en posiciones
$V4\_P$	Variante del modelo de tipología de pacientes con método de programación que divide la disponibilidad de los recursos en posiciones

Tabla 5-6 Nomenclatura de las variantes en la experimentación

Otro punto reseñable es el hecho de que se ha establecido un tiempo límite de resolución del problema de 10 minutos (600 segundos), de modo que para cada instancia, cada modelo cuenta con dicho tiempo para encontrar la mejor solución posible. Si el modelo llegase a una solución óptima, el tiempo de ejecución será menor o igual al tiempo límite, mientras que si la solución encontrada es factible o ni siquiera encuentra solución, el tiempo de ejecución será dicho tiempo límite, instante en el que el modelo dejará de buscar mejorar la mejor solución que ha encontrado hasta ese momento.

Dicho tiempo límite se ha establecido en diez minutos ya que se busca que los modelos den una respuesta dentro del margen que se tiene para atender al paciente más desfavorable en primera consulta, que es el paciente de prioridad el cual tiene un TEPCOF de 15 minutos (se asume que los pacientes de prioridad 1 son atendidos directamente nada más llegar al SUH). Se dejan 5 minutos de margen para que cuando el modelo emita la solución, está pueda ser llevada a cabo en el hospital antes de llegar al tiempo límite.

5.2.1 Caso de estudio 1: "6:00 a.m."

Dentro de este primer caso de estudio, solo serán valorados el modelo básico, y las variantes de asignación de prioridades a recursos y de asignación de actividades maestras a recursos. La cuarta variante será evaluada más adelante como se ha explicado anteriormente. Es claro que para los tres modelos se evaluarán todas las instancias utilizando tanto el método de programación que implica el uso de la variante γ como el método que utiliza la división de la disponibilidad de un recurso en posiciones. A continuación, se vuelve a mostrar la configuración de recursos que se tiene para el horario establecido:

6:00 a.m.	Consulta	Enfermería	Laboratorio	Sala rayos	CT-Scan
Número de recursos	1	6	1	1	1

Tabla 5-7. Número de recursos por tipología del caso de estudio 1 [6]

5.2.1.1 Nivel de saturación del 25%

Para un nivel de saturación bastante bajo del SUH y bajo la configuración de recursos indicada, se han generado diez instancias aleatorias, obteniéndose un promedio de los siguientes pacientes y actividades de los mismos:

| I |	| A |	| J |
2,3	11,6	10

Tabla 5-8. Pacientes, actividades y recursos del caso de estudio 1 con saturación del 25%

Son números bastante bajos dado que se tienen pocos recursos y el nivel de saturación del SUH también es bajo, con lo que apenas hay 2 o 3 pacientes dentro del SUH en este primer caso. En primer lugar, se muestra la Figura 5-6 en que se representa la tipología de soluciones a las que se ha llegado en cada instancia para cada modelo de programación lineal:

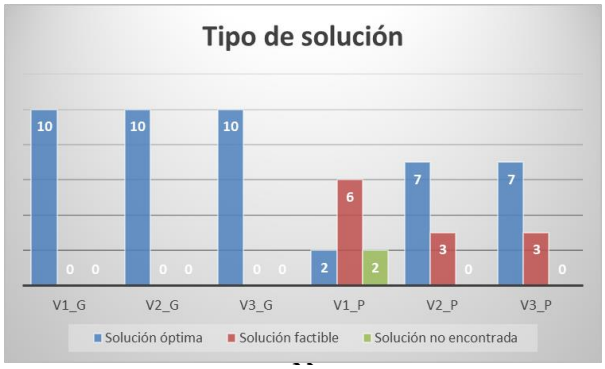

Figura 5-6. Tipología de soluciones para el caso de estudio 1 al 25% de saturación

Todos los modelos responden bastante bien para problemas con dicha saturación y configuración de recursos. En el caso del primer método de programación, los tres modelos (básico y variantes 1 y 2) consiguen llegar a la solución óptima en el 100% de los casos propuestos. Sin embargo, para el segundo de los métodos, no siempre se llega a la solución óptima. Aunque para las variantes 1 y 2 hay un porcentaje alto de optimalidad (70%), en el modelo básico tan solo se llega a la solución óptima en un 20% de los casos propuestos, y hay otro 20% en el que ni siquiera se llega a encontrar una solución factible en el tiempo límite de ejecución establecido de 10 minutos. En la Figura 5-7 se muestra el valor promedio que cada modelo ha obtenido de la función objetivo para todas las instancias propuestas.

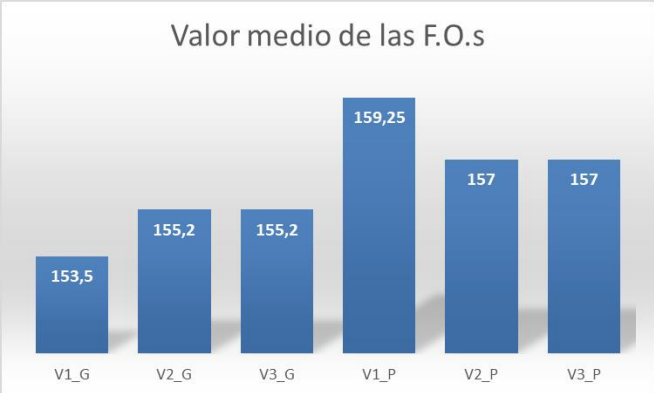

Figura 5-7. Valor medio de la F.O. para el caso de estudio 1 al 25% de satuación

En promedio, el primero de los métodos de programación obtiene mejor solución que el segundo para cada modelo, aunque también es importante conocer en cuanto tiempo ha encontrado cada modelo la solución para poder comparar ambos métodos:

Figura 5-8. Tiempo medio de ejecución de los modelos para el caso de estudio 1 al 25% de saturación

No solo el método de modelado G obtiene mejor valor de la función objetivo, sino que lo hace en bastante menos tiempo. Esto se produce también debido a que para el método de la división en posiciones de la disponibilidad de los recursos hay veces en que, o bien no encuentra solución, o la solución encontrada no es óptima, con lo que el tiempo de ejecución se dispara hasta el valor máximo establecido para dicho tiempo (600 segundos).

Un aspecto a destacar que se dará en el resto de casos de estudio es el hecho de que, al añadir la variante de asignación de actividades maestras a recursos, al no estar cambiando nada del problema, las soluciones obtenidas son las mismas. Caso distinto hubiera sido que por ejemplo se estableciese que las altas solo se pueden dar en una consulta concreta, entre otras opciones. Ahí el problema sería distinto y por tanto el valor de la F.O. si cambiaría.

No obstante, la comparación de los valores de la función objetivo no es del todo pulcra para el modelo básico, ya que en el método de las posiciones hay dos instancias en las que no se encuentra solución. Si se eliminan estas dos instancias y se vuelve a calcular el valor promedio de la función objetivo, se obtiene la siguiente diferencia:

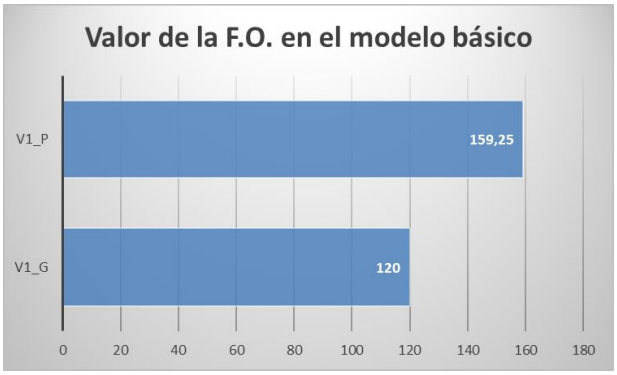

Figura 5-9. Comparativa del valor de la F.O en el modelo básico del caso de estudio 1 con saturación del 25%

La diferencia es aún más significativa si cabe. En todos los casos planteados se irá comparando como responden ambos métodos de programación ante los problemas planteados, pero en este primer caso no cabe duda de que el primero de los métodos obtiene una solución mejor, con mayor porcentaje de optimalidad y en mucho menos tiempo.

Otro aspecto que se obtiene de la resolución de las instancias es el GAP, parámetro que mide cuánto de cerca se ha quedado la solución factible encontrada de la solución óptima. En este primer caso, ante el alto porcentaje de optimalidad obtenido, analizar los valores de dicho parámetro no va a aportar demasiada información, aunque quedan recogidos en la siguiente figura:

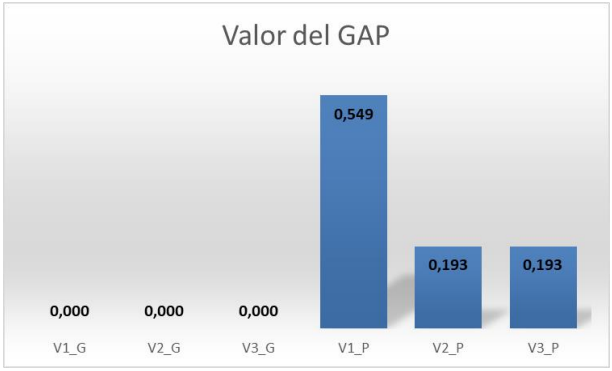

Figura 5-10. Valor promedio del GAP para el caso de estudio 1 con saturación del 25%

El parámetro GAP toma valor nulo en el óptimo, y valor infinito en caso de no encontrar solución. Así pues, cuanto mayor sea el valor de dicho parámetro, más lejos estará la solución factible encontrada de la solución óptima deseada. Es por ello que para el método de modelado G, en todas sus variantes el GAP toma valor cero, ya que el porcentaje de optimalidad es del 100%. Sin embargo, para calcular el GAP en las tres variantes empleando el otro método de modelado (D), solo se han tenido en cuenta aquellas instancias en las que no se ha alcanzado el valor óptimo, obteniéndose el valor promedio que se muestra. Para el modelo básico, la solución encontrada queda bastante lejos del óptimo, pero para las variantes 1 y 2 dicho valor no queda tan lejos del óptimo.

Un aspecto muy importante a tener en cuenta es el hecho de que la solución factible de un modelo para alguna instancia podría ser perfectamente igual al óptimo del problema, solo que en los 600 segundos que se le da al modelo para resolver el problema, no se ha podido demostrar y afirmar que dicho valor sea el óptimo, aunque se esté en el. Esto sucede numerosas veces en la resolución de los problemas, donde se obtiene el mismo valor de la función objetivo para una misma variante utilizando ambos métodos de programación y sin embargo en uno de ellos dicha solución es óptima, mientras que en el otro la solución es factible, es decir, que no se ha podido demostrar que sea óptima.

Por último, para profundizar en la reflexión de cual de los dos métodos de modelado es mejor que el otro, es interesante contemplar el número de restricciones y de variables que ambos métodos suponen en la resolución del problema en promedio para los tres modelos propuestos:

	V1_G	V2_G	V3_G	V1_P	V2_P	V3_P
Número de variables	880	880	880	773	773	773
Número de restricciones	218	230	230	614	626	626

Tabla 5-9. Número de variables y restricciones en el caso de estudio 1 con saturación del 25%

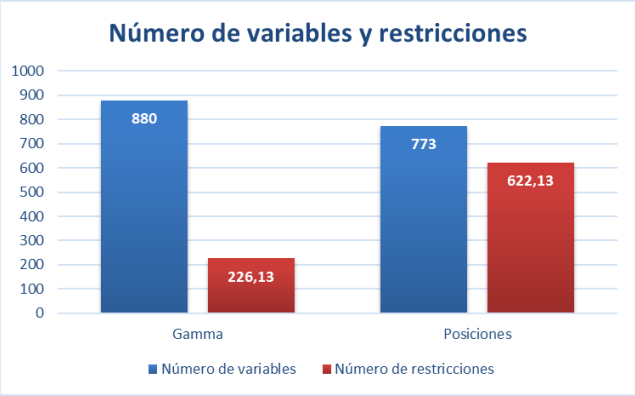

Figura 5-11. Comparativa del número de variables y restricciones del caso de estudio 1 con saturación del 25%

Aunque para el primer método de modelado (G) el número de variables que componen el modelo es ligeramente mayor, la gran diferencia que existe en el número de restricciones que hay en cada modelo hace que, en términos generales, el método de programación que emplea la variable γ sea mejor aún teniendo más variables que resolver. Esto también se ve reflejado en que para el segundo método D el porcentaje de optimalidad es mucho menor que en el primero de ellos, es decir, que para el mismo tiempo máximo de ejecución, el primer método encuentra una solución mejor y en menos tiempo.

No obstante, y sirva esta aclaración para el resto de casos planteados a continuación, en el modelo que divide la disponibilidad de los recursos en *p* posiciones (discretización del dominio de las variables de decisión), el valor de dicho parámetro *p* se puede regular y ajustar de forma que cuanto menor sea el valor de *p*, menos variables se obtendrán y por tanto menos restricciones. Es decir, que si se encuentra el valor óptimo del parámetro *p* para cada problema, la probabilidad de llegar a optimalidad aumenta mientras que el tiempo de ejecución y el GAP disminuyen. Sin embargo, este aspecto no es objeto de estudio de este TFG. Encontrar dicho valor óptimo de *p* es un problema complejo ya que es imposible saber cuántas actividades irán a cada recurso en la solución óptima, ni cual será el número máximo de actividades que abarcará un recurso, con lo que habría que ir probando para cada problema.

Así pues, lo que si es objeto de estudio de este proyecto es demostrar cual de ambos métodos de modelado es más eficiente, y aunque para este primer caso de estudio la respuesta parece evidente, antes de sacar conclusiones precipitadas, ambos van a ser valorados en otros niveles de saturación y configuraciones de recursos de cara a poder afirmar cual de los dos métodos es mejor.

5.2.1.2 Nivel de saturación del 50%

Sin variar la configuración de recursos, se aumenta el doble la saturación del SUH, lo que tiene la consecuencia directa de aumentar el número de pacientes, y con ello de actividades, debido al simple hecho de que los recursos pueden abarcar más tareas antes de llegar al nivel de saturación límite establecido. De este modo, el número de pacientes, actividades y recursos promedio de las diez instancias generadas aleatoriamente queda de la siguiente forma:

|I|	|A|	|J|
4,7	20,1	10

Tabla 5-10. Pacientes, actividades y recursos del caso de estudio 1 con saturación del 50%

Duplicar el nivel de saturación ha supuesto duplicar también el número de pacientes que pueden ser atendidos en el SUH. En lo que a la tipología de soluciones encontradas se refiere, se recogen los datos en la Figura 5-12.

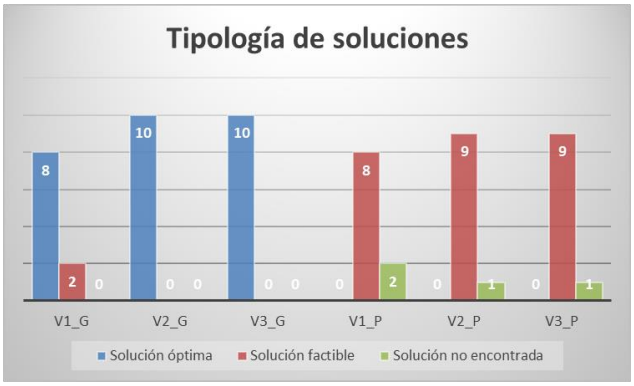

Figura 5-.12 Tipología de soluciones para el caso de estudio 1 al 50% de saturación

Los resultados han empeorado. Los tres modelos realizados con el método de modelado G siguen obteniendo resultados óptimos casi en un 93% de los casos, mientras que el otro método no ha conseguido llegar a la solución (o demostrar la optimalidad de la solución encontrada) en ninguno de los casos propuestos. No obstante, respecto a la factibilidad de las soluciones encontradas, apenas hay cuatro casos en los que algún modelo no ha sido capaz de encontrar solución. Es decir que, en diez minutos, de todas las soluciones encontradas, un 93% de todas ellas son al menos factibles. Cabe reseñar que ese 7% pertenece a casos de modelos en los que se ha empleado el método de programación que divide la disponibilidad de los recursos en p posiciones.

