Trabajo de Fin de Grado
Ingeniería de Organización Industrial

Optimización quirúrgica para maximizar la utilización de los recursos mediante algoritmos Cuckoo Search

Autor: Alejandro Pina Sánchez
Tutor: José Manuel Molina Pariente

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

Sevilla, 2024

Trabajo de Fin de Grado
Ingeniería de Organización Industrial

Optimización quirúrgica para maximizar la utilización de los recursos mediante algoritmos Cuckoo Search

Autor:
Alejandro Pina Sánchez

Tutor:
José Manuel Molina Pariente
Profesor Permanente Laboral

Dpto. de Organización Industrial y Gestión de Empresa I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo de Fin de Grado: Optimización quirúrgica para maximizar la utilización de los recursos mediante algoritmos Cuckoo Search

Autor: Alejandro Pina Sánchez

Tutor: José Manuel Molina Pariente

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia,

A mis amigos,

A mis maestros y compañeros,

Gracias.

Agradecimientos

Expresar mi más sincero agradecimiento a todas las personas que me han apoyado y acompañado en este camino.

En primer lugar, a mi familia, quienes han sido mi pilar fundamental. Gracias a mis padres por su amor incondicional, su apoyo constante y sus palabras de aliento en los momentos más difíciles. A mis hermanos, por estar siempre ahí, brindándome su cariño y comprensión.

A mis amigos externos, que, aunque no compartieron directamente esta experiencia académica, me ofrecieron su apoyo incondicional y su compañía en los momentos de descanso.

A mis compañeros y amigos de la universidad, con quienes compartí risas, estrés, largas horas de estudio y mucho aprendizaje. Gracias por estar ahí y por todos los buenos momentos que hemos vivido juntos.

Finalmente, a los profesores que me han guiado y orientado a lo largo de este camino. Especialmente, quiero agradecer a mi tutor de TFG, José Manuel Molina Pariente, por su paciencia, sus valiosos consejos y su dedicación. Su apoyo ha sido crucial para la realización de este proyecto. A todos mis profesores, gracias por compartir su conocimiento y por inspirarme a dar lo mejor de mí.

A todos ustedes, gracias por formar parte de este logro.

Alejandro Pina Sánchez

Sevilla, 2024

Resumen

La investigación sobre la gestión de listas de espera en los hospitales es crucial debido a la necesidad real de mejorar los tiempos de espera. Se han realizado numerosos estudios en esta área, que se pueden clasificar en dos tipos principales: estudios *"What if?"* y estudios de optimización.

Los estudios *"What if?"* exploran el impacto de modificar ciertos recursos, como agregar o reducir quirófanos, ajustar los tiempos disponibles, entre otros. Por otro lado, los estudios de optimización se enfocan en la mejor asignación de las cirugías con los recursos existentes, con el objetivo de maximizar el número de cirugías realizadas, priorizar las intervenciones más importantes o reducir el tiempo en lista de espera, entre otros fines específicos.

En el siguiente documento se recoge un estudio de optimización sobre la aplicación de algoritmos de *Cuckoo Search* en el problema de gestión de listas de espera, en este concreto, asignar intervenciones en lista de espera a un quirófano y día en el horizonte de planificación.

El objetivo a considerar de este trabajo es maximizar la utilización de los recursos disponibles. Para ello, mediante el lenguaje de programación "Python", teniendo en cuenta las limitaciones del entorno del problema, por ejemplo, los quirófanos disponibles, la disponibilidad de los cirujanos o los días en los que se pueden realizar las intervenciones, entre otros.

Se plantean 3 políticas de asignación básicas, la implementación de la metaheurística de *Cuckoo Search* para lograr el mejor resultado en la resolución del problema y dos heurísticas de mejora basadas en vecindad para intentar obtener mejores resultados.

Concluye con un análisis y comparativa con el fin de decidir la mejor combinación para implementar la metaheurística *Cuckoo Search* a la gestión de listas de espera.

Índice

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

El arte de la medicina consiste en mantener al paciente en buen estado de ánimo mientras la naturaleza cura la enfermedad

- Voltaire-

1.1 Introducción del problema

El sector de la salud, a lo largo del tiempo, ha servido de apoyo para el estudio y análisis de distintos problemas debido a la gran cantidad de actividades y actores que existen en un hospital, realizándose tanto procedimientos teóricos como prácticos. Las relaciones de dependencia entre los distintos sectores, tiempos de ejecución, colas de espera, uso de herramientas informáticas, bases de datos, etc., hacen que sea un sector con infinitos puntos de mejoras. Además, al tratarse de un servicio esencial para las personas, es de vital importancia tener el entorno lo más desarrollado y eficiente posible.

Pese a que se utilizan las máquinas y utensilios más novedosos, se siguen realizando constantes mejoras en las instalaciones, existe una constante investigación, experimentación y desarrollo, el principal problema sigue siendo los tiempos de permanencia en las listas de espera.

Las amplias listas de espera vienen derivadas por la gran diversidad de operaciones que existen y por no ser deterministas, es decir, cada cirugía que se va a realizar tiene un tiempo distinto y no conocido, dificultando así la planificación de estas intervenciones. Esto deriva en que las citas se asignen de manera cautelosa, habiendo ineficiencias por no haber realizado alguna operación que se podría haberse ejecutado finalmente. Se puede hacer una programación de los pacientes que permita orientar la jornada, pero tiene la necesidad de ser actualizada periódicamente.

Otro factor que hace que las listas de espera aumenten es la limitación de recursos disponibles. La realidad es que existe un número limitado de recursos restringiendo la capacidad del centro. Si la demanda clínica es mayor que la capacidad que se puede afrontar, inevitablemente, la lista de espera aumenta.

Como se observa, son muchos factores que se deben tener en cuenta para lograr que el sistema sanitario sea lo más eficiente posible. Gracias a la tecnología y entornos informáticos, se pueden realizar distintos estudios realizando una representación de la realidad los más fiel posible. Cuanto más completa sea esta representación, menores diferencias obtendremos entre los resultados que se obtengan de los estudios con los resultados que finalmente ocurran en la realidad.

Para afrontar los distintos problemas que surgen en los hospitales se requiere tomar decisiones, existen 3 niveles de decisiones según el alcance que presenten, nivel estratégico, táctico y operativo. Basándonos en el artículo [1], se explica a continuación los distintos niveles.

El nivel estratégico comprende decisiones a largo plazo, como son decisiones que afectan la organización y dirección y están relacionadas con el crecimiento, rentabilidad y posición del hospital, por ejemplo, inversiones, cambios de políticas, adquisición de tecnología avanzada, etc.

El nivel táctico tiene un alcance de medio plazo y son decisiones que implementan la estrategia y mejorar la eficiencia del uso de los recursos. Algunas decisiones que se encuentran en este nivel es planificar la contratación y formación de personal, gestión de los turnos del trabajo, gestión de suministros, etc.