En la siguiente figura (5-13), se muestra el valor promedio de la función objetivo que cada modelo ha proporcionado:

Figura 5-13. Valor medio de la F.O. para el caso de estudio 1 al 50% de saturación

Es claro que los resultados obtenidos en los tres primeros modelos (primer método de programación) son considerablemente mejor que los otros. Este resultado era esperable visto el porcentaje de optimalidad del gráfico anterior. Atendiendo al tiempo de ejecución promedio de cada modelo para todos los problemas propuestos, se obtiene el gráfico que se muestra en la Figura 5-14.:

Figura 5-14. Tiempo medio de ejecución de los modelos para el caso de estudio 1 al 50% de saturación

Como los modelos correspondientes al método de modelado D no han obtenido ninguna solución óptima, en las diez instancias propuestas han consumido la totalidad del tiempo límite de ejecución (600s). Ahora los tiempos medios de ejecución se han incrementado considerablemente al tratarse de problemas más complejos. Sin embargo, para una mejor comparación, de nuevo se eliminan las instancias para las que los modelos no han podido encontrar solución, y se vuelve a calcular el valor promedio de la función objetivo en el resto de problemas (Figura 5-15).

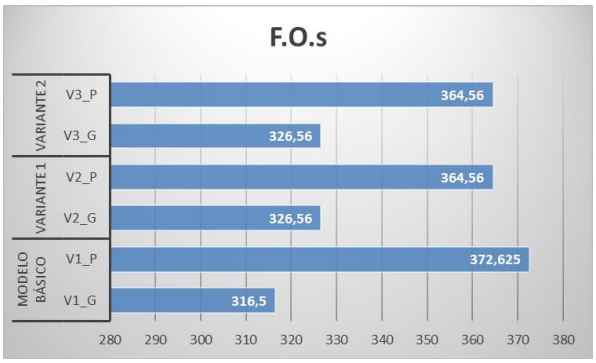

Figura 5-15. F.O. para el caso de estudio 1 al 50% de saturación (soluciones óptimas y factibles)

La diferencia, al igual que en el caso anterior, es bastante significativa, siendo el método G el que aporta mejores soluciones. No obstante, se debe contemplar en aquellos casos en los que la solución es factible, cuanto de cerca se ha quedado el modelo de la optimalidad, para lo que se representa el GAP promedio en la Figura 5-16.

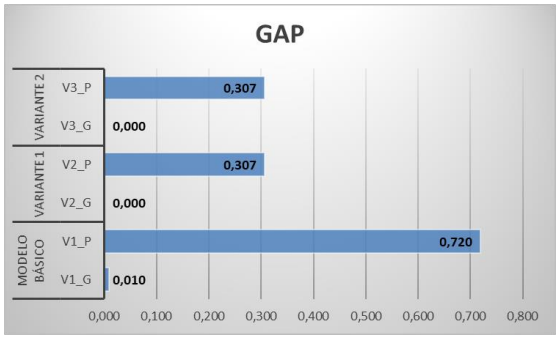

Figura 5-16. Valor promedio del GAP para el caso de estudio 1 con saturación del 50%

En el modelo básico, el método de modelado D queda bastante lejos de la solución óptima, de ahí la gran diferencia entre los valores obtenidos en la función objetivo para el primer método de programación y para éste segundo. No obstante, para las otras dos variantes, aunque no llega al óptimo, no está del todo lejos del mismo, con lo que para la configuración de recursos y el nivel de saturación propuestos, son modelos a tener en cuenta al aportar soluciones asumiblemente buenas dentro del tiempo límite. Por último, se muestra la comparativa de restricciones y variables (Tabla 5-11 y Figura 5-17) empleadas para cada modelo y cada método de programación utilizado, sirviendo la reflexión realizada en el apartado inmediatamente anterior.

	V1_G	V2_G	V3_G	V1_P	V2_P	V3_P
Número de variables	2350,7	2350,7	2350,7	2095	2095	2095
Número de restricciones	488	507,6	507,6	2647	2666,6	2666,6

Tabla 5-11. Número de variables y restricciones en el caso de estudio 1 con saturación del 50%

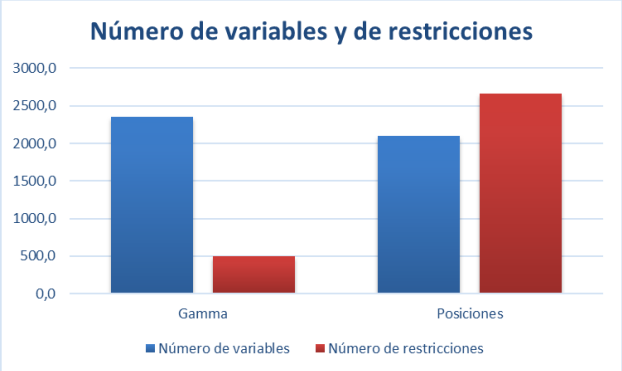

Figura 5-17. Comparativa del número de variables y restricciones del caso de estudio 1 con saturación del 50%

El número de variables y restricciones se ha duplicado y triplicado en algunos casos, lo que es normal pues al tener más actividades y pacientes, se tienen más variables y por ende más restricciones. Sin embargo, el cambio más brusco se ve en el número de restricciones del método de modelado D, que supera incluso al número de variables de método G. Como se ha comentado en el anterior apartado, esto se debe fundamentalmente a un mal ajuste del parámetro p, que podría reducirse considerablemente, eliminando variables y restricciones de estos modelos.

5.2.1.3 Nivel de saturación del 75%

Para la misma configuración de recursos, se aumenta una última vez el nivel máximo de saturación del SUH, con lo que también lo hacen de nuevo el número de pacientes que el mismo puede abordar:

| I |	| A |	| J |
6,4	25,8	10

Tabla 5-12 Pacientes, actividades y recursos del caso de estudio 1 con saturación del 75%

En este caso, los resultados obtenidos en las diez instancias propuestas son de la siguiente clase:

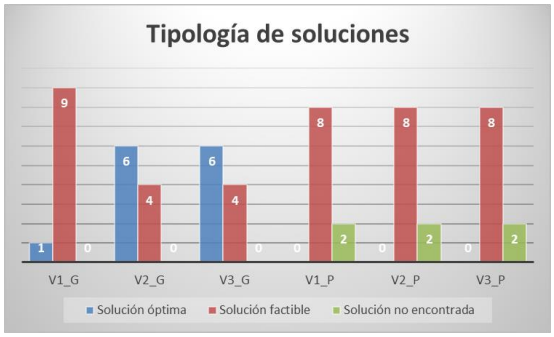

Figura 5-18. Tipología de soluciones para el caso de estudio 1 al 75% de saturación

Obviamente, al aumentar el número de pacientes al que hay que atender al admitir mayor nivel de saturación, la complejidad del problema aumenta, con lo que encontrar una solución óptima en el tiempo establecido es cada vez más difícil. Es por ello que ahora para el método de modelado G solo se han encontrado un 43% de soluciones óptimas, frente al 93% anterior. No obstante, dichos modelos siguen encontrando siempre una solución al problema propuesto, aunque no se pueda demostrar la optimalidad de las mismas. Por otro lado, los resultados que conciernen al método de modelado de discretización del dominio de las variables de decisión han variado muy poco, aunque tampoco eran buenos resultados en el caso anterior frente al primero de los métodos, no consiguiendo ninguna solución óptima, aunque con un 80% de soluciones factibles de todas las encontradas.

Respecto a los valores de la función objetivo y los tiempos de ejecución promedio, se han obtenido los siguientes resultados que se muestran en las Figuras 5-19 y 5-20.

Figura 5-19. Valor medio de la F.O. para el caso de estudio 1 al 75% de saturación

Figura 5-20. Tiempo medio de ejecución de los modelos para el caso de estudio 1 al 75% de saturación

Como consecuencia del aumento del nivel máximo de saturación, ahora los valores de la función objetivo son mayores ya que son más pacientes los que hay que atender, y ante un problema más complejo, el tiempo de ejecución también se incrementa, siendo igual al tiempo límite de ejecución en la mayoría de los casos ya que solo un 21,6% de las soluciones encontradas son óptimas, el resto, al ser factibles o al no haberse encontrado solución, consumen todo el tiempo que se le permite para resolver el problema.

Se muestra a continuación el gráfico que compara, para cada variante, ambos métodos de modelado, eliminando aquellas instancias para los que alguno de los modelos no haya podido encontrar solución, pues al no haber valor de la función objetivo, el GAP es infinito y no se puede comparar con nada:

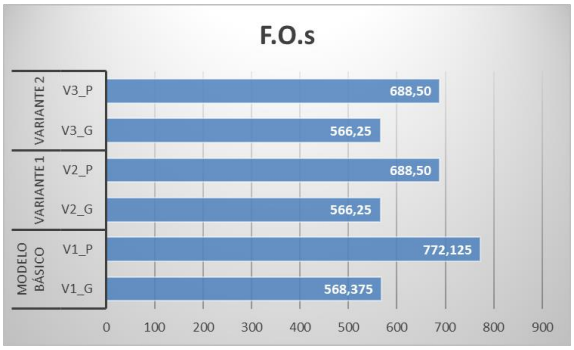

Figura 5-21. F.O. para el caso de estudio 1 al 75% de saturación (soluciones óptimas y factibles)

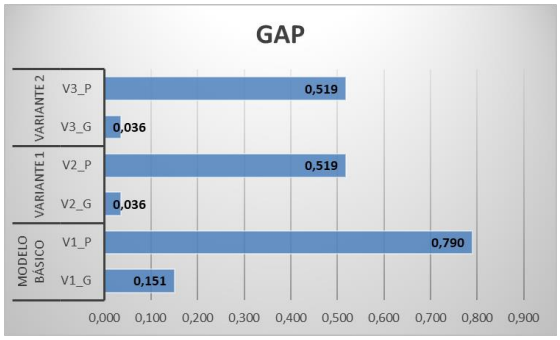

Figura 5-22. Valor promedio del GAP para el caso de estudio 1 con saturación del 75%

Al tratarse de valores más complejos, la diferencia entre ambos métodos cada vez se hace más reseñable. Por un lado, las soluciones factibles encontradas por cada modelo, están ahora más lejos del óptimo que en los dos casos anteriores, donde para alguno de los modelos el GAP era nulo al conseguir encontrar el 100% de soluciones óptimas. Además, para el método de programación D, los resultados factibles cada vez están más lejos de los que encuentra el primer método de programación, y por ende más lejos de la optimalidad.

La mayor diferencia entre ambos métodos está en el modelo básico, que como se ha comentado es un modelo abierto a infinitas soluciones, sin limitaciones demasiado restrictivas que reduzcan el campo de soluciones a menor rango, como en las otras variantes donde al exigir ciertas condiciones, en 10 minutos los modelos pueden conseguir llegar a una solución que aunque no óptima, se acerque al mismo de manera aceptable.

Ya por último solo queda ver cuantas variables y restricciones se han tenido de media en los problemas resueltos para este caso, que se representan en la Tabla 5-13 y Figura 5-23.

	V1_G	V2_G	V3_G	V1_P	V2_P	V3_P
Número de variables	3794,4	3794,4	3794,4	3417	3417	3417
Número de restricciones	826	851,5	851,5	6524	6549,5	6549,5

Tabla 5-13. Número de variables y restricciones en el caso de estudio 1 con saturación del 25%

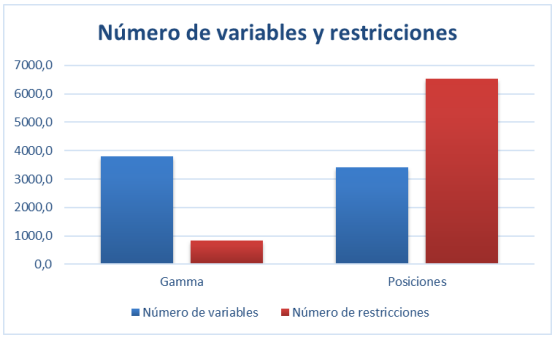

Figura 5-23. Comparativa del número de variables y restricciones del caso de estudio 1 con saturación del 75%

De nuevo la diferencia es muy significativa. El primer método (G) siempre va a tener un número mayor de variables que el segundo al contar con la variable γ, aunque la diferencia no es excesivamente grande. Sin embargo, la diferencia en el número de restricciones es muy significativa, y es el factor clave de los malos resultados que otorga el segundo de los métodos de programación. Al tener tantas restricciones, hace que diez minutos no sea tiempo suficiente para poder llegar a una solución tan buena como la del primer método al tener ocho veces más restricciones que satisfacer que dicho método aún teniendo menos variables en el problema.

5.2.2 Caso de estudio 2: "12:00 p.m."

Al igual que para el primer caso, en este segundo escenario solo serán evaluados los modelos básicos con ambos métodos de modelado, así como las variantes de asignación de prioridades a recursos y de asignación de actividades maestras a recursos para ambos métodos también. Se recuerda a continuación los recursos que se tienen en dicho horario:

12:00 a.m.	Consulta	Enfermería	Laboratorio	Sala rayos	CT-Scan
Número de recursos	4	8	1	1	1

Tabla 5-14. Número de recursos por tipología del caso de estudio 2 [6]

5.2.2.1 Nivel de saturación del 25%

Para un nivel de saturación bastante bajo del SUH y bajo la configuración de recursos indicada, se han generado otras diez instancias aleatorias, en las que se han obtenido un promedio de los siguientes pacientes y actividades de los mismos:

| I |	| A |	| J |
7,3	32,4	15

Tabla 5-15. Pacientes, actividades y recursos a las 12:00 con saturación del 25%

Es claro que al aumentar el número de recursos disponibles en 3 consultas y 2 enfermerías más, aumenta el número de pacientes que se generan antes de alcanzar el nivel de saturación propuesto, y con ello también lo hace el número de actividades que habrá en total dentro del SUH. En el siguiente gráfico se muestra como han respondido los modelos en lo que a la tipología de soluciones obtenidas se refiere para el total de diez instancias generadas:

Figura 5-24. Tipología de soluciones para el caso de estudio 2 al 25% de saturación

Al aumentar el número de recursos disponibles, los modelos no han podido encontrar ni siquiera una solución óptima. Es decir, que al aumentar el número de recursos, la complejidad del problema asciende hasta un nivel en que ninguno de los modelos es capaz de encontrar la solución óptima, o asegurar que la solución encontrada es óptima, en el tiempo límite de ejecución establecido. Asimismo, el número de soluciones no encontradas también asciende respecto a la cifra anterior del primer caso de estudio. No obstante, para un nivel de saturación del 25%, los modelos son capaces de dar una solución factible en la mayoría de los casos. El valor promedio de la función objetivo para cada modelo en las distintas instancias es el que se muestra en la Figura 5-25.

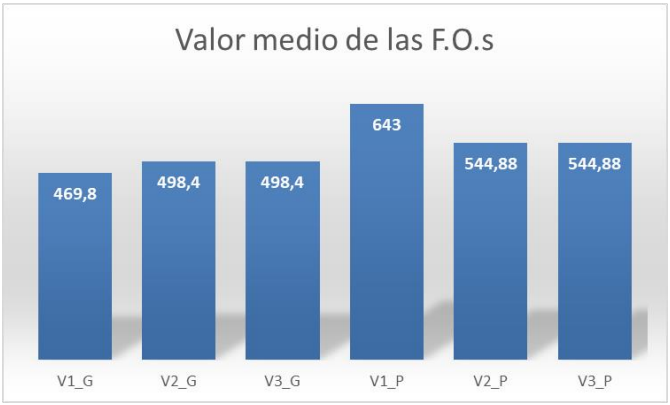

Figura 5-25. Valor medio de la F.O. para el caso de estudio 2 al 25% de saturación

En promedio, teniendo en cuenta para cada modelo solo aquellas soluciones en las que el modelo ha encontrado solución, se ve claramente que el método de modelado G encuentra una solución mejor que el segundo de los métodos. No obstante, el tiempo de ejecución obtenido en todas las soluciones de todas las instancias y para todos los modelos es de 600 segundos, ya que en ninguno de los casos se ha llegado al óptimo.