Y, por último, el nivel operativo. Este nivel comprende decisiones a corto plazo que deben ir actualizándose con frecuencia. Estas decisiones buscan la eficiencia de los recursos definidos en el nivel táctico. Algunas de las decisiones que se toman en el nivel operativo son la gestión de lista de espera, el ajuste de la programación por eventos imprevisto, gestión de los quirófanos, etc.

En este trabajo se toman decisiones a nivel operativo, decidiendo el día, la hora de inicio y final, el quirófano y el cirujano responsable de cada cirugía. Estas decisiones comprenden la gestión de listas de espera, que comprende la asignación de cirugías a quirófanos y días, la asignación de cirujanos a quirófanos y días y la gestión de los quirófanos en el que se decide el orden de las cirugías.

En la literatura, se pueden encontrar diversos métodos para optimizar este tipo de decisiones. Cuando el problema es sencillo, con pocas restricciones y variables, los autores suelen optar por la Programación Lineal Entera Mixta (MILP), que modela el problema mediante restricciones matemáticas y busca encontrar la solución óptima. Sin embargo, cuando existen múltiples funciones objetivo, las restricciones son complejas de modelar linealmente o el tamaño del problema es demasiado grande y requiere mucho tiempo de computación, los autores prefieren utilizar metaheurísticas. Estas técnicas no garantizan encontrar la solución óptima, pero proporcionan buenas soluciones que suelen estar cerca del óptimo, gracias a la eficiente exploración del espacio de soluciones que realizan.

La finalidad de este proyecto es mejorar la eficiencia de la asignación de los pacientes de modo que se realice el mayor tiempo de cirugías posible. Para optimizar, se realizará la programación en Python de la metaheurística *Cuckoo Search (Búsqueda del Cuco)*. Este enfoque ha sido elegido por su gran interés, su eficiente modo de explorar soluciones y porque, hasta donde sabemos, no se ha aplicado previamente a este tipo de problemas. En el resto del documento se presenta las características del problema, en qué consiste la metaheurística *Cuckoo Search* y cómo se ha aplicado y un estudio de las distintas variantes estudiadas.

1.2 Definiciones

En este apartado, se presentan las definiciones de los términos clave y las nociones básicas necesarias que se utilizarán a lo largo del documento.

1.2.1 Programación de la cirugías

Para desarrollar una actividad de manera más eficiente, es fundamental comenzar con la programación de las cirugías.

La programación de las intervenciones implica definir las fechas de inicio y fin de las distintas tareas pendientes, así como el espacio y los recursos necesarios para su desarrollo.

El resultado es un cronograma detallado, generalmente representado un diagrama de Gantt, que muestra mediante barras horizontales, la hora de inicio y fin, el quirófano y el día en el que se asigna cada cirugía. Atendiendo a las distintas características del problema, podemos determinar qué tan buena es una solución en comparación con otra, evaluando aspectos como el tiempo máximo de finalización, las tareas que se han retrasado respecto a su fecha de vencimiento, el menor uso de recursos, etc.

1.2.2 Secuencia

La secuencia es la representación escrita de una solución. Representa el orden en el que se asignan las cirugías. *De ahora en adelante, se hablará de secuencia y π indistintamente.*

1.2.3 Decoding

El *decoding* es el conjunto de instrucciones que, con una secuencia de entrada y la información de los parámetros del problema, es capaz de asignar las tareas y proporcionarnos la solución para esta secuencia. Según como se definan las instrucciones, la solución variará, generando distintos diagramas de Gantt para una misma secuencia de entrada.

Suele interpretarse como una caja negra la cual nos proporciona un valor de la función objetivo para una secuencia de entrada.

1.2.4 Heurística

Una heurística es un procedimiento basado en reglas prácticas y métodos empíricos para encontrar soluciones satisfactorias a problemas complejos de manera rápida y eficiente, aunque no necesariamente óptima. Son útiles especialmente cuando el espacio de búsqueda es vasto y las soluciones exactas son computacionalmente costosas o imposibles de determinar, definido en [2].

1.2.5 Metaheurística

Las metaheurísticas son técnicas de optimización de búsqueda basadas en algoritmos heurísticos que se utilizan para resolver problemas complejos en los que el espacio de búsqueda es muy grande o desconocido. Las metaheurísticas son algoritmos de propósito general que no están diseñados para un problema específico, se basan en la exploración iterativa del espacio de soluciones, utilizando estrategias para moverse de forma inteligente por el espacio de búsqueda y para escapar de los óptimos locales, como define [3].

1.2.6 Óptimo global Vs Óptimos locales

El concepto de óptimo hace referencia a la mejor solución del problema, que será el que tiene el valor más alto, si estamos maximizando la función objetivo, o el valor más bajo si nos encontramos minimizando.

Un óptimo global corresponde a la solución con el mejor valor de la función objetivo de todo el problema. Cuando es computacionalmente imposible explorar todas las combinaciones, se opta por definir un óptimo local, que corresponde a la solución con el mejor valor de la función objetivo de las soluciones exploradas [4].

2 DESCRIPCIÓN DEL PROBLEMA

En este apartado vamos a conocer la estructura principal sobre la que se basará el resto del trabajo. En él, se explica la descripción del problema y las restricciones que aplican a nuestro alcance. El modelo base se encuentra recogido en el artículo [5].

En este artículo se describe un problema de asignación Cirugías Plástica y Quemaduras Mayores del Hospital Universitario Virgen del Rocío.

Para realizar la asignación, el decisor debe tener en cuenta las características del problema, los recursos y restricciones.

En este problema, estudiamos un horizonte temporal de ***H*** días en el que existe un número limitado de cirujanos ***(Surgeons),*** que trabajan un número limitado de días en el horizonte definido (***mds***), y quirófanos ***(J)***. Los quirófanos y los cirujanos tienen una capacidad máxima diaria ***(Cap_S y Cap_OR)*** que no pueden ser sobrepasadas, y las cirugías que se están en lista de espera para ser programadas ***(I)*** están limitadas por cuestión de logística y equipamiento a una serie de quirófanos y días ($\boldsymbol{\delta_{ijh}}$, una cirugía puede ser realizada en un día y en un quirófano concreto debido a las limitaciones que pueda presentar la sala). El número de quirófanos en los que se puede asignar un cirujano es limitado (***Lim_OR***) para reducir el tiempo de inactividad del cirujano. Por lo que los cirujanos deben asignarse a los quirófanos y días necesarios sin sobrepasar este límite.

Cada cirugía tiene una duración de intervención (***t_surgeries),*** se asigna a uno o varios cirujanos responsables en función de la especialidad del cirujano (***gamma***). Cada una de ellas tiene una fecha de alta (***rd)*** a partir de la cual se permite comenzar a asignarse.

La asignación de las cirugías se realizará a un único día, quirófano y cirujano responsable, no pudiéndose realizar dos operaciones distintas en el mismo espacio y tiempo, ni un cirujano puede realizar dos cirugías a la vez (en un mismo quirófano no se pueden realizar dos cirugías a la vez y un cirujano no puede estar realizando más de una intervención al mismo tiempo).

Las operaciones que no puedan ser programadas debido a limitaciones de recursos quedarán sin ser programadas, tendrán que ser consideradas en el siguiente horizonte de planificación.

En el artículo, se supone que el resto de los recursos necesarios están siempre disponibles y que no representan cuello de botella.

La función objetivo considerada hace referencia a maximizar la utilización de los quirófanos. El decisor cuenta con los recursos disponibles predefinidos, por lo que conviene la utilización plena de todos los quirófanos cada día, evitando dejar alguno de ellos sin actividad. En consecuencia, la función objetivo se define como la suma de los tiempos de las cirugías asignadas.

3 PROCEDIMIENTO DE RESOLUCIÓN

En este capítulo se explica los algoritmos y procedimientos que se aplican para intentar lograr optimizar el problema descrito en el ***capítulo 2***. Se realizan distintos métodos básicos de programación con las restricciones definidas para evitar incompatibilidades de recursos, el procedimiento de mejora basado en la *Búsqueda del Cuco* y unas heurísticas de mejora basada en vecindad para obtener una solución mejor final.

3.1 Decoding

Para poder evaluar una lista de espera se requiere tomar la decisión de qué operación se realiza, en qué quirófano, qué día se realiza, en qué momento inicia la operación y qué cirujano de todos los disponibles la realiza.

Para ello, se ha diseñado tres *decodings* básicos para compararlos entre ellos y poder proporcionar el modelo más eficiente. Según como se haya definido el *decoding* la asignación y valor de la función objetivo variará.

Definido en ***1.2.31.2.3 Decoding***.

3.1.1 Underloaded

La asignación *Underloaded,* a partir de la π, asigna de 1 en 1 cada paciente en el quirófano con menos tiempo de cirugía asignado, respetando todas las restricciones mencionadas anteriormente. Si una cirugía no puede ser asignada en el día 1, se realiza el mismo procedimiento en el día siguiente. Así sucesivamente hasta acabar las posibilidades de ser asignado. Si no se asigna, esta cirugía queda sin asignar y se intenta asignar la siguiente de π.

A continuación, se presenta un ejemplo gráfico del funcionamiento del *decoding*, $\pi = [1,2,3,4,5]$,***véase Ilustración 1.*** Ejemplo Underloaded 1

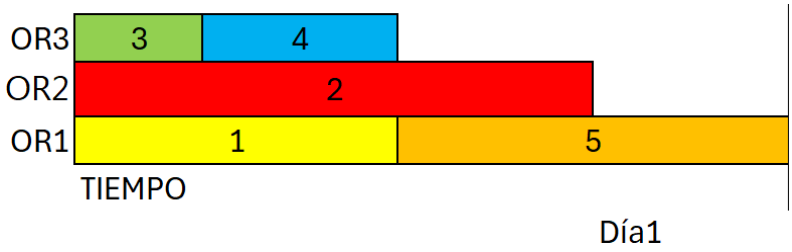


Ilustración 1. Ejemplo Underloaded 1

3.1.2 Overloaded

La asignación *Overloaded,* a partir de la π, asigna de 1 en 1 cada paciente en el quirófano con más tiempo de cirugía asignado, respetando todas las restricciones mencionadas anteriormente. Si una cirugía no puede ser

asignada en el día 1, se realiza el mismo procedimiento en el día siguiente. Así sucesivamente hasta acabar las posibilidades de ser asignado. Si no se asigna, esta cirugía queda sin asignar y se intenta asignar la siguiente de π.

A continuación, se presenta un ejemplo gráfico del funcionamiento del *decoding*, $\pi = [1,2,3,4,5]$, ***véase Ilustración 2.*** Ejemplo Overloaded 1

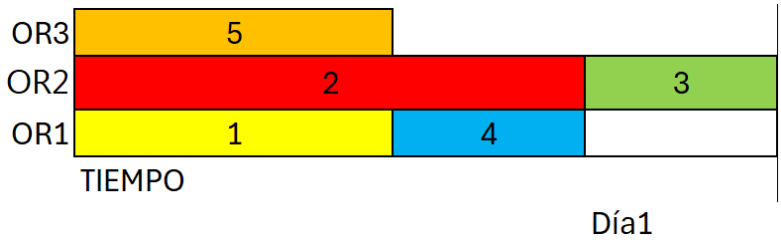


Ilustración 2. Ejemplo Overloaded 1

3.1.3 First Available

La asignación *First Available,* a partir de la π, asigna de 1 en 1 cada paciente en el primer quirófano que haya tiempo suficiente para la realización de su cirugía, respetando todas las restricciones mencionadas anteriormente. Si una cirugía no puede ser asignada en el día 1, se realiza el mismo procedimiento en el día siguiente. Así sucesivamente hasta acabar las posibilidades de ser asignado. Si no se asigna, esta cirugía queda sin asignar y se intenta asignar la siguiente de π.

A continuación, se presenta un ejemplo gráfico del funcionamiento del *decoding*, $\pi = [1,2,3,4,5]$, ***véase Ilustración 3.*** Ejemplo FirstAvailable 1

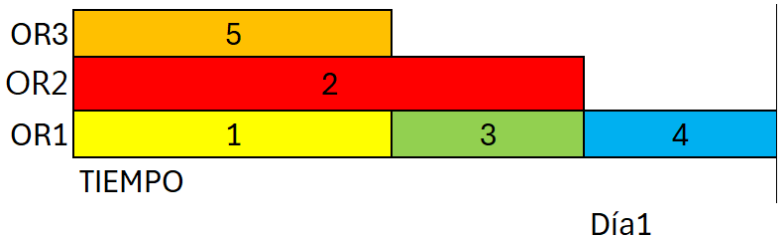


Ilustración 3. Ejemplo FirstAvailable 1

3.2 Descripción de la metaheurística Cuckoo Search

Como muchas de las metaheurísticas que se emplean para resolver problemas de programación de la producción, el *Cuckoo Search* también está inspirado en un procedimiento natural. El método *Cuckoo Search* se trata de una metaheurística basada en el comportamiento de la cría del pájaro de cuco propuesto [6].

El fundamento principal es que el cuco deposita sus huevos en los nidos de otras aves para que estas los incuben y críen. Los cucos depositan los huevos en los nidos más favorables para que prosperen, suelen

cumplir que tengan características parecidas al cascarón y que sea un ave más pequeña.

Si los padres del nido detectan al huevo de cuco, estos lo desecharán, si no es detectado, a menudo nace antes que el resto y es más grande que los polluelos del hospedador lo que le permite competir con éxito por la comida y posteriormente expulsar a los demás del nido asegurando su supervivencia.