Para una mejor visualización de la comparación de métodos, se muestran en la Figura 5-26 los gráficos que enfrentan cada método para cada una de las variantes, teniendo en cuenta solo y únicamente aquellas instancias en las que ambos métodos de programación han obtenido alguna solución (eliminando aquellas en las que el modelo no ha encontrado solución en el tiempo límite):

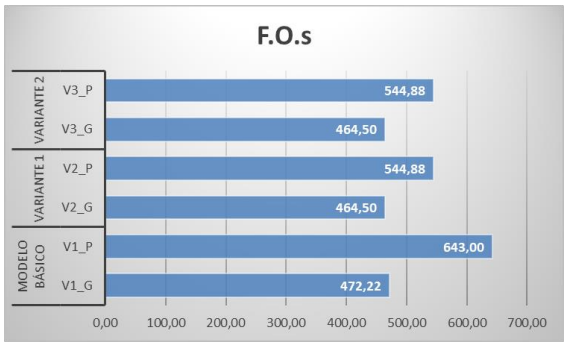

Figura 5-26. F.O. para el caso de estudio 2 al 25% de saturación (soluciones óptimas y factibles)

De nuevo se demuestra que las soluciones que obtiene el primer método de modelado son mucho mejores que las que obtiene el método D planteado para un mismo tiempo de ejecución 10 minutos. Asimismo, los valores de la función objetivo han aumentado su valor respecto al primer caso de estudio, lo que es normal al tener que ubicar a un mayor número de pacientes con más actividades dentro del SUH. Se analiza ahora el GAP para todas las soluciones factibles en el siguiente gráfico:

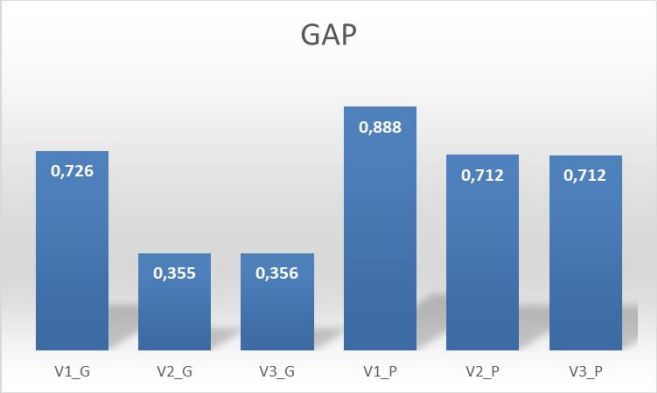

Figura 5-27. Valor promedio del GAP para el caso de estudio 2 con saturación del 25%

Al igual que se ha analizado con el valor de la función objetivo, para un mejor análisis se muestra la comparativa del valor promedio del GAP para cada variante en función de los métodos de programación utilizados teniendo en cuenta solo aquellas instancias para las que ambos métodos han encontrado una solución factible:

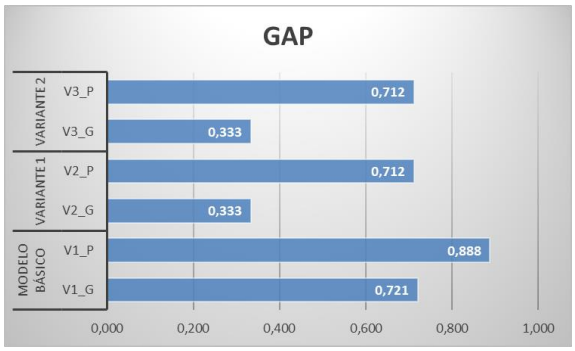

Figura 5-28. Valor promedio del GAP para el caso de estudio 2 con saturación del 25%

El resultado obtenido del parámetro GAP es razonable dado que como los valores de la función objetivo obtenidos para el método G eran menores que las obtenidos en el método D, quiere decir que el primero de los métodos se queda más cerca de la solución óptima del problema planteado, con lo que el GAP será menor, tal y como se muestra en las gráficas. Dicha diferencia se incrementa significativamente al implementar las variantes, que dejan una solución no tan abierta como la del modelo básico, al exigir restricciones en las que los pacientes solo pueden ir a según que recursos. Al no tener tantos casos que contemplar en 10 minutos de tiempo de ejecución, se llega a una mejor solución (menor valor de la función objetivo) en el mismo tiempo, lo que también ocurre al implementar el segundo método de modelado pero en menor medida.

Por último, también se puede comparar el número de variables y de restricciones que se han obtenido de media para ambos métodos de programación, que se muestran en la Tabla 5-16 y Figura 5-29.

	V1_G	V2_G	V3_G	V1_P	V2_P	V3_P
Número de variables	8758	8758	8758	7930	7930	7930
Número de restricciones	2562	2594	2594	24763	24795	24795

Tabla 5-16. Número de variables y restricciones en el caso de estudio 2 con saturación del 25%

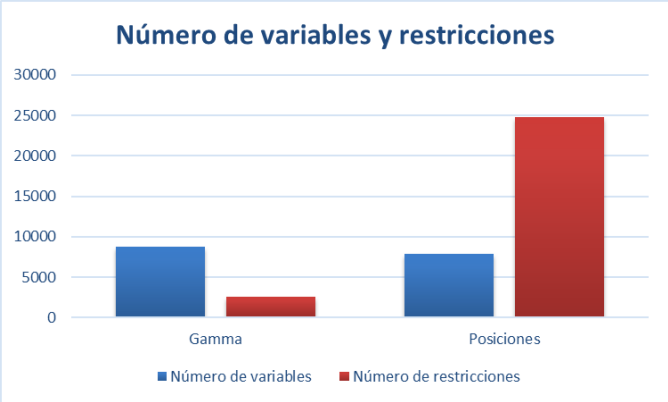

Figura 5-29. Comparativa del número de variables y restricciones del caso de estudio 2 con saturación del 25%

De nuevo se visualiza que, aunque para el método de modelado G el número de variables que componen el modelo es ligeramente mayor, la gran diferencia que existe en el número de restricciones que hay en cada modelo hace que, en términos generales, el método de programación que emplea la variable γ sea mejor aún teniendo más variables que resolver. Esto también se ve reflejado en que para el método D, el valor obtenido en la función objetivo siempre es peor, ya que para el tiempo de ejecución establecido no es capaz de llegar a la misma solución que el primero de los métodos.

5.2.2.2 Nivel de saturación del 50%

Continuando con la misma distribución de recursos, se aumenta ahora el nivel de saturación al doble, lo que supone problemas más complejos con mayor número de pacientes a los que asignar recursos dentro de su proceso de urgencias:

| I |	| A |	| J |
13,1	58,9	15

Tabla 5-17. Pacientes, actividades y recursos del caso de estudio 2 con saturación del 50%

Los resultados obtenidos en lo referente a la tipología de soluciones se presentan en el siguiente diagrama:

Figura 5-30. Tipología de soluciones para el caso de estudio 2 al 50% de saturación

Como se puede observar, no se ha obtenido ni una solución óptima. Si se vuelve la mirada atrás al caso de estudio 1 para este mismo nivel de saturación, es fácil comprobar como los resultados han empeorado considerablemente. No obstante, para el primero de los métodos de modelado, el resultado obtenido sigue siendo bueno, con un 97% de las soluciones obtenidas factibles, es decir, que dentro del horizonte de programación los modelos han conseguido encontrar al menos una solución factible para el problema. Sin embargo, para el método D, los resultados son peores, con un 56% de soluciones no encontradas.

Para estas nuevas diez instancias, el valor promedio de la función objetivo obtenido para los distintos modelos,

aunque haya un elevado número de soluciones no encontradas, es:

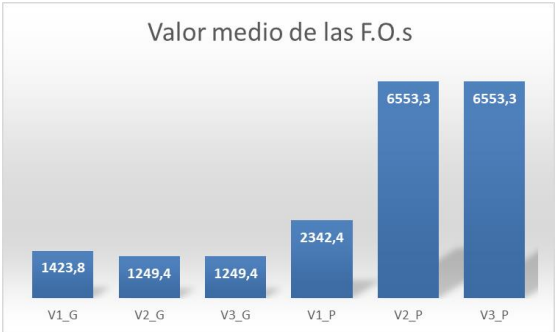

Figura 5-31. Valor medio de la F.O. para el caso de estudio 2 al 50% de saturación

Un apunte importante es el hecho de que todos los tiempos de ejecución han sido igual al tiempo límite establecido en 600 segundos, ya que en ninguno de los casos ninguno de los modelos ha encontrado una solución óptima. La diferencia entre unos modelos y otros se ha disparado, especialmente en las variantes V2_P y V3_P, debido a que en algún caso, estos modelos han encontrado una solución factible, pero en la que los pacientes salen del SUH en un instante excesivamente tardío, haciendo que el valor del parámetro LOS de la función objetivo también se dispare. No obstante, en la Figura 5-32 se comparan los valores de la función objetivo solo en aquellas instancias en las que ambos modelos han encontrado solución factible:

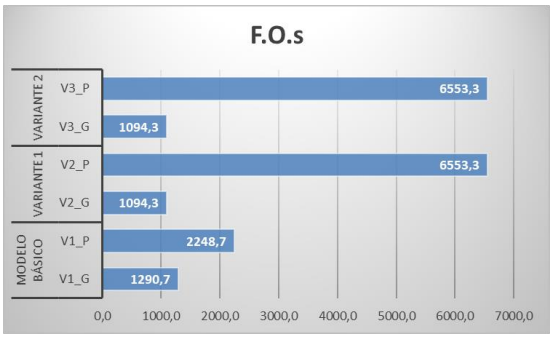

Figura 5-32. F.O. para el caso de estudio 2 al 50% de saturación (soluciones óptimas y factibles)

La diferencia se hace aún más grande si cabe. Esto hace esperar que, si se atiende al valor del parámetro GAP obtenido, el método de modelado D tenga un valor del mismo bastante alto y cercano a la unidad en las soluciones factibles encontradas, mientras que ocurrirá lo contrario para el método G. En la Figura 5-33, se representa el valor medio obtenido del parámetro GAP para todos los modelos.

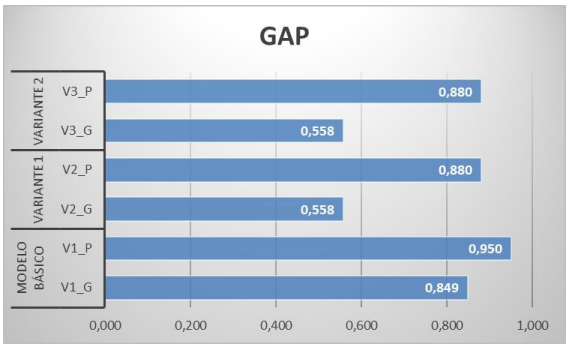

Figura 5-33. Valor promedio del GAP para el caso de estudio 2 con saturación del 50%

El valor del parámetro que se analiza ahora (GAP), también ha empeorado considerablemente con la complejidad del problema. Los modelos no responden de la misma forma que hacían en el primer caso de estudio para este nivel de saturación. Para el modelo básico, como ocurría anteriormente, al tener tantas posibles soluciones, en 10 minutos no consigue obtener de media una solución que se acerque a la optimalidad. En las otras dos variantes, las soluciones son mejores, pero también quedan lejos de una hipotética solución óptima.

Seguidamente, se muestran el número de variables y restricciones en la Tabla 5-18 y Figura 5-34 que se han obtenido de media en este caso de estudio. Se vuelve a remarcar el hecho de que la mala calibración del parámetro p hace que se eleve el número de restricciones y variables en este método de programación, ya que posiblemente los problemas podrían ser resueltos para un número menor de dicho parámetro.

	V1_G	V2_G	V3_G	V1_P	V2_P	V3_P
Número de variables	27583	27583	27583	23950	23950	23950
Número de restricciones	6711	6769,4	6769,4	131173	131232	131232

Tabla 5-18. Número de variables y restricciones en el caso de estudio 2 con saturación del 50%

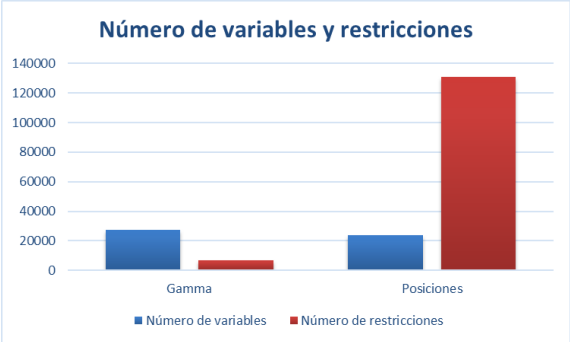

Figura 5-34. Comparativa del número de variables y restricciones del caso de estudio 2 saturación del 50%

La diferencia es cada vez más notoria, siendo el número de restricciones del método de modelado D excesivamente grande. No obstante, aún con la mala calibración del parámetro p, es visual e inmediato ver que la diferencia entre el número de variables de uno y otro siempre es mínima, mientras que la del número de restricciones es demasiado voluminoso.

5.2.2.3 Nivel de saturación del 75%

La tercera situación del caso de estudio 2 implica incrementar el nivel de saturación del SUH hasta un 75%, llevando el número de pacientes que van a ser atendidos hasta los siguientes datos:

| I |	| A |	| J |
18,4	83,7	15

Tabla 5-19. Pacientes, actividades y recursos del caso de estudio 2 con saturación del 75%

Esta es la situación que va a generar los problemas más complejos que se van a resolver en este TFG, llevando a un extremo a los modelos para ver como responden ante situaciones de alta dificultad. Primeramente, los resultados obtenidos presentan la tipología que se muestra en la Figura 5-35.

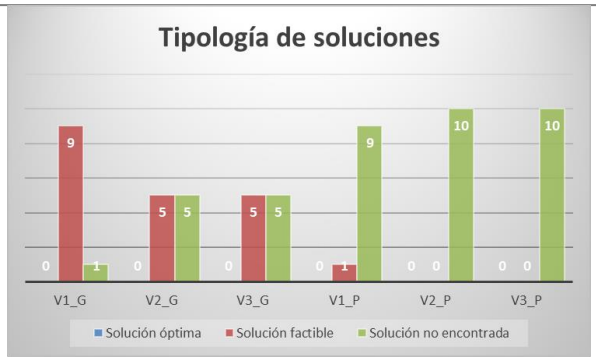

Figura 5-35. Tipología de soluciones para el caso de estudio 2 al 75% de saturación

Ahora los modelos solo son capaces de aportar una solución factible en un 33% de los casos, y por supuesto, ninguna solución óptima. Al tener 15 recursos y una media de 18 pacientes en cada problema, los modelos de programación lineal generados no son capaces de encontrar solución dentro del tiempo límite de ejecución en la mayoría de los casos. Así pues, solo se pueden analizar los valores de la función objetivo (Figuras 5-36 y 5-37) obtenidos utilizando el método de modelado G (ya que la única solución encontrada en V1_P tiene un valor del GAP de 0,99, y queda muy lejos de lo que se está buscando, así que se va a omitir).

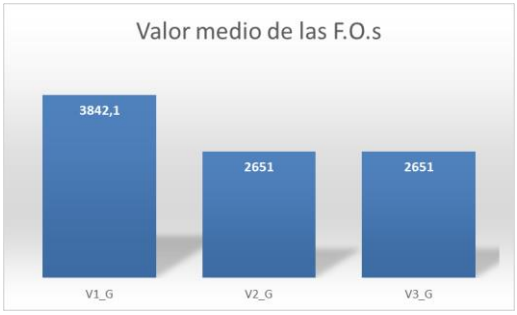

Figura 5-36. Valor medio de la F.O. para el caso de estudio 2 al 75% de saturación

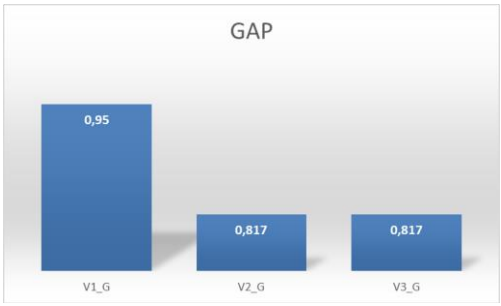

Figura 5-37. Valor promedio del GAP para el caso de estudio 2 con saturación del 75%

Atendiendo tanto al valor medio de la función objetivo obtenido como al GAP, se ve claramente como las variantes 1 y 2 obtienen mejor solución que el modelo básico, aunque los tres modelos quedan bastante lejos de la optimalidad. Se muestran para este último caso el número de variables y restricciones promedio de los que se han compuesto las diez instancias analizadas en la Tabla 5-20 y Figura 5-38.