Para aplicar el procedimiento de generación de los nidos, nos basaremos en el artículo [7]. En este enfoque, se asume que en cada iteración hay un número constante de huevos. Cada huevo, al eclosionar y buscar un nuevo nido, solo puede poner un único huevo (un huevo se encuentra en un nido, por lo que huevo y nido se consideran equivalentes).

En el contexto de la metaheurística, esto representa un escenario continuo donde cada huevo es una solución potencial. Dado que en un espacio continuo no es posible evaluar todas las soluciones posibles, este método nos permite una buena aproximación mediante la exploración eficiente del espacio de búsqueda. Un cuco que prospere en un nido (una solución prometedora) buscará otro nido para depositar su huevo (generar una nueva solución).

Las áreas donde haya más supervivencia de los cucos (mejores valores de la función objetivo) serán aquellas en las que se depositarán más huevos, generando nuevas soluciones y explorando de manera más intensiva las regiones más prometedoras del espacio de búsqueda. Este proceso iterativo permite una búsqueda adaptativa y eficiente de soluciones óptimas.

Se presenta a continuación el procedimiento de manera general, en siguientes apartados se ve cada paso con más detalle, ***véase Figura 1.*** Pseudocódigo general Cuckoo Search

%1ª iteración
Generación de nidos iniciales.
Evaluación mediante el valor de función objetivo.

%Siguientes iteraciones
De cada nido, se explora 1 nido nuevo
Con cierta probabilidad, p, el nuevo nido es desechado por la madre hospedadora, se busca uno nuevo.
Evaluación del nuevo nido mediante el valor de la función objetivo.
Elegimos para la siguiente exploración el mejor nido entre el inicial y el nuevo que genere.

%Final
La mejor solución es el mejor nido de todas las iteraciones.

Figura 1. Pseudocódigo general Cuckoo Search

3.2.1 Codificación de la solución

La codificación e interpretación de la solución es esencial para el correcto funcionamiento de los procedimientos.

Para el problema de asignación de las cirugías que estamos estudiando, la solución que se evalúa es una secuencia que interpreta una lista ordenada de los pacientes en lista de espera.

Sin embargo, para poder aplicar la metaheurística, necesitamos que la solución sea una lista de números continuos (números reales). Para ello, se establece un proceso de transformación que permite pasar de la solución continua obtenida del método *Cuckoo Search* a la secuencia, que nos servirá para introducirlas en el *decoding* correspondiente y obtener el valor de la función objetivo.

Para aplicar los movimientos y conversiones entre las soluciones en continuo y en secuencia, nos apoyaremos en el artículo [7] y en el vídeo [8], quedando definido por el siguiente código y ejemplo, ***véase Figura 2.*** Conversión continuo-secuencia

%Coloca a los pacientes del 1 al nº cirugías (I), de uno en uno, en la posición con mayor valor de la secuencia en continuo que aún no estén posicionados.

```
aux = sorted_index_desc (Solución_Continuo)
for i in range (I):
   Secuencia [aux[i]] = i
```

Figura 2. Conversión continuo-secuencia

SOL_CONT	***1.25***	***0.85***	***0.63***	***1.45***	***0.23***	***1.32***
Ordeno SOL_CONT						
Aux	***4***	***6***	***1***	***2***	***3***	***5***
Aux indica las posiciones en las que deben asignarse las cirugías por orden (del 1 a I)						
SOL_SEC	***3***	***4***	***5***	***1***	***6***	***2***

Tabla 1. Ejemplo conversión continuo-secuencia

Para pasar de secuencia (*IT*) a continuo se realiza el procedimiento inverso, necesario para poder convertir las secuencias iniciales a continuo y generar los nidos de la 1ª *iteración,* **véase Figura 3.** Conversión secuencia-continuo

%Colocamos el mayor valor en la posición del paciente 1 y el menor al paciente |I|, siguiendo la siguiente función:

$$valor_{pos_i} = 100 - \frac{Paciente_{pos_i}}{|I|} \cdot 100$$

Figura 3. Conversión secuencia-continuo

A la posición en la que se encuentre el paciente 1 (corresponderá a 0 en Python) le corresponde el valor 100, y a la posición del paciente |*I*|, el valor será 0.

3.2.2 Levy flights

En [7] se define el movimiento del cuco mediante los *vuelos de Lévy*, estudiado por el matemático francés Paul Lévy. El *vuelo de Lévy* es un movimiento con carácter aleatorio que se caracteriza por tener pasos de longitudes variables siguiendo una distribución de cola larga (***véase Ilustración 4.*** Distribución de cola pesada), lo que favorece pasos con mayores longitudes que si se hiciese con una distribución normal. Esto permite que existan vuelos con largas distancias y explore zonas totalmente distintas a las que se encontraban, como se expresa en el apartado 3 del artículo [9].

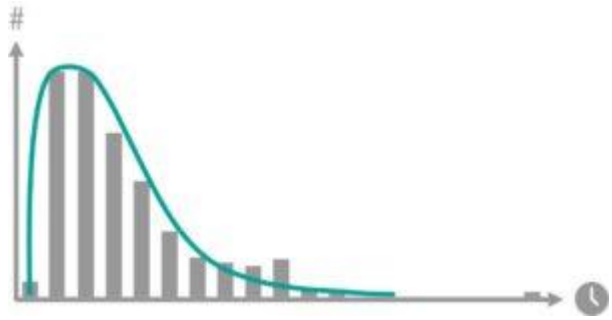


Ilustración 4. Distribución de cola pesada

A partir de un huevo (solución en continuo) de la última iteración, se generará un huevo para la siguiente iteración.

Se ha tomado el vídeo [8] como referencia para definir el procedimiento y función para generar nuevas soluciones, ***véase Ecuación 1.*** Parámetro SEn esta definición, se genera el *parámetro 'S'* que representa el paso o vuelo que debe hacer una solución, cuanto mayor distancia haya entre ella y la mejor, mayor será el vuelo que realice.

$$X_{new} = X + rand \cdot 0{,}01 \cdot s \cdot (X - best)$$

$$\sigma_u = \left(\frac{\Gamma(1+\beta) \cdot \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \rightarrow u = randn \cdot \sigma_u \quad v = randn \rightarrow s = \frac{u}{|v|^{1/\beta}}$$

Ecuación 1. Parámetro S

Gráficamente, se exploran soluciones con valores de función objetivo similares a la mejor de la generación anterior, favoreciendo vuelos largos si están muy alejadas. Al encontrar una zona (nidos) con buenos valores de función objetivo, se fomenta la exploración de esa área, ***véase el ejemplo de exploración Ilustración 5.*** Representación del Vuelo de Lévy