	V1_G	V2_G	V3_G	V1_P	V2_P	V3_P
Número de variables	54669	54669	54669	48396	48396	48396
Número de restricciones	12845	12928	12928	389632	389715	389715

Tabla 5-20. Número de variables y restricciones en el caso de estudio 2 con saturación del 75%

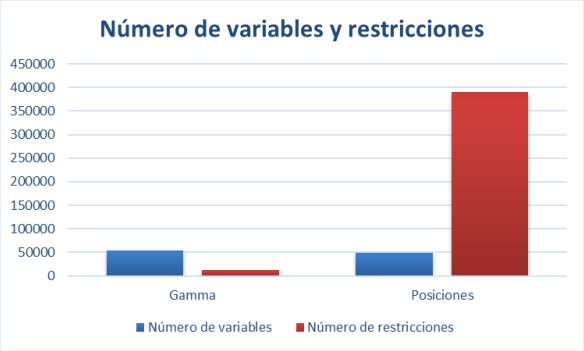

Figura 5-38. Comparativa del número de variables y restricciones del caso de estudio 2 con saturación del 75%

5.2.3 Elección del mejor método de modelado

Antes de seguir con la experimentación y evaluar la última de las variantes para una determinada configuración de los recursos, los resultados obtenidos hasta el momento son suficientes para poder extraer conclusiones respecto a cuál de los dos métodos de modelado es mejor: la que emplea la variable γ para la secuenciación del proceso de urgencias; o la que divide la disponibilidad de los recursos en posiciones utilizando el parámetro p (discretización del dominio de las variables de decisión).

Se muestra a continuación dos gráficas que muestran una media del número de variables y restricciones que han presentado cada uno de los métodos (teniendo en cuenta los tres modelos con los que se ha trabajado hasta ahora en la experimentación), así como un resumen de la tipología de recursos que ha obtenido cada uno:

Figura 5-39. Comparativa de ambos métodos en tipología de soluciones

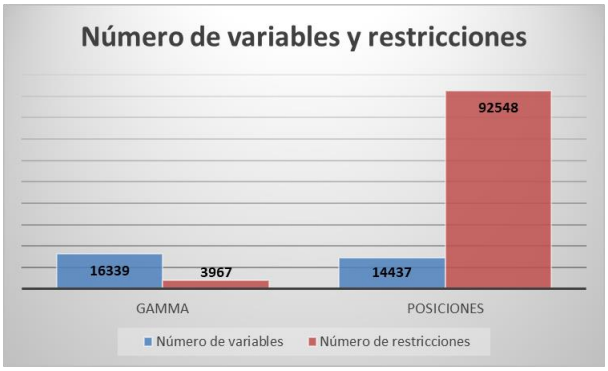

Figura 5-40. Comparativa de ambos métodos en número de variables y restricciones

La conclusión es muy visual, y es que mientras el método G obtiene un 39% de soluciones óptimas y un 54% de soluciones factibles, con tan solo un 7% de problemas en los que no ha encontrado solución; el segundo método (D) solo llega a la optimalidad en un 8% de los casos, a una solución factible en un 56% y no

encuentra solución para un 36% de los problemas planteados. Es decir, que los resultados que el método (G) aporta para los problemas, son mucho mejores que los que da el segundo de ellos, que numerosas veces ni siquiera alcanza una solución factible.

Esto se puede deber fundamentalmente, como se ha mencionado varias veces a lo largo del documento, a que el parámetro p que determina en cuantas posiciones se divide la disponibilidad de los recursos del problema, hace que en este método de programación se generen variables de más, ya que hay posiciones que no son necesarias, y por ende se generan un gran número de restricciones vinculadas a esas variables innecesarias que hacen que el tiempo de cómputo para llegar a una solución factible utilizando este método deba ser mayor que el que se propone en este proyecto. No obstante, la reducción del parámetro p también puede conducir la experimentación directamente a la infactibilidad, al no haber posiciones suficientes en un recurso para asignarle las actividades necesarias.

Se recuerda que el tiempo límite de computación está fijado en diez minutos ya que se busca que los modelos den una respuesta dentro del margen que se tiene para atender al paciente más desfavorable en primera consulta, que es el paciente de prioridad el cual tiene un TEPCOF de 15 minutos (se asume que los pacientes de prioridad 1 son atendidos directamente nada más llegar al SUH). Se dejan 5 minutos de margen para que cuando el modelo emita la solución, está pueda ser llevada a cabo en el hospital antes de llegar al tiempo límite.

Asimismo, solo con una instancia que se tomó de prueba para desarrollar todos los modelos y comprobar que las restricciones estaban funcionando a través de la visualización de la solución en un diagrama de Gantt, ya se podía ver que el método D, aunque alcanzaba el óptimo, lo hacía en mucho más tiempo que el método G, aun cogiendo el valor de *p* óptimo. No obstante, quedaba iniciar la experimentación para poder desacreditar el segundo de los métodos, como se ha podido demostrar.

A modo de conclusión, ambos métodos pueden resolver el problema de organización a nivel operativo del SUH, pero el segundo método exige más trabajo para equilibrar el parámetro p, y supone mayor número de restricciones que el primer método, lo que conlleva que necesite un tiempo de computación más elevado. Como ambos métodos cumplen con la misma función pero el primero de ellos lo hace de manera más eficiente, para el tercer caso de estudio solo se va a utilizar el método de modelado G para la secuenciación del proceso de urgencia de los pacientes en cuestión.

5.2.4 Caso de estudio 3: “12:00 p.m. con variantes”

Para este tercer y último caso de estudio, se situará la configuración de recursos en el mismo punto horario que el del caso de estudio 2 según el artículo [6], sin embargo, cambiarán las asunciones:

- ✓ Se tienen dos laboratorios para las pruebas de extracción de sangre siempre disponibles.
- ✓ Se tienen dos salas de radiodiagnóstico (rayos X) siempre disponibles.
- ✓ Se tiene dos salas para pruebas TAC (CT-Scan, Computed-Tomography Scan) siempre disponibles.

Esto se hace para poder evaluar la tercera de las variantes, donde los pacientes son distribuidos según su tipología. Para poder abordar esta variante es necesario que haya al menos dos unidades de cada tipología de recurso para poder distinguir al menos entre dos tipos de pacientes. Otra forma de abordarlo, sería considerar tiempos de set-up que hay que tener en cuenta a la hora de llevar a cabo la planificación y asignación de recursos a actividades ya que aunque un paciente haya acabado su actividad, el siguiente paciente debe esperar hasta la limpieza y desinfección del recurso utilizado en caso de que el paciente predecesor sea de distinta tipología.

En el caso de este TFG, se ha optado por la duplicidad de recursos. Más concretamente, se va a distinguir entre dos tipos de pacientes atendiendo a una situación que sigue muy presente en nuestras vidas, la conocida comúnmente como pandemia del COVID-19. Por tanto, dentro del SUH habrá pacientes contagiados por dicho virus, y otros que no, y por tanto habrá que sectorizar el SUH de modo que los pacientes no contagiados nunca sean atendidos en recursos en los que pueden ser atendidos pacientes que si están contagiados. Esta situación se abordará en la variante número 3, la más restrictiva de todas las propuestas.

Así pues, se recuerda la configuración de recursos para este caso de estudio:

12:00 a.m.	Consulta	Enfermería	Laboratorio	Sala rayos	CT-Scan
Número de recursos	4	8	2	2	2

Tabla 5-21. Número de recursos por tipología del caso de estudio 3 [6]

5.2.4.1 Nivel de saturación del 25%

Este es el caso con mayor número de recursos de los tres estudiados, con lo que cabe esperar que el número de pacientes para un mismo nivel de saturación aumente, siendo muy similar al obtenido en el caso de estudio 2, ya que los recursos que tienden a saturarse primero son las consultas y las enfermerías ya que tienen que acoger a los pacientes más de una vez a lo largo de sus procesos de urgencia. En la siguiente tabla se muestra la media de pacientes y actividades que se han generado en los diez problemas de este primer caso:

| I |	| A |	| J |
9,4	39,1	18

Tabla 5-22. Pacientes, actividades y recursos para el caso de estudio 3 al 25% de saturación

Las soluciones obtenidas para cada modelo se muestran en la Figura 5-41.

Figura 5-41. Tipología de soluciones para el caso de estudio 3 al 25% de saturación

En este caso, todos los modelos han podido encontrar solución a todos los problemas propuestos. El modelo de la variante número 3 de asignación por tipología de pacientes ha sido capaz incluso de alcanzar la optimalidad hasta en dos ocasiones, a pesar del gran número de recursos que presenta este tercer caso de estudio. Los valores promedio obtenidos de la función objetivo se exponen en la Figura 5-42.

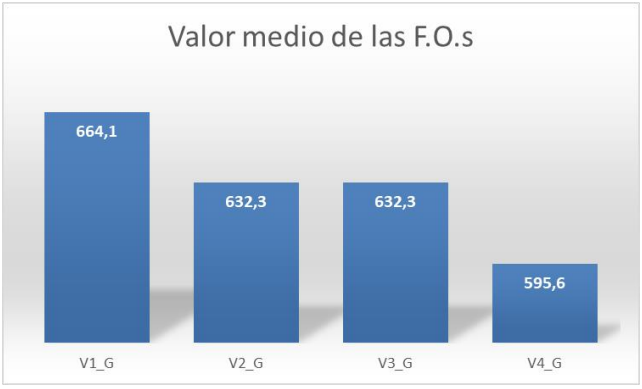

Figura 5-42. Valor medio de la F.O. para el caso de estudio 3 al 25% de saturación

Aquí cabe hacer una reflexión. Realmente para los cuatro modelos se tiene el mismo problema: *I* pacientes con *A* actividades que deben ser asignadas a *J* recursos de forma que se cumpla el TEPCOF y que los pacientes estén el menor tiempo posible dentro del SUH. Si no hubiera COVID, la solución que dicha variante plantea es totalmente válida también para el resto de modelos, y puede interesar en caso de que el valor de la función objetivo que se obtiene sea menor que el obtenido por los otros modelos.

Lo normal cuando se añaden restricciones a un problema es que el valor de la función objetivo empeore, ya que se está limitando el rango de soluciones. Sin embargo, en este problema tan complejo con un rango tan amplio de posibles soluciones, el hecho de añadir restricciones que limitan dicho rango, hace que, como el ordenador no tiene tantas soluciones que chequear, éste consiga llegar a una solución mejor en el mismo tiempo. Es por ello que el modelo correspondiente a la variante número 4 es el que, de media, consigue el mejor valor de la función objetivo, y con bastante diferencia respecto a los demás.

Otro aspecto interesante de analizar es, dentro de las soluciones factibles encontrada, cual de los cuatro modelos se ha quedado más cerca de alcanzar la optimalidad, lo que es analizable a través del parámetro GAP (Figura 5-43).

Figura 5-43. Valor promedio del GAP para el caso de estudio 3 con saturación del 25%

Es decir, el modelo más restrictivo (distribución por tipología de pacientes), no solo es el que mejor valor de la función objetivo aporta de media, sino que también es el que dentro de las soluciones factibles, queda más cerca del óptimo. Este gráfico sirve para refutar lo mencionado en los párrafos anteriores, y es que al modelo básico, al tener tantas posibles soluciones, le resulta mucho más difícil que al resto llegar al óptimo o ni siquiera acercarse a éste. Mientras que lo contrario le ocurre al modelo cuatro, que al tener menos soluciones chequeables, queda bastante más cerca de dicho óptimo.

Llegados a este punto del trabajo, el número de variables y restricciones se convierte en un dato irrelevante para conclusiones, aunque se introduce por si resulta de interés (Figura 5-44).

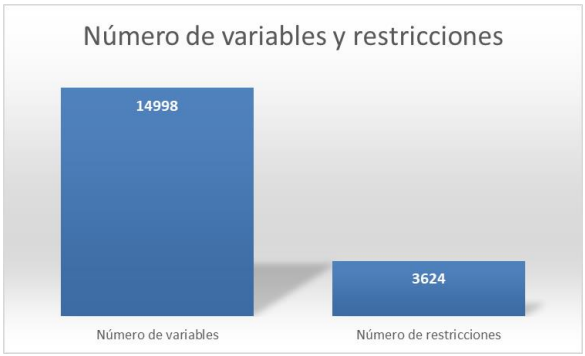

Figura 5-44. Número de variables y restricciones en el caso de estudio 3 con saturación del 25%

5.2.4.2 Nivel de saturación del 50%

Ahora se incrementa, al igual que los dos casos de estudio anteriores, el nivel de saturación hasta el 50%, obteniéndose una media de número de pacientes y actividades tal que:

|I|	|A|	|J|
16,8	71,7	18

Tabla 5-23. Pacientes, actividades y recursos para el caso de estudio 3 al 50% de saturación

Así pues, el problema pasa a tener el doble de pacientes y de actividades, con lo que la complejidad del problema en cuestión también aumenta. En lo que a la tipología de soluciones obtenidas se refiere, se representa en la Figura 5-45.

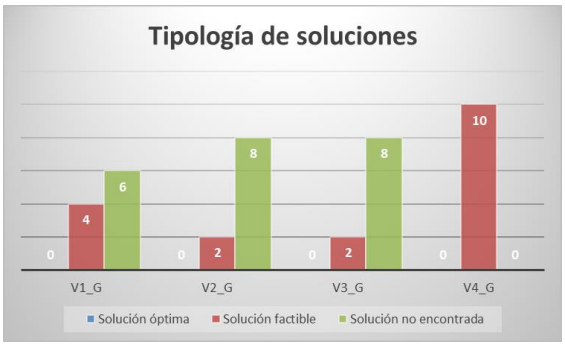

Figura 5-45. Tipología de soluciones para el caso de estudio 3 al 50% de saturación

Los resultados han empeorado considerablemente, pasando del pleno en soluciones óptimas y factibles en el caso anterior, a solo un 45% de soluciones encontradas. Es decir, que 22 de las 40 soluciones son de tipo no encontrada. No obstante, la variante de distribución por tipología de pacientes sigue dando los mejores resultados, encontrando una solución al problema planteado en todos los casos, aunque sin demostrar la optimalidad de las mismas. En cuanto al valor de la función objetivo y al GAP obtenido, en las Figuras 5-46 y 5-47 se muestra el valor medio que cada modelo ha proporcionado.

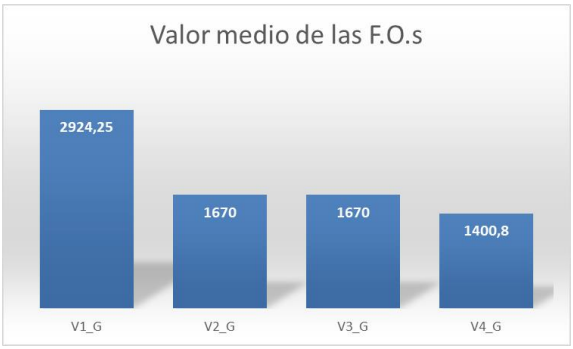

Figura 5-46. Valor medio de la F.O. para el caso de estudio 3 al 50% de saturación

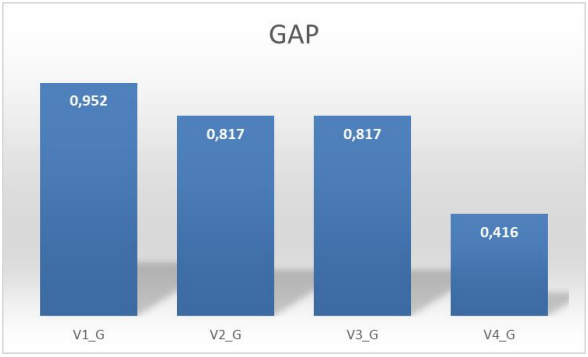

Figura 5-47. Valor promedio del GAP para el caso de estudio 3 con saturación del 50%

Por los mismos motivos que los explicados en el punto inmediatamente anterior, el modelo básico lejos queda de una solución óptima, al tener un valor medio de la función objetivo que duplica al que encuentran los otros modelos y un valor medio del GAP cercano a la unidad. Asimismo, la última de las variantes es capaz de encontrar el mejor valor de la función objetivo para los problemas, aunque en este caso con un valor más alto del GAP, con lo que al aumentar la saturación, las soluciones encontradas son peores y quedan más lejos de la solución óptima.

Por otro lado, de los diez problemas planteados, solo hay uno para el que los cuatro modelos han encontrado solución, y otro de los problemas en el que tres de los cuatro modelos han también llegado a la factibilidad. Se muestran los resultados de esos problemas:

		V1_G	V2_G	V3_G	V4_G
Instancia 1	*Valor F.O.*	2393	1648	1648	1180
	GAP	0,9412	0,8477	0,8477	0,299
Instancia 10	*Valor F.O.*	None	1692	1692	1436
	GAP	inf	0,7874	0,7875	0,467

Tabla 5-24. Resultados instancias 1 y 10 en el caso de estudio 3 con nivel de saturación del 50%

Realizando la comparativa con solo aquellos problemas en las que la mayoría de modelos han encontrado solución se ve más claro aún que el cuarto modelo consigue llegar a una solución bastante mejor que los otros tres, y por tanto más cerca del óptimo. Por último, la Figura 5-48 muestra el número de variables y restricciones obtenidas en este caso.

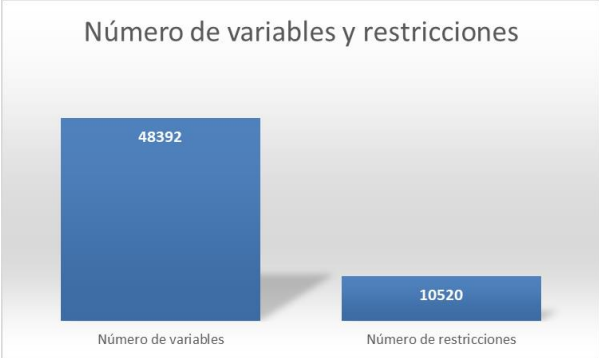

Figura 5-48. Número de variables y restricciones en el caso de estudio 3 con saturación del 50%

5.2.4.3 Nivel de saturación del 75%

Por ultimo, el nivel de saturación asciende hasta el 75%, de modo que los recursos tienen capacidad para asistir a una media de pacientes tal que:

|I|	|A|	|J|
24,5	103,9	18

Tabla 5-25. Pacientes, actividades y recursos para el caso de estudio 3 al 75% de saturación

Este caso es el más complejo, al ser el caso en que mayor media de pacientes y actividades deben ser

atendidos. Así pues, cabe esperar que en lo que a tipología de soluciones encontradas se refiere, también se haya empeorado debido al alto nivel de complejidad que suponen estos problemas (Figura 5-49).

Figura 5-49. Tipología de soluciones para el caso de estudio 3 al 75% de saturación

Para esta configuración de recursos y el nivel de saturación establecido, los tres primeros modelos no son capaces de encontrar una solución dentro del tiempo límite de computación establecido. El cuarto modelo si llega a soluciones factibles la mayoría de veces, pero hay que analizar si dichas soluciones son buenas o si por el contrario quedan excesivamente lejos del óptimo. Para ello, en la siguiente tabla se muestran el valor medio de la función objetivo y del parámetro GAP para el cuarto de los modelos de programación lineal:

	V4_G
Valor medio de la F.O.	2745
GAP	0,637

Tabla 5-26. Valor de la F.O. y GAP del modelo 4 para el caso de estudio 3 al 75% de saturación

El valor del parámetro GAP asciende respecto al caso anterior, lo que quiere decir que al aumentar la saturación, las soluciones factibles que dicho modelo encuentran se van alejando cada vez más del óptimo que se busca alcanzar. En la siguiente tabla, se muestran los resultados del único problema en el que los cuatro modelos han sido capaces de al menos encontrar una solución factible, para ver el contraste entre las soluciones que cada uno de ellos ofrece:

		V1_G	**V2_G**	**V3_G**	**V4_G**
Instancia 3	*Valor F.O.*	7949	6325	6325	3145
	GAP	0,971	0,9498	0,9498	0,687

Tabla 5-27. Resultados instancia 3 en el caso de estudio 3 con nivel de saturación del 75%

Es claro que para este problema, el cuarto modelo llega a una solución mucho mejor que los otros tres, siendo la solución del modelo básico una solución factible pero que muy lejos queda de ser una solución buena para

Figura 5-50. Número de variables y restricciones en el caso de estudio 3 con saturación del 75%

la organización del SUH. Para finalizar, la Figura 5-50 muestra el número de variables y restricciones promedio obtenidos en estos problemas.

5.2.5 Términos de la función objetivo

A lo largo de toda la experimentación, se han resuelto un total de 90 instancias, de las cuales 60 de ellas han sido resueltas para seis de los modelos de programación lineal creados (modelo básico y variantes 1 y 2 de ambos métodos de programación), y las otras 30 han sido resueltas para cuatro de estos modelos (el modelo básico y las tres variantes utilizando el método de modelado G para la secuenciación del proceso de urgencias).

Del total de todas las soluciones obtenidas, la tipología de todas ellas se resume en la Figura 5-51.

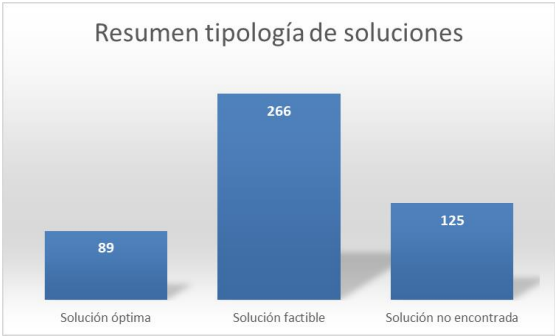

Figura 5-51. Tipología de soluciones de toda la experimentación

Es decir, que 355 de las 480 soluciones son factibles u óptimas en las que hay un valor de la función objetivo, sin importar lo lejos que estén dichos valores de la optimalidad (74% de las soluciones). Se recuerda cual era la función objetivo del problema planteado:

$$Min. \sum_{i=1}^{I} T_i + \sum_{i=1}^{I} \sum_{j=1}^{J} CT_{last_i,j}$$

El primero de los términos hace referencia al cumplimiento del TEPCOF, mientras que el segundo de ellos concierne al LOS de cada paciente. Es claro que el LOS nunca podrá tomar valor nulo ya que el instante en que el paciente abandone el SUH siempre será un valor mayor que la nulidad. Sin embargo, minimizar el primer término de la función objetivo conlleva hacer que el mayor número de pacientes posibles cumplan el TEPCOF, de modo que, en una situación ideal, si todos los pacientes que hay en el SUH han cumplido con su TEPCOF, el valor de la variable T_i si tomará valor nulo.

Que el término de la función objetivo de la variable T_i tome valor mayor que cero, implica que al menos un paciente dentro del SUH no ha cumplido con su TEPCOF. Dicho término (T_i) tomará valores más grandes en dos casos: si el único paciente que no ha cumplido el TEPCOF lo ha hecho con mucha diferencia (por ejemplo, un paciente de prioridad 2 que tiene que ser atendido en 15 minutos y no ha sido atendido hasta el minuto 60); o bien si son varios los pacientes que no han sido atendidos dentro de su TEPCOF, haciendo que haya varias variables T_i con valor distinto a la nulidad.

Pues bien, del total de las 355 soluciones factibles, los resultados obtenidos de la variable T_i se muestran en la siguiente tabla:

$\sum_{i=1}^{I} T_i = 0$	329 veces
$0 < \sum_{i=1}^{I} T_i < 50$	17 veces
$\sum_{i=1}^{I} T_i > 50$	9 veces

Tabla 5-28. Estadísticas del término TEPCOF en la función objetivo

En resumen, en un 93% de los casos en que se ha encontrado solución, todos los pacientes han cumplido con

su TEPCOF, con lo que el valor de la función objetivo se limita al valor del sumatorio del LOS de todos los pacientes. Solo en un 2,5% de los casos, el valor que dicho término de la función objetivo se ha disparado por encima del valor de 50, lo que quiere decir que o bien más de un paciente a incumplido el TEPCOF, o hay alguno de los pacientes que ha sido atendido excesivamente tarde. Esto es normal debido a que ha habido modelos que puntualmente han ofrecido una solución que, aunque factible, era bastante mala con un GAP cercano a la unidad.

En definitiva, el valor de la función objetivo ha tomado el valor del sumatorio del LOS de todos los pacientes del problema en un 93% de los casos, mientras que el primer término que envuelve el cumplimiento del TEPCOF de los pacientes solo ha afectado en el 7% de las soluciones restantes. Este resultado implica que los modelos asignan bien los pacientes que inicialmente tienen que pasar por 1º Consulta por orden de prioridad, ya que casi siempre lo hacen dentro del tiempo establecido, sin embargo, tratar de minimizar la duración de cada paciente dentro del SUH se vuelve una tarea compleja, con una dificultad que asciende bruscamente al aumentar el número de recursos o el nivel de saturación del SUH.

6 CONCLUSIONES

En el presente TFG se han desarrollado modelos de programación lineal utilizando el programa Python con el fin de abordar el problema de la gestión del flujo de pacientes dentro del SUH. Actualmente hay numerosas líneas de investigación que abordan esta temática debido al hecho de que la saturación de los SUH es una realidad que atañe a toda la población, que ha vivido un momento extremo de colapso hace escasos años y a que es un problema al que no se le consigue poner solución.

Este proyecto ha sido enfocado en la creación de modelos de programación lineal como alternativa para tratar de resolver problemas reales de un SUH. Así pues, no solo se han generado un modelo básico con tres variantes aplicables en el ámbito real del SUH, sino que también se han analizado como responden esos modelos programados con dos formas de modelar distintas, con el fin de comprobar cual de estas dos formas de modelar muy comunes en la literatura es más eficiente.

Así pues, se desarrollaron los cuatro modelos para ambos métodos, utilizando una instancia pequeña y un código que permite pintar la solución obtenida en un diagrama de Gantt, de forma que se viera visualmente que las restricciones se cumplían y que el modelo respondía debidamente. Es claro que al tomar la misma instancia en ambos métodos de modelado, los resultados tenían que ser los mismos para la misma variante. Pues bien, ya solo con esta instancia de prueba nos dimos cuenta de que el segundo método, aunque alcanzaba el óptimo, lo hacía en mucho más tiempo que el primero de los métodos, aún cogiendo el valor de p óptimo. No obstante, quedaba iniciar la experimentación para poder desacreditar el segundo de los métodos.

Tal y como se ha explicado, hallar el valor de p óptimo para estos problemas es una tarea compleja que no concierne a este TFG, con lo que se decidió determinar un valor de p admisible y comenzar con la experimentación. Tras los dos primeros casos de estudio, era más que evidente que el segundo método se quedaba atrás con respecto al primero, por lo que se decidió descarta el mismo para el tercer caso de estudio.

Se puede decir que los modelos creados responden correctamente para hospitales pequeños, cuya configuración de recursos no sea grande y que no tenga que atender diariamente a demasiados pacientes. De hecho, en el primer caso de estudio y para niveles de saturación del 25% y 50%, los resultados son en general bastante buenos, con un alto porcentaje de optimalidad y un GAP aceptable para las soluciones factibles. De hecho, para las tres configuraciones de recursos propuestas, si se establece un nivel de saturación del 25%, es decir, si son pocos los pacientes que tienen que ser atendidos, los modelos responden muy bien al problema. Sin embargo, cuando se aumenta la saturación a niveles más altos con demasiados recursos, los modelos empiezan a no encontrar solución al problema, o las soluciones que encuentran están demasiado lejos de la óptima.

En resumen, para una distribución de recursos pequeña e independientemente del nivel de saturación establecido, o bien para una distribución más grande pero con niveles de saturación bajos (pocos pacientes), los modelos que en este TFG se proponen aportan una solución factible buena en un tiempo razonable. No obstante, si la distribución de recursos es grande y hay muchos pacientes que atender, la complejidad del problema aumenta hasta el punto en que los modelos ya no responden de igual forma, dejando problemas sin solución dentro del tiempo establecido o encontrando soluciones que implican un alto incumplimiento del TEPCOF y largas estancias de los pacientes dentro del SUH.

6.1 Líneas de trabajo futuras

Hay hospitales en los que en vez de que los recursos estén en una posición fija y los pacientes se muevan, los pacientes son asignados a un box fijo, y son los recursos los que se desplazan para atender a los pacientes. Esta podría ser una línea de trabajo futura, de modo que los modelos matemáticos de programación lineal diseñados y desarrollados podrían ser adaptados de forma que se consideren otros parámetros como el tiempo de desplazamiento de los recursos de un paciente a otro, entre otros.

Asimismo, se ha visto que los modelos no responden lo bien que se desea para una saturación del 75%. Peor respuesta dará si cabe para una saturación del 100%, o más aún para el 125% (sobresaturación), cabe esperar

que en 10 minutos no encuentren ni siquiera una solución. Para estos casos y a modo de línea de investigación futura, habría que considerar otras metodologías de resolución del campo de algoritmos avanzados de optimización como metaheurísticas.

REFERENCIAS

[1] N. S. y. C. Gutiérrez, «SEMES,» 2023. [En línea]. Available: https://www.semes.org/los-urgenciologos-alertan-los-servicios-de-urgencias-de-todo-el-pais-estan-saturados-con-un-incremento-de-la-carga-asistencial-que-llega-al-40-en-algunas-comunidades/.

[2] M. d. S. y. P. Social, Unidad de Ugrencia Hospitalaria: Estándares y recomendaciones, 2010.

[3] F. J. &. B.-P. G. Martín-Sánchez, Los servicios de urgencias hospitalarios en España: realidad y desafíos, 2018.

[4] M. d. sanidad, Informe Anual del Sistema Nacional de Salud, 2020-2021.

[5] M. B. Orduna, «Saturación en los Servicios de Urgencias hospitalarios: análisis causal y búsqueda de soluciones,» 2015-2016.

[6] L. B.-V. y. E. Kirac, «Evaluating alternative resource allocation in an emergency department using discrete event simulation,» 2016.

[7] C. Teresa Morales y F. J. Garcia Berrocal, Hôtel-Dieu de París; orígenes y aparición de las primeras enfermeras religiosas de la historia, 2013.

[8] A. P. Álvarez-Arenas, Los hospitales de Franco, 2003.

[9] J.-H. L. y. Y.-H. T. Kuan-Chia Lin, Emergency Department Overcrowding: A Global Perspective, Revista Academia Emergency Medicine, 2001.

[10] R. F. M. M. &. F. R. Colom, Planificación, gestión y dirección de los servicios de urgencias y emergencias sanitarias, Elsevier España.

[11] E. S. D. M. C. y. G. M. P. Navarro, Gestión clínica y calidad asistencial en los servicios de urgencias hospitalarios, Index de Enfermería, 2013.

[12] A. C.-B. M. y. N.-V. J. R. Zapata-López, Planificación y organización de los servicios de urgencias hospitalarios: guía práctica, 2017.

[13] M. y. L.-F. L. A. Córdoba, Organización y gestión de los servicios de urgencias hospitalarios, Revista de la Sociedad Española de Enfermería de Urgencias y Emergencias, 2013.

[14] D. G. Medina, Algoritmos de Optimización para el servicio de urgencias: Caso de estudio en el hospital Virgen del Rocío, Sevilla, 2021.

[15] S. A. d. Salud, Plan de Mejora de los Servicios de Urgencias de Hospital del Sistema Sanitario Público de Andalucía, 2019.

[16] O. P. d. l. Salud, «Organización Panamericana de la Salud,» [En línea]. Available: https://www.paho.org/es/emergencias-salud/equipos-medicos-

emergencia#:~:text=Los%20Equipos%20M%C3%A9dicos%20de%20Emergencia%20(EMT%20por%20sus%20siglas%20en,los%20sistemas%20locales%20de%20salud..

[17] I. VIANA, «ABC Historia,» 19 04 2020. [En línea]. Available: https://www.abc.es/historia/abci-funcionaba-edad-media-hospital-mas-antiguo-mundo-curando-desde-hace-1400-anos-202004170103_noticia.html.

[18] J. Erich, «EMSWORLD,» 30 01 2020. [En línea]. Available: https://www.hmpgloballearningnetwork.com/site/emsworld/article/1223826/el-salon-de-la-fama-clawson-cowley.

[19] J. E. Tintinalli, Emergency Medicine: A Comprehensive Study Guide, McGraw-Hill Education, 2016.

[20] A. P.-S. Á. R.-P. J. D. N.-H. F. y. Ú.-B. I. Martínez-Rubio, Gestión de los servicios de urgencias hospitalarios: modelo y organización, 2018.

[21] J. N. J. A. M. A. G. M. B. M. C. Molina-Pariente, Soporte a la toma de decisiones en la gestión de pacientes COVID en un servicio de urgencia hospitalaria. En B. Puebla-Martínez (Ed.). Ecosistema de una pandemia. COVID-19, la transformación mundial (pp. 27-51), Dykinson, S.L., 2021.

[22] O. P. d. l. Salud, «Organización Panamericana de la Salud,» [En línea]. Available: https://www.paho.org/es/emergencias-salud/equipos-medicos-emergencia#:~:text=Los%20Equipos%20M%C3%A9dicos%20de%20Emergencia%20(EMT%20por%20sus%20siglas%20en,los%20sistemas%20locales%20de%20salud..