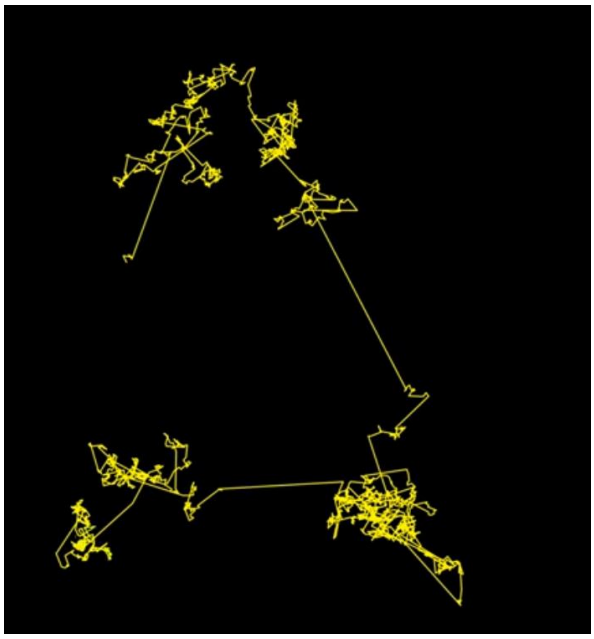


Ilustración 5. Representación del Vuelo de Lévy

3.3 Generación de nuevas soluciones

En la generación de los nidos, distinguimos 2 fases, la 1ª iteración y el resto de iteraciones. Para observar las iteraciones y huevos/nidos que evaluamos, definimos '*Z'* (parámetro que se actualizará según el tiempo computacional) y '*n*' (índice para recorrer los nidos. Basándonos en [8], se definen cinco nidos).

3.3.1 Iteración 1

La primera iteración (*Z=1*) se genera con 2 reglas contrarias, ordenar por orden decreciente las cirugías según el tiempo de cirugía y la contraria, por orden creciente. Estas 2 reglas básicas funcionan considerablemente bien en determinados escenarios siendo una buena opción contemplarlas como punto de partida. El resto de los huevos se generan de manera aleatoria, suponiendo una población de cucos iniciales dispersa por el espacio, ***véase Figura 4.*** Pseudocódigo Iteración 1

```
% Se definen 5 huevos por iteración, como en el vídeo [8]
%Iteración1
    Huevo1 = Pacientes ordenados por T_surgeries de mayor a menor
    Huevo2 = Pacientes ordenados por T_surgeries de menor a mayor
    for n =3…5:
        r = Random.Uniform(0, 1)
        Huevo[1][n][posi] = xmax*r    %xmax dimensiona para trabajar con nº mayores (xmax=100)
    Guardamos el mejor Huevo para la siguiente generación de nido
```

Figura 4. Pseudocódigo Iteración 1

3.3.2 Iteraciones Z

El resto de las iteraciones se generan a partir de las iteraciones anteriores.

Considérese que se está generando la *iteración Z*. A partir de cada huevo de la *iteración Z-1* (anterior), se calcula el *parámetro S*, que mide la distancia de este huevo al mejor de esa iteración, cuanto más distante, mayor será el vuelo, y cuanto menor sea la distancia, la búsqueda del nuevo huevo será en un entorno cercano, pues se encuentra en una buena zona de cría. En la generación de los nuevos nidos hay que tener en cuenta la posibilidad de que el huevo sea detectado por los padres hospedadores, siguiendo la función definida en [7]. Si se detecta, se pone como punto de partida el huevo detectado más un valor correspondiente a la distancia entre dos huevos de la *iteración Z-1, **véase Figura 5.*** Pseudocódigo Iteración Z

%IteraciónZ
While tiempo_ejecución < tiempo_limite:
Genera nueva iteración (Z=Z+1)
for n =1…5:
Definimos S para cada huevo (Parámetro de dispersión) con Ecuación 1. Parámetro S
%Generación mediante vuelo de Lévy de nueva posible solución
Huevo[Z][n][pos_i] = Huevo [Z-1][n][pos_i] + rand · 0,01 · S (Huevo [Z-1][n][pos_i] - Huevo [Z-1][best][pos_i])
Se convierte en secuencia y evalúa cada Huevo para obtener el valor de la función objetivo

%Detección del Huevo
%Con un probabilidad del 10% el huevo será desechado, Si se detecta:
Seleccionamos 2 huevos de la iteración[Z-1] → d1, d2
ran= random.uniform(0, 1)
%Para cada posición
Huevo[Z][n][pos_i] = Huevo [Z][n][pos_i] + ran (Huevo [Z-1][d1][pos_i] - Huevo [Z-1][d2][pos_i])
Se convierte en secuencia y aplicamos LocalSearch_GeneralSwap_FI → Huevo[Z][n]

%Actualización (se mantiene el mejor)
Si Huevo[Z][n] NO mejora a Huevo[Z-1][n] → Huevo[Z][n]= Huevo[Z-1][n]

Guardamos el mejor Huevo para la siguiente generación de nido

Figura 5. Pseudocódigo Iteración Z

En [6], establece una probabilidad de cuco de 25%. Para evitar eliminar posibles soluciones buenas y que se asemeje a *RandomWalk*, se establece una probabilidad de 10%.

La solución final que registraremos como la mejor solución explorada, será la mejor de todos los nidos explorados. Esta será la solución para el procedimiento de *Cuco Básico* y servirá de base para aplicar las heurísticas de mejora propuestas a continuación.

3.3.3 Heurística de mejora basada en vecindad

Para contrarrestar la detección del huevo intruso, hemos decidido aplicar la vecindad *General Swap* para intentar realizar una mejora de la solución inicial en caso de que el cuco tenga que buscar un nido nuevo.

Además, se proponen 2 heurísticas de mejora para aplicar a la solución final obtenida mediante la *Búsqueda del Cuco*.

3.3.3.1 General Swap First Improvement

General Swap es una técnica de optimización basada en vecindad. Consiste en intercambiar dos elementos de una solución actual y si la resultante mejora, se actualiza como nueva solución y se aplica el intercambio a esta. En el caso de que no mejore, se realiza el intercambio de otros dos elementos. Los intercambios se hacen de forma sistemática para facilitar el control. Es una estrategia eficaz para encontrar soluciones de alta calidad en problemas de optimización combinatoria.

La variante *First Improvement,* en cuanto un intercambio conlleva una mejora, se actualiza la solución a la que aplicamos *General Swap*, frente a la *Best Improvement* que evalúa todos los intercambios posibles y actualiza a la mejor de todas.

Como criterio de parada, se establece por tiempo computacional o cuando no hay intercambio que mejore la solución que se está estudiando.