ANEXOS

modelos.py

```
import random
import numpy as np
import mip
from mip import *
from gantt import *
import time
#Modelo básico con modelaje gamma
def
modelo1_gamma(I,pu,pp,first,last,r,A,pt,s,m,am,maxA,J,t,d,M,tepcof,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo1_gamma",sense=MINIMIZE,solver_name=CBC)
    X_aj=[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for j
in range(J)]for a in range(A)]
    GAMMA_aa1j=[[[Modelo.add_var(name="Gamma_"+str(a+1)+str(a1+1)+str(j+1),
var_type=BINARY) for j in range(J) if a1>a]for a1 in range(A)] for a in range
(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if s[a]==t[j])==1),
"R1_"+str(a+1)

    #Restriccion 2
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range (J) if
s[a]!=t[j])==0),"R2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
               Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*X_aj[first[i]-1][j], "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(X_aj[pu[i][a]-1][j]+X_aj[pu[i][a-1]-1][j1]-2),
"R4_"+str(i+1)+str(a+1)

    #Restriccion 5: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a<a1:
                for j in range (J):
```

```
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a1][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a][j]+pt[a1]-M*(1-GAMMA_aa1j[a][a1][j]),"R5_"+str(j+1)

    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a<a1:
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a1][j]+pt[a]-M*(GAMMA_aa1j[a][a1][j]),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]>=(pt[a]+d[j][0])*X_aj[a][j],
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*X_aj[a][j],
"R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCO
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))

    Modelo.write("Modelo1_gamma.lp")

    Modelo.optimize(max_seconds=600)

    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for a in range (A):
        for j in range (J):
            if (X_aj[a][j].x>0):
                print("La actividad "+str(a+1)+" se desarrolla en el recurso
"+str(j+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
```

```
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)

#Modelo Básico con modelaje posiciones
def modelo1_posiciones(I,pu,pp,first,last,r,A,pt,s,m,
am,maxA,J,t,d,M,tepcof,P,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo1_posiciones",sense=MINIMIZE,solver_name=CBC)
    X_ajp=[[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for
p in range (P)] for j in range(J)]for a in range(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
s[a]==t[j])==1), "R1_"+str(a+1)

    #Restriccion 2
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range (J) if
s[a]!=t[j])==0),"R2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
                Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*xsum(X_ajp[first[i]-1][j][p] for p in range (P)), "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(xsum(X_ajp[pu[i][a]-1][j][p]for p in range
(P))+xsum(X_ajp[pu[i][a-1]-1][j1][p] for p in range (P))-2),
"R4_"+str(i+1)+str(a+1)
    #Restricción 5: en un recurso una posicion solo puede estar asignada a
una actividad
    for j in range(J):
        for p in range (P):
```

```
            Modelo+= (xsum(X_ajp[a][j][p] for a in range
(A))<=1),"R5_"+str(j+1)
    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a!=a1:
                for j in range (J):
                    for p in range (1,P):
                        if(t[j]==s[a1] and t[j]==s[a]):
                            Modelo+=CT_aj[a][j]>=CT_aj[a1][j]+pt[a]-M*(2-
X_ajp[a][j][p]-xsum(X_ajp[a1][j][p1] for p1 in range (p)),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo +=
CT_aj[a][j]>=(pt[a]+d[j][0])*(xsum(X_ajp[a][j][p]for p in range (P))),
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*(xsum(X_ajp[a][j][p] for p in
range (P))), "R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCO
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))


    Modelo.write("Modelo1_posiciones.lp")

    Modelo.optimize(max_seconds=600)


    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for j in range (J):
        for p in range (P):
            for a in range (A):
                if (X_ajp[a][j][p].x>0):
                    print("La actividad "+str(a+1)+" se desarrolla en el
recurso "+str(j+1)+"en la poscion "+str(p+1)+" y termina en ",CT_aj[a][j].x)
```

```
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)

#Modelo prioridades con modelaje gamma
def modelo2_gamma(I,pu,pp,first,last,r,A,pt,pa,s,m,
am,maxA,J,t,d,pr,n,M,tepcof,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo1_gamma",sense=MINIMIZE,solver_name=CBC)
    X_aj=[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for j
in range(J)]for a in range(A)]
    GAMMA_aa1j=[[[Modelo.add_var(name="Gamma_"+str(a+1)+str(a1+1)+str(j+1),
var_type=BINARY) for j in range(J) if a1>a]for a1 in range(A)] for a in range
(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if s[a]==t[j])==n[pa[a]-
1][m[a]-1]), "R1_"+str(a+1)

    #Restricción 2.1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if (s[a]==t[j] and
pr[pa[a]-1][j]==0))==0),"R2_1_"+str(a+1)

    #Restriccion 2.2
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range (J) if
s[a]!=t[j])==0),"R2_2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
               Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*X_aj[first[i]-1][j], "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
```

```
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(X_aj[pu[i][a]-1][j]+X_aj[pu[i][a-1]-1][j1]-2),
"R4_"+str(i+1)+str(a+1)

    #Restriccion 5: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a<a1:
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a1][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a][j]+pt[a1]-M*(1-GAMMA_aa1j[a][a1][j]),"R5_"+str(j+1)

    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A): #(1,P)
        for a1 in range(A):
            if a<a1: #a != a1
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a1][j]+pt[a]-M*(GAMMA_aa1j[a][a1][j]),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]>=(pt[a]+d[j][0])*X_aj[a][j],
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*X_aj[a][j],
"R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCOF
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))


    Modelo.write("Modelo1_gamma.lp")

    Modelo.optimize(max_seconds=600)

    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))
```

```
    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for a in range (A):
        for j in range (J):
            if (X_aj[a][j].x>0):
                print("La actividad "+str(a+1)+" se desarrolla en el recurso
"+str(j+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)
#Modelo prioridades con modelaje posiciones
def modelo2_posiciones(I,pu,pp,first,last,r,A,pt,pa,s,m,
am,maxA,J,t,d,pr,n,M,tepcof,P,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo1_posiciones",sense=MINIMIZE,solver_name=CBC)
    X_ajp=[[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for
p in range (P)] for j in range(J)]for a in range(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
s[a]==t[j])==n[pa[a]-1][m[a]-1]), "R1_"+str(a+1)
    #Restricción 2.1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
(s[a]==t[j] and pr[pa[a]-1][j]==0))==0),"R2_1_"+str(a+1)
    #Restriccion 2.2
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range (J) if
s[a]!=t[j])==0),"R2_2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
                Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*xsum(X_ajp[first[i]-1][j][p] for p in range (P)), "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
```

```
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-
1][j1]+pt[pu[i][a]-1]+M*(xsum(X_ajp[pu[i][a]-1][j][p]for p in range
(P))+xsum(X_ajp[pu[i][a-1]-1][j1][p] for p in range (P))-2),
"R4_"+str(i+1)+str(a+1)
    #Restricción 5: en un recurso una posicion solo puede estar asignada a
una actividad
    for j in range(J):
        for p in range (P):
            Modelo+= (xsum(X_ajp[a][j][p] for a in range
(A))<=1),"R5_"+str(j+1)
    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a!=a1:
                for j in range (J):
                    for p in range (1,P):
                        if(t[j]==s[a1] and t[j]==s[a]):
                            Modelo+=CT_aj[a][j]>=CT_aj[a1][j]+pt[a]-M*(2-
X_ajp[a][j][p]-xsum(X_ajp[a1][j][p1] for p1 in range (p))),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo +=
CT_aj[a][j]>=(pt[a]+d[j][0])*(xsum(X_ajp[a][j][p]for p in range (P))),
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*(xsum(X_ajp[a][j][p] for p in
range (P))), "R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCO
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))


    Modelo.write("Modelo1_posiciones.lp")

    Modelo.optimize(max_seconds=600)
```

```
    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for j in range (J):
        for p in range (P):
            for a in range (A):
                if (X_ajp[a][j][p].x>0):
                    print("La actividad "+str(a+1)+" se desarrolla en el
recurso "+str(j+1)+"en la poscion "+str(p+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)

#Modelo prioridades y rec-act con modelaje gamma
def modelo3_gamma(I,pu,pp,first,last,r,A,pt,pa,s,m,
am,maxA,J,t,d,pr,b,n,M,tepcof,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo2_gamma",sense=MINIMIZE,solver_name=CBC)
    X_aj=[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for j
in range(J)]for a in range(A)]
    GAMMA_aa1j=[[[Modelo.add_var(name="Gamma_"+str(a+1)+str(a1+1)+str(j+1),
var_type=BINARY) for j in range(J) if a1>a]for a1 in range(A)] for a in range
(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if s[a]==t[j]
                      and b[m[a]-1][j]==1)==n[pa[a]-1][m[a]-1]),
"R1_"+str(a+1)

    #Restricción 2.1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if (s[a]==t[j] and
pr[pa[a]-1][j]==0))==0),"R2_1_"+str(a+1)

    #Restriccion 2.2
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range (J) if
s[a]!=t[j])==0),"R2_2_"+str(a+1)

    #Restriccion 3:
```

```
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
               Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*X_aj[first[i]-1][j], "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(X_aj[pu[i][a]-1][j]+X_aj[pu[i][a-1]-1][j1]-2),
"R4_"+str(i+1)+str(a+1)
    #Restriccion 5: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a<a1:
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a1][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a][j]+pt[a1]-M*(1-GAMMA_aa1j[a][a1][j]),"R5_"+str(j+1)

    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A): #(1,P)
        for a1 in range(A):
            if a<a1: #a != a1
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a1][j]+pt[a]-M*(GAMMA_aa1j[a][a1][j]),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]>=(pt[a]+d[j][0])*X_aj[a][j],
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*X_aj[a][j],
"R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCO
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)
```

```
    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))


    Modelo.write("Modelo2_gamma.lp")

    Modelo.optimize(max_seconds=600)

    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for a in range (A):
        for j in range (J):
            if (X_aj[a][j].x>0):
                print("La actividad "+str(a+1)+" se desarrolla en el recurso
"+str(j+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)

#Modelo prioridades y rec-act con modelaje posiciones
def modelo3_posiciones(I,pu,pp,first,last,r,A,pt,pa,s,m,
am,maxA,J,t,d,pr,b,n,M,tepcof,P,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo2_posiciones",sense=MINIMIZE,solver_name=CBC)
    X_ajp=[[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for
p in range (P)] for j in range(J)]for a in range(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
s[a]==t[j]
                      and b[m[a]-1][j]==1)==n[pa[a]-1][m[a]-1]),
"R1_"+str(a+1)
    #Restricción 2.1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
(s[a]==t[j] and pr[pa[a]-1][j]==0))==0),"R2_1_"+str(a+1)
    #Restriccion 2.2
    for a in range(A):
```

```
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range (J) if
s[a]!=t[j])==0),"R2_2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
                Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*xsum(X_ajp[first[i]-1][j][p] for p in range (P)), "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(xsum(X_ajp[pu[i][a]-1][j][p]for p in range
(P))+xsum(X_ajp[pu[i][a-1]-1][j1][p] for p in range (P))-2),
"R4_"+str(i+1)+str(a+1)
    #Restricción 5: en un recurso una posicion solo puede estar asignada a
una actividad
    for j in range(J):
        for p in range (P):
            Modelo+= (xsum(X_ajp[a][j][p] for a in range
(A))<=1),"R5_"+str(a+1)
    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a!=a1:
                for j in range (J):
                    for p in range (1,P):
                        if(t[j]==s[a1] and t[j]==s[a]):
                            Modelo+=CT_aj[a][j]>=CT_aj[a1][j]+pt[a]-M*(2-
X_ajp[a][j][p]-xsum(X_ajp[a1][j][p1] for p1 in range (p))),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo +=
CT_aj[a][j]>=(pt[a]+d[j][0])*(xsum(X_ajp[a][j][p]for p in range (P))),
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*(xsum(X_ajp[a][j][p] for p in
range (P))), "R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCO
    for i in range (I):
        if (m[first[i]-1]==1):
```

```
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))

    Modelo.write("Modelo2_posiciones.lp")

    Modelo.optimize(max_seconds=600)


    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for j in range (J):
        for p in range (P):
            for a in range (A):
                if (X_ajp[a][j][p].x>0):
                    print("La actividad "+str(a+1)+" se desarrolla en el
recurso "+str(j+1)+"en la poscion "+str(p+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)

#Modelo COVID (final) con modelaje gamma (prioridades y rec-act)
def modelo4_gamma(I,pu,pp,first,last,r,tp,A,pt,pa,s,m,cv,
am,maxA,J,t,d,pr,b,covid,n,M,tepcof,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo3_gamma",sense=MINIMIZE,solver_name=CBC)
    X_aj=[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for j
in range(J)]for a in range(A)]
    GAMMA_aa1j=[[[Modelo.add_var(name="Gamma_"+str(a+1)+str(a1+1)+str(j+1),
var_type=BINARY) for j in range(J) if a1>a]for a1 in range(A)] for a in range
(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if s[a]==t[j]
```

```
                    and b[m[a]-1][j]==1  and covid[cv[a]-
1][j]==1)==n[pa[a]-1][m[a]-1]), "R1_"+str(a+1)

    #Restricción 2.1
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range(J) if (s[a]==t[j] and
pr[pa[a]-1][j]==0))==0),"R2_1_"+str(a+1)

    #Restriccion 2.2
    for a in range(A):
        Modelo+=(xsum(X_aj[a][j]for j in range (J) if
s[a]!=t[j])==0),"R2_2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
               Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*X_aj[first[i]-1][j], "R3_P"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(X_aj[pu[i][a]-1][j]+X_aj[pu[i][a-1]-1][j1]-2),
"R4_"+str(i+1)+str(a+1)

    #Restriccion 5: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a<a1:
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a1][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a][j]+pt[a1]-M*(1-GAMMA_aa1j[a][a1][j]),"R5_"+str(j+1)

    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A): #(1,P)
        for a1 in range(A):
            if a<a1: #a != a1
                for j in range (J):
                    if(t[j]==s[a1] and t[j]==s[a]):
                        Modelo+=CT_aj[a][j]+M*(2-X_aj[a][j]-
X_aj[a1][j])>=CT_aj[a1][j]+pt[a]-M*(GAMMA_aa1j[a][a1][j]),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]>=(pt[a]+d[j][0])*X_aj[a][j],
"R7_"+str(a+1)+str(j+1)
```

```
    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*X_aj[a][j],
"R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCOF
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))

    Modelo.write("Modelo4_gamma.lp")

    Modelo.optimize(max_seconds=600)

    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for a in range (A):
        for j in range (J):
            if (X_aj[a][j].x>0):
                print("La actividad "+str(a+1)+" se desarrolla en el recurso
"+str(j+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)

#Modelo COVID (final) con modelaje posiciones (prioridades y rec-act)
def modelo4_posiciones(I,pu,pp,first,last,r,tp,A,pt,pa,s,m,cv,
am,maxA,J,t,d,pr,b,k,n,M,tepcof,P,r_tepcof):
    ini=time.time()
    Modelo=Model("Modelo3_posiciones",sense=MINIMIZE,solver_name=CBC)
    X_ajp=[[[Modelo.add_var(name="X_"+str(a+1)+str(j+1), var_type=BINARY) for
p in range (P)] for j in range(J)]for a in range(A)]
    CT_aj=[[Modelo.add_var(name="CT_"+str(a+1)+str(j+1), var_type=INTEGER,
lb=0, ub=INF) for j in range(J)]for a in range(A)]
```

```
    T_i=[Modelo.add_var(name="T_"+str(i+1), var_type=INTEGER, lb=0, ub=INF)
for i in range (I)]

    #Restriccion 1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
s[a]==t[j]
                      and b[m[a]-1][j]==1  and k[cv[a]-1][j]==1)==n[pa[a]-
1][m[a]-1]), "R1"+str(a+1)
    #Restricción 2.1
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range(J) if
(s[a]==t[j] and pr[pa[a]-1][j]==0))==0),"R2_1_"+str(a+1)

    #Restriccion 2.2
    for a in range(A):
        Modelo+=(xsum(X_ajp[a][j][p] for p in range (P) for j in range (J) if
s[a]!=t[j])==0),"R2_2_"+str(a+1)

    #Restriccion 3:
    for i in range(I):
        for j in range (J):
            if s[first[i]-1]==t[j]:
                Modelo += CT_aj[first[i]-1][j]>=(r[i]+pt[first[i]-
1])*xsum(X_ajp[first[i]-1][j][p] for p in range (P)), "R3_"+str(i+1)+str(j+1)

    #Restriccion 4: para un paciente i, que la actividad a no empiece hasta
que la a-1 no haya temrinado
    for i in range (I):
        for a in range(1,maxA):
            if (pu[i][a]!=0): #JM
                for j in range(J):
                    for j1 in range(J):
                        if(t[j]==s[pu[i][a]-1] and t[j1]==s[pu[i][a-1]-1]):
                            Modelo += CT_aj[pu[i][a]-1][j]>=CT_aj[pu[i][a-1]-
1][j1]+pt[pu[i][a]-1]+M*(xsum(X_ajp[pu[i][a]-1][j][p]for p in range
(P))+xsum(X_ajp[pu[i][a-1]-1][j1][p] for p in range (P))-2),
"R4_"+str(i+1)+str(a+1)
    #Restricción 5: en un recurso una posicion solo puede estar asignada a
una actividad
    for j in range(J):
        for p in range (P):
            Modelo+= (xsum(X_ajp[a][j][p] for a in range
(A))<=1),"R5_"+str(j+1)
    #Restriccion 6: actividades en un mismo recurso no se solapen
    for a in range(A):
        for a1 in range(A):
            if a!=a1:
                for j in range (J):
                    for p in range (1,P):
                        if(t[j]==s[a1] and t[j]==s[a]):
                            Modelo+=CT_aj[a][j]>=CT_aj[a1][j]+pt[a]-M*(2-
X_ajp[a][j][p]-xsum(X_ajp[a1][j][p1] for p1 in range (p))),"R6_"+str(j+1)

    #Restricción 7: disponibilidad de los recursos (comienzo)
    for a in range (A):
        for j in range (J):
```

```
            if s[a]==t[j]:
                Modelo +=
CT_aj[a][j]>=(pt[a]+d[j][0])*(xsum(X_ajp[a][j][p]for p in range (P))),
"R7_"+str(a+1)+str(j+1)

    #Restricción 8: disponibilidad de los recursos (final)
    for a in range(A):
        for j in range(J):
            if s[a]==t[j]:
                Modelo += CT_aj[a][j]<=d[j][1]*(xsum(X_ajp[a][j][p] for p in
range (P))), "R8_"+str(a+1)+str(j+1)

    #Restricción 9: variable auxiliar para minimizar los pacientes que NO
cumplen TEPCO
    for i in range (I):
        if (m[first[i]-1]==1):
            for j in range (J):
                if(t[j]==s[first[i]-1]):
                    Modelo += T_i[i]>=CT_aj[first[i]-1][j]-pt[first[i]-1]-
r_tepcof[i]-tepcof[pp[i]-1], "R9_"+str(i+1)+str(j+1)

    Modelo.objective = (xsum(T_i[i] for i in range
(I))+xsum(xsum(CT_aj[last[i]-1][j] for j in range (J)) for i in range(I)))

    Modelo.write("Modelo3_posiciones.lp")

    Modelo.optimize(max_seconds=600)

    print("El valor de la funcion objetivo es: "
+str(Modelo.objective_value))

    print("El status de la solucion es: " + str(Modelo.status))

    """print("Los procesos planificados son: ")
    for j in range (J):
        for p in range (P):
            for a in range (A):
                if (X_ajp[a][j][p].x>0):
                    print("La actividad "+str(a+1)+" se desarrolla en el
recurso "+str(j+1)+"en la poscion "+str(p+1)+" y termina en ",CT_aj[a][j].x)
    """
    fin=time.time()
    if(Modelo.status==OptimizationStatus.FEASIBLE or
Modelo.status==OptimizationStatus.OPTIMAL):
        tepcof_fo=xsum(T_i[i].x for i in range (I))
        LOS_fo=xsum(xsum(CT_aj[last[i]-1][j].x for j in range (J)) for i in
range(I))
    else:
        tepcof_fo=0
        LOS_fo=0

    return(Modelo.objective_value,Modelo.status,fin-
ini,Modelo.gap,600,Modelo.num_cols,Modelo.num_rows,I,A,J,tepcof_fo,LOS_fo)
```