La variante *First Improvement* (*GS_FI)* ha sido elegida debido a su eficiencia en problemas con una gran cantidad de secuencias. A diferencia de *Best Improvement* (*GS_BI)*, que busca exhaustivamente la mejor mejora en cada iteración y consume mucho tiempo computacional, *GS_FI* acepta la primera mejora encontrada, permitiendo realizar más iteraciones en el mismo tiempo. Esta capacidad de realizar múltiples pequeñas mejoras rápidamente lo hace más adecuado para problemas de gran escala, ya que permite una exploración más rápida y adaptativa del espacio de soluciones, ***véase el funcionamiento de GS_FI en Figura 6.*** Pseudocódigo GS_FI

```
%GeneralSwap_FI
% INPUT: Secuencia
While tiempo_ejecución  < tiempo_limite AND ExistaMejora==1:
    ExisteMejora=0  #Si no se actualiza, no existe vecino mejor, paramos de generar vecino.
    for i = 1...|I|
        for j = i+1…|I|
            Vecino = Intercambio las posiciones i y j de Secuencia
            Evaluamos Vecino

            Si Vecino es mejor que Secuencia:
                Secuencia = Vecino
                ExisteMejora=1  #Ha mejorado, exploramos los vecinos de la nueva secuencia
                %Salimos del bucle
                Break
```

Figura 6. Pseudocódigo GS_FI

3.3.3.2 Random Extraction

Random Extraction es un procedimiento basado también en vecindad. A diferencia de evaluar todos los vecinos intercambiando cada posición con el resto del procedimiento *GeneralSwap_FI*, este funcionamiento contiene aleatoriedad, ***véase Figura 7.*** Pseudocódigo Random Extraction

Consiste en intercambiar 2 posiciones aleatorias y evaluar la secuencia generada. Este intercambio corresponde a un vecino concreto de toda la vecindad de *GS_FI*.

No es conveniente realizar muchos intercambios, pues se aleja demasiado y se parece más a una solución

aleatoria, en la parte de experimentación se verifica que el mejor valor es realizar *N_intercambios = 1*.

```
%Random Extraction
% INPUT: Secuencia
While tiempo_ejecución  < tiempo_limite:
    Se define 'N_intercambios'
    Vecino= Secuencia

    %Realizamos N_intercambios veces el siguiente código
     i= random (1…|I|)
    j= random (1…|I|  AND  j ≠ i)

    %Actualización
    Vecino = Intercambio las posiciones i y j de Vecino
    Evaluamos Vecino

    Si Vecino es mejor que Secuencia:
                Secuencia = Vecino
```

Figura 7. Pseudocódigo Random Extraction

Resumiendo, se pretende realizar el procedimiento *Cuckoo Search* general y comparar los resultados obtenidos con los resultados que se obtengan si aplicamos *Random Extraction* y con los resultados que se obtengan si aplicamos *GeneralSwap_FI.*

4 GENERACIÓN DE INSTANCIAS DE DATOS

Para poder evaluar el funcionamiento de la *Búsqueda del Cuco*, se necesita de ficheros de datos. Estos datos han sido generados apoyándonos en el artículo [5], donde se definen los valores para los distintos factores y parámetros.

En esta sección se recoge de forma resumida los parámetros y variables del problema y, a continuación, los valores que adoptan.

4.1 Factores, parámetros y variables del problema

Una vez descrito el modelo base y visto qué alcance tiene el problema a estudiar, los autores definen los datos del problema de la siguiente manera:

Factores:

J: *Nº de quirófanos*

β: *Factor de exceso, las cirugías se generan de una en una hasta que excedan un $\beta\%$ del tiempo de quirófano disponible para todo el horizonte de planificación*

α: *Factor de control, regula el nº de cirujanos. El tiempo disponible de todos los cirujanos debe ser mayor o igual que $\alpha\%$ del tiempo disponible de quirófanos*

mds: *Nº máximo de días disponibles por cirujano en el horizonte*

Índices y conjuntos:

H: *Nº de días en el período de planificación*

I: *Nº de pacientes (cirugías) en lista de espera*

$Surgeons$: *Nº de cirujanos*

Parámetros

r_{jh}: *Capacidad regular del OR j en el día h (en minutos)*

a_{sh}: *Capacidad regular del cirujano s en el día h (en minutos)*

u_s: *Límite de quirófanos por día en los que puede intervenir el cirujano s*

γ_{is}: *Para cada cirugía i, contiene un vector de longitud 'Surgeons' con valores binarios que representa si el cirujano puede realizar la cirugía*

rd_i: *Fecha de alta de la cirugía i, a partir de la que puede ser asignada*

δ_{ijh}: *Parámetro binario, 1 si la cirugía i puede ser intervenida en el OR j el día h, 0 en caso contrario*

t_i: *Tiempo de intervención de la cirugía i (en minutos)*

l: *Nº de semanas en el horizonte de planificación*

4.1.1 Generación de la instancia

En este apartado, se recogen los valores de los factores que se tendrán en cuenta para la generación de la batería de problemas y los valores de los parámetros.

Factores:

$J =$ $3, 9$

$\beta =$ $100, 125$

$\alpha =$ $1.5,\ 2$

$mds =$ $3, 4$

Índices y conjuntos:

$H =$ 5

$I =$ *Generamos Cirugías (I) mientras que la suma de los tiempos de cirugías en lista de espera no exceda un β%, del tiempo de los quirófanos en el horizonte de planificación.,* $\sum_i t_i \leq J \cdot H \cdot r \cdot \beta\%$

$Surgeons =$ *Se define tantos cirujanos como necesarios para que exceda α% el tiempo de los quirófanos disponibles en el horizonte de planificación, representándose en el numerador la capacidad de los quirófanos en el horizonte temporal y en el denominador, la capacidad de un cirujano, dándonos como resultado el nº de cirujanos necesarios para igualar los tiempos disponibles. Se aplica el factor* $'\alpha'$ *para poder cubrir las posibles ineficiencias e incompatibilidades de las asignaciones, se expresa con la siguiente fórmula,* $\alpha \cdot \frac{\sum_{j \in J} \sum_{h \in H} r_{jh}}{l \cdot a \cdot mds}$

Parámetros

r_{jh}: 480 min → r = 480 min

a_{sh}: 480 min

$u_s =$ | J |

$*\gamma_{is} =$ *Este parámetro se ha modificado respecto al artículo referencia. Para representar un escenario más real, se define para cada cirugía un número aleatorio entre 1 y nº de cirujanos que define cuantos cirujanos pueden realizarla. Posteriormente, de manera aleatoria, se asignan tantos cirujanos como se hayan definido*

$rd_i =$ *Entero aleatorio entre 0 y 3 días*

δ_{ijh}: *Aleatorio binario (0,1)*

$t_i \coloneqq$ $\log_{normal}(\mu, \sigma)\ min$

μ = elección aleatoria entre los siguientes valores, [60, 120, 180, 240]

σ = elección aleatoria entre los siguientes valores, [0.1μ …0.5μ]

$l =$ 1

4.1.2 Batería de problemas

Con los factores definidos por el autor en el apartado anterior, se crean combinaciones entre ellos generando escenarios con distintas configuraciones.