instancias.py

```
import numpy as np
import mip
from mip import *
import random
from random import choice
from random import triangular
from gantt import *

def Generar_Instancia(sat):
    prioridad=[1,2,3,4,5]
    r_t=[0,14,59,99,119] #TEPCOF-1 porque en el momento foto no sabemos
cuanto puede llevar esperando ese paciente
    tiempo=100
    saturacion=[0 for i in range(5)]
    maxA=8
    I=0
    i=0
    pu = np.empty((0,maxA), int)
    pt=[]
    pp=[]
    first=[]
    last=[]
    tp=[]
    r=[]
    r_tepcof=[]
    pa=[]
    s=[]
    m=[]
    cv=[]
    pac_activo=[]
    flag=1
    LOS_medio=((3.15+3.14+2.86+1.49+1.31)*60)/5
    """num_recursos=[4,8,2,2,1]"""
    num_recursos=[1,6,1,1,1]
    #num_recursos=[4,8,1,1,1]
    #num_recursos=[4,8,2,2,2]
    while(flag==1):
        I=I+1
        prior=random.choices(prioridad,weights=[0.3,10.8,48.4,36.7,1.5])
        pp.append(prior[0])
        r.append(0)
        if(prior==[1]):
            r_tepcof.append(0)
        else:
            r_tepcof.append(np.random.randint(0,r_t[prior[0]-1]))
        #Para la variante del COVID, vamos a suponer que un 20% son COVID y
un 80% no lo son
        covid=random.choices([1,2],weights=[20,80])
        if (prior==[1]):
            #Puede haber CT-scan con una probabilidad del 70%
            ct_scan=random.choices([1,2],weights=[70,30])
            if(ct_scan==[1]):
                inst_act=random.choices([1,2,3,4,5,6,7])
                if(inst_act==[1]):
```

```
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,i+8]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(10,20,25)))
                    pt.append(round(triangular(20,30,75)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(20)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+5]+10
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+6]
                    saturacion[2]=saturacion[2]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i+3]
                    saturacion[4]=saturacion[4]+20
                    i=i+8
                    last.append(i)
                    for j in range (8):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(8):
                            cv.append(1)
                    else:
                        for j in range(8):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(20,30,75)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(20)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+4]+10
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+5]
```

```
                    saturacion[2]=saturacion[2]+pt[i+1]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    saturacion[4]=saturacion[4]+20
                    i=i+7
                    last.append(i)
                    for j in range (7):
                        pa.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(7):
                            cv.append(1)
                    else:
                        for j in range(7):
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(20)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+3]+10
                    saturacion[1]=saturacion[1]+pt[i+4]
                    saturacion[2]=saturacion[2]+pt[i]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    saturacion[4]=saturacion[4]+20
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(1)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(3)
                    m.append(4)
                    m.append(5)
```

```
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(20)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+2]+10
                    saturacion[1]=saturacion[1]+pt[i+3]
                    saturacion[3]=saturacion[3]+pt[i]
                    saturacion[4]=saturacion[4]+20
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(1)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(20)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+1]+10
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[4]=saturacion[4]+20
                    i=i+4
```

```
                        last.append(i)
                        for j in range (4):
                            pa.append(1)
                        s.append(5)
                        s.append(1)
                        s.append(2)
                        s.append(1)
                        m.append(5)
                        m.append(6)
                        m.append(7)
                        m.append(8)
                        if(covid==[1]):
                            for j in range(4):
                                cv.append(1)
                        else:
                            for j in range(4):
                                cv.append(2)
                    if(inst_act==[6]):
                        pac_activo.append(I)
                        pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                        first.append(i+1)
                        pt.append(round(triangular(20,30,40)))
                        pt.append(round(triangular(20,30,40)))
                        pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                        saturacion[0]=saturacion[0]+pt[i]+10
                        saturacion[1]=saturacion[1]+pt[i+1]
                        i=i+3
                        last.append(i)
                        for j in range (3):
                            pa.append(1)
                        s.append(1)
                        s.append(2)
                        s.append(1)
                        m.append(6)
                        m.append(7)
                        m.append(8)
                        if(covid==[1]):
                            for j in range(3):
                                cv.append(1)
                        else:
                            for j in range(3):
                                cv.append(2)
                    if(inst_act==[7]):
                        pac_activo.append(I)
                        pu = np.append(pu,
np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                        first.append(i+1)
                        pt.append(round(triangular(20,30,40)))
                        pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                        saturacion[0]=saturacion[0]+10
                        saturacion[1]=saturacion[1]+pt[i]
                        i=i+2
                        last.append(i)
                        for j in range (2):
```

```
                        pa.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
                            cv.append(2)
            else:
                inst_act=random.choices([1,2,3,4,5,6])
                if(inst_act==[1]):
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(10,20,25)))
                    pt.append(round(triangular(20,30,75)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+4]+10
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+5]
                    saturacion[2]=saturacion[2]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i+3]
                    i=i+7
                    last.append(i)
                    for j in range (7):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(7):
                            cv.append(1)
                    else:
                        for j in range(7):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
```

```
                    pt.append(round(triangular(20,30,75)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+3]+10
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+4]
                    saturacion[2]=saturacion[2]+pt[i+1]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+2]+10
                    saturacion[1]=saturacion[1]+pt[i+3]
                    saturacion[2]=saturacion[2]+pt[i]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(1)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
```

```
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i+1]+10
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i]
                    i=i+4
                    last.append(i)
                    for j in range (4):
                        pa.append(1)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(4):
                            cv.append(1)
                    else:
                        for j in range(4):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+pt[i]+10
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(1)
                    s.append(1)
```

```
                    s.append(2)
                    s.append(1)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
                            cv.append(1)
                    else:
                        for j in range(3):
                            cv.append(2)
                if(inst_act==[6]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(20,30,40)))
                    pt.append(10) #las altas vamos a poner que duran 10min en
prior 1
                    saturacion[0]=saturacion[0]+10
                    saturacion[1]=saturacion[1]+pt[i]
                    i=i+2
                    last.append(i)
                    for j in range (2):
                        pa.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
                            cv.append(2)
        if(prior==[2]):
            #Puede haber CT-scan con una probabilidad del 70%
            ct_scan=random.choices([1,2],weights=[70,30])
            if(ct_scan==[1]):
                inst_act=random.choices([1,2,3,4,5,6,7])
                if(inst_act==[1]):
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,i+8]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(10,20,25)))
                    pt.append(round(triangular(18,28,45)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+5]+8
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+6]
                    saturacion[2]=saturacion[2]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i+3]
```

```
                    saturacion[4]=saturacion[4]+15
                    i=i+8
                    last.append(i)
                    for j in range (8):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(8):
                            cv.append(1)
                    else:
                        for j in range(8):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(18,28,45)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+4]+8
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+5]
                    saturacion[2]=saturacion[2]+pt[i+1]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    saturacion[4]=saturacion[4]+15
                    i=i+7
                    last.append(i)
                    for j in range (7):
                        pa.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(3)
```

```
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(7):
                            cv.append(1)
                    else:
                        for j in range(7):
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+3]+8
                    saturacion[1]=saturacion[1]+pt[i+4]
                    saturacion[2]=saturacion[2]+pt[i]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    saturacion[4]=saturacion[4]+15
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(1)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(3)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
```

```
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+2]+8
                    saturacion[1]=saturacion[1]+pt[i+3]
                    saturacion[3]=saturacion[3]+pt[i]
                    saturacion[4]=saturacion[4]+15
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(1)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(15)
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+1]+8
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[4]=saturacion[4]+15
                    i=i+4
                    last.append(i)
                    for j in range (4):
                        pa.append(1)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(4):
                            cv.append(1)
                    else:
                        for j in range(4):
```

```
                            cv.append(2)
                if(inst_act==[6]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i]+8
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
                            cv.append(1)
                    else:
                        for j in range(3):
                            cv.append(2)
                if(inst_act==[7]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+8
                    saturacion[1]=saturacion[1]+pt[i]
                    i=i+2
                    last.append(i)
                    for j in range (2):
                        pa.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
                            cv.append(2)
            else:
                inst_act=random.choices([1,2,3,4,5,6])
                if(inst_act==[1]):
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,0]]),axis=0)
```

```
                    first.append(i+1)
                    pt.append(round(triangular(10,20,25)))
                    pt.append(round(triangular(18,28,45)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+4]+8
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+5]
                    saturacion[2]=saturacion[2]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i+3]
                    i=i+7
                    last.append(i)
                    for j in range (7):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(7):
                            cv.append(1)
                    else:
                        for j in range(7):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(18,28,45)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+3]+8
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+4]
                    saturacion[2]=saturacion[2]+pt[i+1]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(1)
                    s.append(2)
```

```
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+2]+8
                    saturacion[1]=saturacion[1]+pt[i+3]
                    saturacion[2]=saturacion[2]+pt[i]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(1)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0]]),axis=0)
                    first.append(i+1)
```

```
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i+1]+8
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i]
                    i=i+4
                    last.append(i)
                    for j in range (4):
                        pa.append(1)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(4):
                            cv.append(1)
                    else:
                        for j in range(4):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(15,20,25)))
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+pt[i]+8
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
                            cv.append(1)
                    else:
                        for j in range(3):
                            cv.append(2)
                if(inst_act==[6]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
```

```
                    pt.append(round(triangular(15,20,30)))
                    pt.append(8) #las altas vamos a poner que duran 8min en
prior 2
                    saturacion[0]=saturacion[0]+8
                    saturacion[1]=saturacion[1]+pt[i]
                    i=i+2
                    last.append(i)
                    for j in range (2):
                        pa.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
                            cv.append(2)
        if(prior==[3]):
            #Puede haber CT-scan con una probabilidad del 30%
            ct_scan=random.choices([1,2],weights=[30,70])
            if(ct_scan==[1]):
                inst_act=random.choices([1,2,3,4,5,6,7])
                if(inst_act==[1]):
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,i+8]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,15,20)))
                    pt.append(round(triangular(15,22,30)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+5]+7
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+6]
                    saturacion[2]=saturacion[2]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i+3]
                    saturacion[4]=saturacion[4]+15
                    i=i+8
                    last.append(i)
                    for j in range (8):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(3)
```

```
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(8):
                            cv.append(1)
                    else:
                        for j in range(8):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(15,22,30)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+4]+7
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+5]
                    saturacion[2]=saturacion[2]+pt[i+1]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    saturacion[4]=saturacion[4]+15
                    i=i+7
                    last.append(i)
                    for j in range (7):
                        pa.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(7):
                            cv.append(1)
                    else:
                        for j in range(7):
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
```

```
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+3]+7
                    saturacion[1]=saturacion[1]+pt[i+4]
                    saturacion[2]=saturacion[2]+pt[i]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    saturacion[4]=saturacion[4]+15
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(1)
                    s.append(3)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(3)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,10,20)))
                    pt.append(15)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+2]+7
                    saturacion[1]=saturacion[1]+pt[i+3]
                    saturacion[3]=saturacion[3]+pt[i]
                    saturacion[4]=saturacion[4]+15
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(1)
                    s.append(4)
                    s.append(5)
                    s.append(1)
                    s.append(2)
```

```
                    s.append(1)
                    m.append(4)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(15)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+1]+7
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[4]=saturacion[4]+15
                    i=i+4
                    last.append(i)
                    for j in range (4):
                        pa.append(1)
                    s.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(5)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(4):
                            cv.append(1)
                    else:
                        for j in range(4):
                            cv.append(2)
                if(inst_act==[6]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i]+7
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(1)
```

```
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
                            cv.append(1)
                    else:
                        for j in range(3):
                            cv.append(2)
                if(inst_act==[7]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+7
                    saturacion[1]=saturacion[1]+pt[i]
                    i=i+2
                    last.append(i)
                    for j in range (2):
                        pa.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
                            cv.append(2)
            else:
                inst_act=random.choices([1,2,3,4,5,6])
                if(inst_act==[1]):
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,i+7,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,15,20)))
                    pt.append(round(triangular(15,22,30)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+4]+7
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+5]
                    saturacion[2]=saturacion[2]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i+3]
                    i=i+7
                    last.append(i)
                    for j in range (7):
```

```
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(7):
                            cv.append(1)
                    else:
                        for j in range(7):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(15,22,30)))
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+3]+7
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+4]
                    saturacion[2]=saturacion[2]+pt[i+1]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(1)
                    s.append(2)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
```

```
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(3,5,15)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+2]+7
                    saturacion[1]=saturacion[1]+pt[i+3]
                    saturacion[2]=saturacion[2]+pt[i]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(1)
                    s.append(3)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(3)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7) #las altas vamos a poner que duran 7min en
prior 3
                    saturacion[0]=saturacion[0]+pt[i+1]+7
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i]
                    i=i+4
                    last.append(i)
                    for j in range (4):
                        pa.append(1)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
```

```
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(4):
                            cv.append(1)
                    else:
                        for j in range(4):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(8,15,18)))
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7)
                    saturacion[0]=saturacion[0]+pt[i]+7
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(1)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
                            cv.append(1)
                    else:
                        for j in range(3):
                            cv.append(2)
                if(inst_act==[6]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(10,15,20)))
                    pt.append(7)
                    saturacion[0]=saturacion[0]+7
                    saturacion[1]=saturacion[1]+pt[i]
                    i=i+2
                    last.append(i)
                    for j in range (2):
                        pa.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
```

```
                                cv.append(2)
            if(prior==[4]):
                inst_act=random.choices([1,2,3,4,5])
                if(inst_act==[1]):
                    pu = np.append(pu,
    np.array([[i+1,i+2,i+3,i+4,i+5,i+6,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(2,3,5)))
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(2,5,10)))
                    pt.append(round(triangular(5,10,15)))
                    pt.append(round(triangular(5,8,12)))
                    pt.append(5)
                    saturacion[0]=saturacion[0]+pt[i]+pt[i+3]+5
                    saturacion[1]=saturacion[1]+pt[i+1]+pt[i+4]
                    saturacion[3]=saturacion[3]+pt[i+2]
                    i=i+6
                    last.append(i)
                    for j in range (6):
                        pa.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(6):
                            cv.append(1)
                    else:
                        for j in range(6):
                            cv.append(2)
                if(inst_act==[2]):
                    pac_activo.append(I)
                    pu = np.append(pu,
    np.array([[i+1,i+2,i+3,i+4,i+5,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,10,20)))
                    pt.append(round(triangular(2,5,10)))
                    pt.append(round(triangular(5,10,15)))
                    pt.append(round(triangular(5,8,12)))
                    pt.append(5)
                    saturacion[0]=saturacion[0]+pt[i+2]+5
                    saturacion[1]=saturacion[1]+pt[i]+pt[i+3]
                    saturacion[3]=saturacion[3]+pt[i+1]
                    i=i+5
                    last.append(i)
                    for j in range (5):
                        pa.append(4)
                    s.append(2)
                    s.append(4)
```

```
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(2)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(5):
                            cv.append(1)
                    else:
                        for j in range(5):
                            cv.append(2)
                if(inst_act==[3]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,i+4,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(2,5,10)))
                    pt.append(round(triangular(5,10,15)))
                    pt.append(round(triangular(5,8,12)))
                    pt.append(5)
                    saturacion[0]=saturacion[0]+pt[i+1]+5
                    saturacion[1]=saturacion[1]+pt[i+2]
                    saturacion[3]=saturacion[3]+pt[i]
                    i=i+4
                    last.append(i)
                    for j in range (4):
                        pa.append(4)
                    s.append(4)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(4)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(4):
                            cv.append(1)
                    else:
                        for j in range(4):
                            cv.append(2)
                if(inst_act==[4]):
                    pac_activo.append(I)
                    pu = np.append(pu,
np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,10,15)))
                    pt.append(round(triangular(5,8,12)))
                    pt.append(5)
                    saturacion[0]=saturacion[0]+pt[i]+5
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(4)
```

```
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(6)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
                            cv.append(1)
                    else:
                        for j in range(3):
                            cv.append(2)
                if(inst_act==[5]):
                    pac_activo.append(I)
                    pu = np.append(pu, np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(5,8,12)))
                    pt.append(5)
                    saturacion[0]=saturacion[0]+5
                    saturacion[1]=saturacion[1]+pt[i]
                    i=i+2
                    last.append(i)
                    for j in range (2):
                        pa.append(4)
                    s.append(2)
                    s.append(1)
                    m.append(7)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(2):
                            cv.append(1)
                    else:
                        for j in range(2):
                            cv.append(2)
            if(prior==[5]):
                inst_act=random.choices([1,2])
                if(inst_act==[1]):
                    pu = np.append(pu,
    np.array([[i+1,i+2,i+3,0,0,0,0,0]]),axis=0)
                    first.append(i+1)
                    pt.append(round(triangular(2,3,5)))
                    pt.append(round(triangular(2,4,8)))
                    pt.append(3)
                    saturacion[0]=saturacion[0]+pt[i]+3
                    saturacion[1]=saturacion[1]+pt[i+1]
                    i=i+3
                    last.append(i)
                    for j in range (3):
                        pa.append(5)
                    s.append(1)
                    s.append(2)
                    s.append(1)
                    m.append(1)
                    m.append(2)
                    m.append(8)
                    if(covid==[1]):
                        for j in range(3):
```

```
                        cv.append(1)
                else:
                    for j in range(3):
                        cv.append(2)
            if(inst_act==[2]):
                pac_activo.append(I)
                pu = np.append(pu, np.array([[i+1,i+2,0,0,0,0,0,0]]),axis=0)
                first.append(i+1)
                pt.append(round(triangular(2,4,8)))
                pt.append(3)
                saturacion[0]=saturacion[0]+3
                saturacion[1]=saturacion[1]+pt[i]
                i=i+2
                last.append(i)
                for j in range (2):
                    pa.append(5)
                s.append(2)
                s.append(1)
                m.append(2)
                m.append(8)
                if(covid==[1]):
                    for j in range(2):
                        cv.append(1)
                else:
                    for j in range(3):
                        cv.append(2)
        for k in range(5):
            if(saturacion[k]>=LOS_medio*num_recursos[k]*sat):
                flag=0
    A=len(pt)
    #Esta d es para [4,8,2,2,1], ni siquiera sirve para el tercer caso

"""d=[[np.random.randint(0,20),960],[np.random.randint(0,20),960],[np.random.
randint(0,4),960],

[np.random.randint(0,20),960],[np.random.randint(0,36),960],[np.random.randin
t(0,36),960],

[np.random.randint(0,36),960],[np.random.randint(0,23),960],[np.random.randin
t(0,23),960],

[np.random.randint(0,11),960],[np.random.randint(0,36),960],[np.random.randin
t(0,36),960],

[np.random.randint(0,7),960],[np.random.randint(0,7),960],[np.random.randint(
0,7),960],
       [np.random.randint(0,7),960],[np.random.randint(0,20),960]]"""
    #Esta d es para [1,6,1,1,1]

d=[[np.random.randint(0,20),960],[np.random.randint(0,36),960],[np.random.ran
dint(0,36),960],

[np.random.randint(0,36),960],[np.random.randint(0,23),960],[np.random.randin
t(0,23),960],

[np.random.randint(0,11),960],[np.random.randint(0,7),960],[np.random.randint
(0,7),960],
       [np.random.randint(0,20),960]]
```

```
"""d=[[np.random.randint(0,20),960],[np.random.randint(0,20),960],[np.random.
randint(0,4),960],

[np.random.randint(0,20),960],[np.random.randint(0,36),960],[np.random.randin
t(0,36),960],

[np.random.randint(0,36),960],[np.random.randint(0,23),960],[np.random.randin
t(0,23),960],

[np.random.randint(0,11),960],[np.random.randint(0,36),960],[np.random.randin
t(0,36),960],

[np.random.randint(0,7),960],[np.random.randint(0,7),960],[np.random.randint(
0,20),960]]"""
    #Esta d es para [4,8,2,2,2]

"""d=[[np.random.randint(0,20),960],[np.random.randint(0,20),960],[np.random.
randint(0,4),960],

[np.random.randint(0,20),960],[np.random.randint(0,36),960],[np.random.randin
t(0,36),960],

[np.random.randint(0,36),960],[np.random.randint(0,23),960],[np.random.randin
t(0,23),960],

[np.random.randint(0,11),960],[np.random.randint(0,36),960],[np.random.randin
t(0,36),960],

[np.random.randint(0,7),960],[np.random.randint(0,7),960],[np.random.randint(
0,7),960],

[np.random.randint(0,7),960],[np.random.randint(0,20),960],[np.random.randint
(0,20),960]]"""
    if(pac_activo!=[]):
        for j in range(15):
            if(d[j][0]!=0):
                pac=random.choice(pac_activo)
                r[pac-1]=d[j][0]
                indice=pac_activo.index(pac)
                pac_activo.pop(indice)
                if(pac_activo==[]):
                    break
    return(I,pu,pp,first,last,r,tp,A,pt,pa,s,m,cv,r_tepcof,d)
```

codigo.py

```
from instancias import *
from modelos import *
import xlsxwriter

libro = xlsxwriter.Workbook('Resultados.xlsx')
hoja = libro.add_worksheet('Resultados')
random.seed(2) #Semilla 2 para [1,6,1,1,1] y semilla 1 para [4,8,1,1,1]

am=[1,2,3,4,5,6,7,8] #1°Cons, Enf, Sangre, Rayos, TAC, Reeval, 2ªEnfermería,
Alta
maxA=8
tr=[1,2,3,4,5] #Consulta, enfermería, laboratorio, rayos y sala TAC
n=[[1,2,1,1,1,1,2,1],[1,2,1,1,1,1,2,1],[1,1,1,1,1,1,1,1],[1,1,1,1,1,1,1,1],[1
,1,1,1,1,1,1,1]]
M=1600
tepcof=[0,15,60,100,120]
sat=[0.25,0.5,0.75]

#A las 6 de la mañana se tienen 1 consulta, 6 enfermerías, 1 rayos, 1 lab y 1
TAC
J=10
t=[1,2,2,2,2,2,2,3,4,5]
pr=[[1,1,1,1,0,0,0,1,1,1],[1,1,1,1,0,0,0,1,1,1],[1,0,1,1,1,1,0,1,1,1],
    [1,0,0,0,1,1,1,1,1,1],[1,0,0,0,0,1,1,1,1,1]]
b=[[1,0,0,0,0,0,0,0,0,0],[0,1,1,1,1,1,1,0,0,0],[0,0,0,0,0,0,0,1,0,0],
   [0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,0,0,1],[1,0,0,0,0,0,0,0,0,0],
   [0,1,1,1,1,1,1,0,0,0],[1,0,0,0,0,0,0,0,0,0]]

#A las 12 de la mañana se tienen 4 consultas, 8 enfermerías, 1 rayos, 1 lab y
1 TAC
"""J=15
t=[1,1,1,1,2,2,2,2,2,2,2,2,3,4,5]
pr=[[1,1,0,0,1,1,1,1,0,0,0,0,1,1,1],[1,1,0,0,1,1,1,1,0,0,0,0,1,1,1],
    [1,1,0,0,0,1,1,1,1,1,0,0,1,1,1],[0,0,1,1,0,0,0,0,1,1,1,1,1,1,1],
    [0,0,1,1,0,0,0,0,0,1,1,1,0,0,0]]
b=[[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,1,1,1,1,1,1,1,1,0,0,0],
   [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0],
   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0],
   [0,0,0,0,1,1,1,1,1,1,1,1,0,0,0],[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0]]"""

#A las 12 de la mañana se tienen 4 consultas, 8 enfermerías, 2 rayos, 2 lab y
2 TAC
"""J=18
t=[1,1,1,1,2,2,2,2,2,2,2,2,3,3,4,4,5,5]
pr=[[1,1,0,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1],[1,1,0,1,1,1,1,1,0,0,1,1,1,1,1,1,
1],

[1,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1],[0,0,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1],
    [0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0]]
b=[[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0
],

[0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1],[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
```

```
[0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0],[1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0]]
covid=[[0,0,0,1,0,0,0,0,0,0,1,1,0,1,0,1,0,1],[1,1,1,0,1,1,1,1,1,1,0,0,1,0,1,0
,1,0]]"""

hoja.write(0,1,'Número de instancia')
hoja.write(0,2,'Modelo')
hoja.write(0,3,'Valor F.O.')
hoja.write(0,4,'Status')
hoja.write(0,5,'Tiempo ejecución')
hoja.write(0,6,'GAP')
hoja.write(0,7,'Tiempo límite')
hoja.write(0,8,'Número vbles')
hoja.write(0,9,'Número restricciones')
hoja.write(0,10,'Número de pacientes')
hoja.write(0,11,'Número de actividades')
hoja.write(0,12,'Número de recursos')
hoja.write(0,13,'TEPCOF F.O.')
hoja.write(0,14,'LOS F.O.')
hoja.write(1,0,'Saturación de 25%')
hoja.write(61,0,'Saturación de 50%')
hoja.write(121,0,'Saturación de 75%')
f=1
for k in (1,7,13,19,25,31,37,43,49,55,61,67,73,79,85,91,97,103,109,115,121,
          127,133,139,145,151,157,163,169,175):
    hoja.write(k,1,str(f))
    f=f+1
    for h in range(6):
        hoja.write(k,2,str(h+1))
        k=k+1

k=1
m=[3,4,5,6,7,8,9,10,11,12,13,14]
for sat in (0.25,0.5,0.75):
    for h in range(10):
        hoja.write(k,1,str(h+1))
        instancia=Generar_Instancia(sat)
        pos=0
        P=0
        s=instancia[10]
        for f in range (len(tr)):
            pos=0
            for i in range(len(s)):
                if(s[i]==f+1):
                    pos=pos+1
            if(pos>P):
                P=pos

modelo1g=modelo1_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],
                       instancia[5],instancia[7],instancia[8],
instancia[10],instancia[11],am,maxA,J,
                       t,instancia[14],M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo1g[v-3]))
        k=k+1
```

```
modelo2g=modelo2_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],
                       instancia[5],instancia[7],instancia[8],instancia[9],
                       instancia[10],instancia[11],am,maxA,J,t,instancia[14],
                       pr,n,M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo2g[v-3]))
        k=k+1

modelo3g=modelo3_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],
                       instancia[5],instancia[7],instancia[8],instancia[9],
                       instancia[10],instancia[11],am,maxA,J,t,instancia[14],
                       pr,b,n,M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo3g[v-3]))
        k=k+1

modelo1p=modelo1_posiciones(instancia[0],instancia[1],instancia[2],instancia[
3],instancia[4],
                            instancia[5],instancia[7],instancia[8],

instancia[10],instancia[11],am,maxA,J,t,instancia[14],M,tepcof,P,instancia[13
])
        for v in (m):
            hoja.write(k,v,str(modelo1p[v-3]))
        k=k+1

modelo2p=modelo2_posiciones(instancia[0],instancia[1],instancia[2],instancia[
3],instancia[4],

instancia[5],instancia[7],instancia[8],instancia[9],

instancia[10],instancia[11],am,maxA,J,t,instancia[14],
                            pr,n,M,tepcof,P,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo2p[v-3]))
        k=k+1

modelo3p=modelo3_posiciones(instancia[0],instancia[1],instancia[2],instancia[
3],instancia[4],

instancia[5],instancia[7],instancia[8],instancia[9],

instancia[10],instancia[11],am,maxA,J,t,instancia[14],
                            pr,b,n,M,tepcof,P,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo3p[v-3]))
        k=k+1

libro.close()

"""m=[3,4,5,6,7,8,9,10,11,12,13,14]
f=1
for k in
(1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61,65,69,73,77,81,85,89,93,97,101,
105,107):
```

```
    hoja.write(k,1,str(f))
    f=f+1
    for h in range(4):
        hoja.write(k,2,str(h+1))
        k=k+1
k=1
for sat in (0.25,0.5,0.75):
    for h in range(10):
        hoja.write(k,1,str(h+1))
        instancia=Generar_Instancia(sat)

modelo1g=modelo1_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],
                       instancia[5],instancia[7],instancia[8],
instancia[10],instancia[11],am,maxA,J,
                       t,instancia[14],M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo1g[v-3]))
        k=k+1

modelo2g=modelo2_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],
                       instancia[5],instancia[7],instancia[8],instancia[9],
                       instancia[10],instancia[11],am,maxA,J,t,instancia[14],
                       pr,n,M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo2g[v-3]))
        k=k+1

modelo3g=modelo3_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],
                       instancia[5],instancia[7],instancia[8],instancia[9],
                       instancia[10],instancia[11],am,maxA,J,t,instancia[14],
                       pr,b,n,M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo3g[v-3]))
        k=k+1

modelo4g=modelo4_gamma(instancia[0],instancia[1],instancia[2],instancia[3],in
stancia[4],

instancia[5],instancia[6],instancia[7],instancia[8],instancia[9],

instancia[10],instancia[11],instancia[12],am,maxA,J,t,instancia[14],
                       pr,b,covid,n,M,tepcof,instancia[13])
        for v in (m):
            hoja.write(k,v,str(modelo4g[v-3]))
        k=k+1

libro.close()"""
```