Las distintas configuraciones de los escenarios vendrán definidas por los factores número de días (J= [3, 9]), *beta* (β= [100, 125]), *alpha* (α= [1.50, 2.00]) y los días de trabajo de los cirujanos (*mds*= [3, 4]), es decir, un total de 16 configuraciones de datos distintas.

Para poder realizar un estudio contundente, de cada escenario se generarán 10 instancias que, debido a la aleatoriedad en la definición de algunos parámetros, serán distintas entre ellas. Por lo tanto, quedarán finalmente 160 instancias para analizar.

A continuación, se presenta un resumen de las características de las instancias, mostrando la magnitud de los problemas y las particularidades de cada configuración, ***véase Tabla 2.*** Resumen instancias

	J	Beta	Alpha	Mds	I	Surgeons	tiempo_parada (seg)
Datos1	3	100	1.50	3	48.4	7	9.08
Datos2	3	100	1.50	4	47.8	5	8.96
Datos3	3	100	2	3	49.1	10	9.21
Datos4	3	100	2	4	47.7	7	8.94
Datos5	3	125	1.50	3	48.2	7	9.04
Datos6	3	125	1.50	4	49	5	9.19
Datos7	3	125	2	3	49.6	10	9.30
Datos8	3	125	2	4	48	7	9
Datos9	9	100	1.50	3	141.3	22	79.46
Datos10	9	100	1.50	4	141.4	16	79.54
Datos11	9	100	2	3	146.5	30	82.41
Datos12	9	100	2	4	144.9	22	81.51
Datos13	9	125	1.50	3	143.2	22	80.55
Datos14	9	125	1.50	4	142.5	16	80.16
Datos15	9	125	2	3	147.7	30	26.81
Datos16	9	125	2	4	143.8	22	80.89

Tabla 2. Resumen instancias

Los datos para cada instancia se encuentran en ***Resumen detallado de las instancias****.*

5 CALIBRACIÓN

El procedimiento general se ha explicado en el capítulo ***3. Procedimiento de Resolución.*** En este apartado se explica las consideraciones y decisiones oportunas que se han tenido que enfrentar para llegar a la conclusión final.

El *RPD*, *Relative Percentage Deviation,* es una medida que se utiliza para evaluar y comparar la calidad de las soluciones obtenidas por diferentes métodos midiendo a qué distancia, en porcentaje, se encuentra una solución hasta la mejor. Más concretamente, mide el porcentaje del mejor elemento que hace falta para que una solución esté a su nivel. Por definición, se buscan soluciones cuyo *RPD* tienda a '0', pues presentan menos desviación hasta la mejor solución obtenida.

Para cada instancia y variante a evaluar, siguiendo con las indicaciones del artículo [5], generamos el *RPD* con la fórmula que ***puede ver en Ecuación 2.*** RPD

$$RPD = \frac{FoMax - Fo}{FoMax} \cdot 100$$

Ecuación 2. RPD

Para decidir qué variante es mejor en una configuración, calcularemos el *ARPD*, *Average Relative Percentage Deviation, **véase Ecuación 3.*** ARPD

$$ARPD = \frac{1}{10}\sum_{i=1}^{10} RPD_i$$

Ecuación 3. ARPD

Como tiempo de parada, se utiliza el definido en el artículo [5], en el que se proponen 2 variantes existiendo entre ellas muy poca mejora respecto al tiempo computacional. Define el tiempo limite como ***tiempo_limite*** = ***|I| · |H| · ORs · factor_tiempo***, siendo el *factor_tiempo* 0.0125 o 0.0250. La variante de tiempo de parada que propone el autor será la que se utilice en este proyecto, definiéndose, por lo tanto, el tiempo límite como (***Ecuación 4.*** Tiempo de parada):

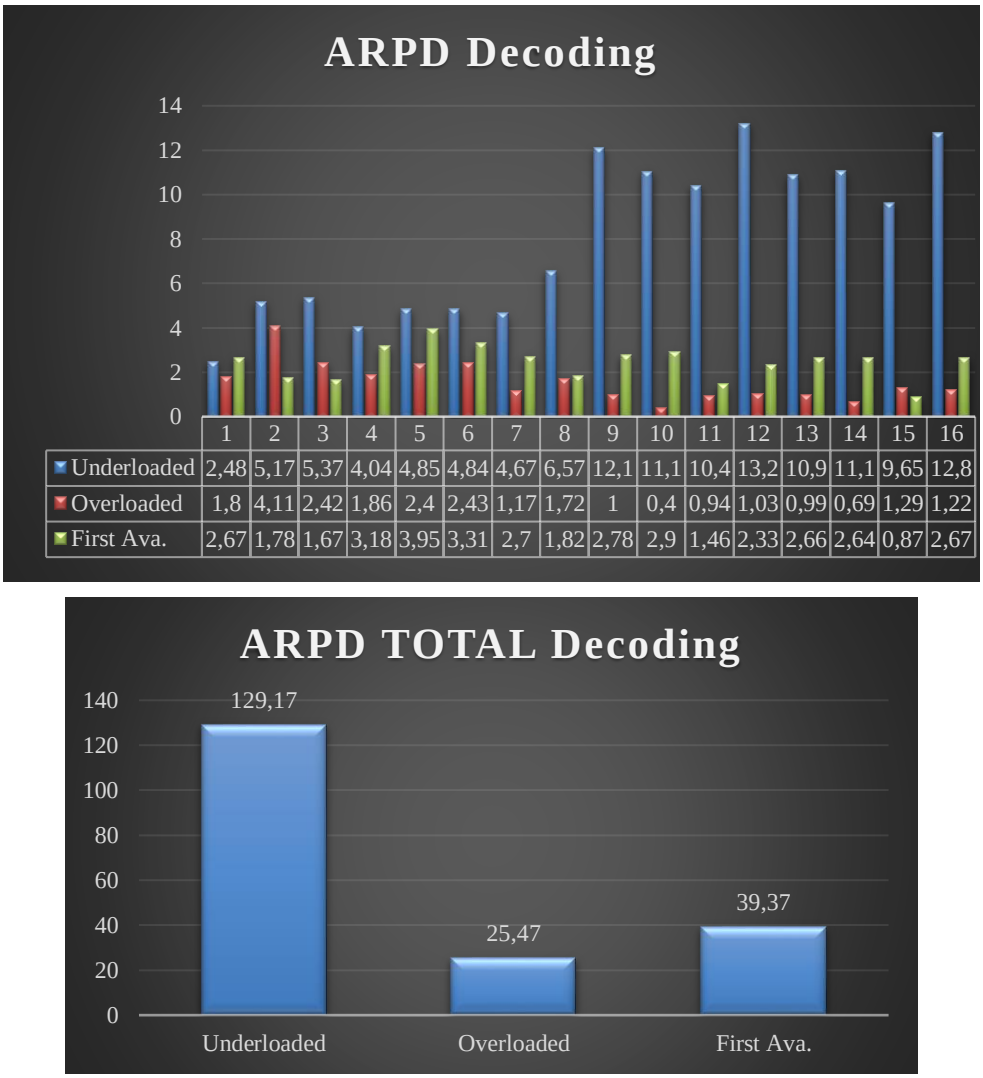
$$tiempo\_limite = |I| \cdot |H| \cdot ORs \cdot 0.0125$$

Ecuación 4. Tiempo de parada

5.1 Calibración de Decoding

Para poder realizar un estudio elaborado acerca de la implementación de la metaheurística *Cuckoo Search* en el problema de asignación para maximizar la utilización de los recursos no es suficiente aplicar el procedimiento básico de la *Búsqueda del Cuco*, se deben proponer diferentes combinaciones posibles y evaluarlas.

La primera decisión a la que se enfrenta el decisor es, ¿qué *decoding* de los 3 propuestos es el mejor? Para tomar esta decisión, se han generado secuencias aleatorias para diferentes ficheros de datos y hemos realizado la asignación mediante los distintos *decoding*. Los resultados obtenidos se pueden observar en la siguiente tabla, ***véase Ilustración 6.*** Resumen elección Decoding



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Underloaded	2,48	5,17	5,37	4,04	4,85	4,84	4,67	6,57	12,1	11,1	10,4	13,2	10,9	11,1	9,65	12,8
Overloaded	1,8	4,11	2,42	1,86	2,4	2,43	1,17	1,72	1	0,4	0,94	1,03	0,99	0,69	1,29	1,22
First Ava.	2,67	1,78	1,67	3,18	3,95	3,31	2,7	1,82	2,78	2,9	1,46	2,33	2,66	2,64	0,87	2,67

Ilustración 6. Resumen elección Decoding

Se han evaluado las 10 instancias de las 16 configuraciones (distintas a las que posteriormente se usarán para la comparación de los algoritmos) y, para cada configuración, se ha calculado el *ARPD* de cada instancia y este

valor es el que se compara, mostrándose en verde el *decoding* con menos valor de *ARPD*, el *Overloaded*, y en rojo el mayor, *Underloaded*.

Por lo tanto, el *decoding* que tomaremos para el resto de la experimentación será el *Overloaded*.

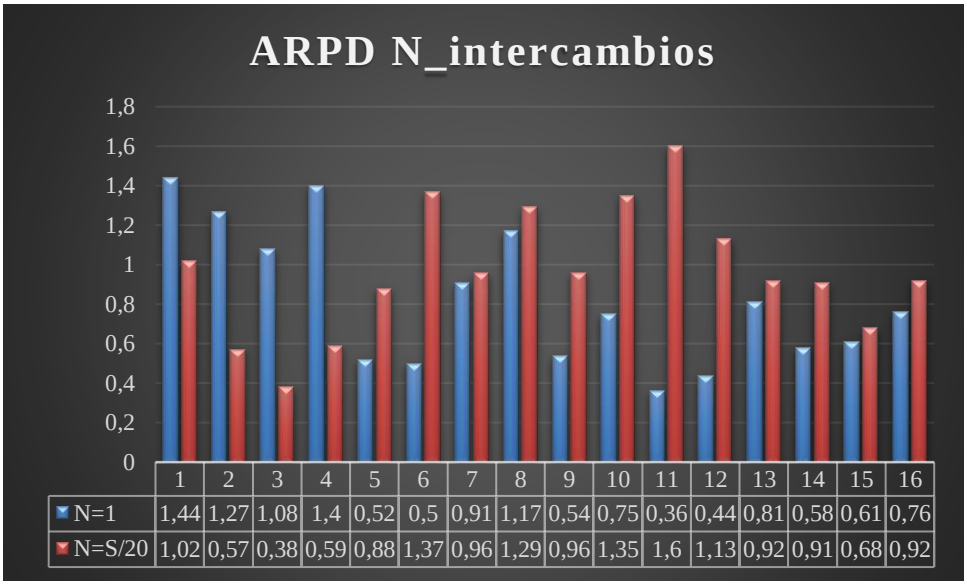
Para más detalles, consultar **Decisión Decoding.**

5.2 Calibración de N_intercambios en heurística de mejora Random Extraction

Para verificar que se toman los mejores valores entre las alternativas que tenemos, debemos realizar una serie de experimentaciones que nos permitan evaluar y comparar las distintas alternativas.

Cuando nos encontramos con un óptimo aparente del problema, este suele estar cerca de las mejores soluciones. Se requieren pequeños movimiento para observar si existen mejoras. Se decide estudiar la variante Random Extraction generando vecinos con 1 extracción (solo se intercambia una pareja de posiciones) o generando vecinos de acuerdo a la magnitud del problema, realizándose 1 intercambio de posiciones por cada 20 cirugías existentes (*N_intercambio= entero superior (|I| / 20)*).

Realizando el mismo procedimiento y batería de datos que para la elección del *decoding*, obtenemos la siguiente tabla resumen, ***véase Ilustración 7.*** Resumen elección N_intercambios



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N=1	1,44	1,27	1,08	1,4	0,52	0,5	0,91	1,17	0,54	0,75	0,36	0,44	0,81	0,58	0,61	0,76
N=S/20	1,02	0,57	0,38	0,59	0,88	1,37	0,96	1,29	0,96	1,35	1,6	1,13	0,92	0,91	0,68	0,92

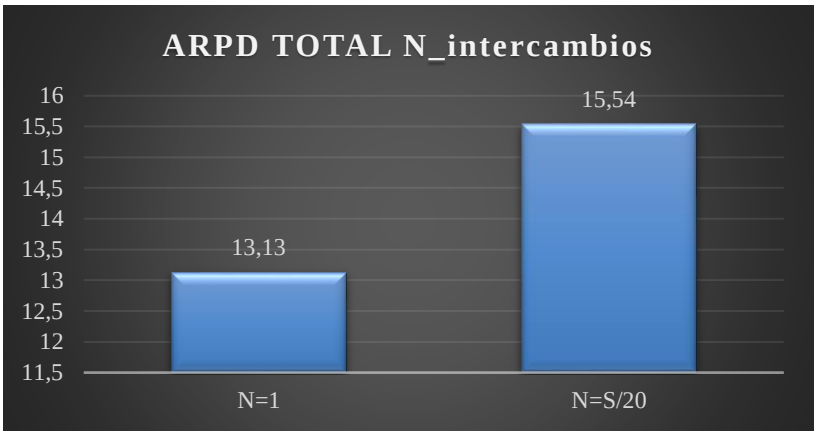


Ilustración 7. Resumen elección N_intercambios

Como se observa, realizando intercambios por pareja se obtienen mejores resultados, pues se aleja menos de la solución inicial y es la zona con mejores soluciones de las observadas. Será entonces el valor N_intercambios=1 el que se tome en consideración para realizar la mejora correspondiente.

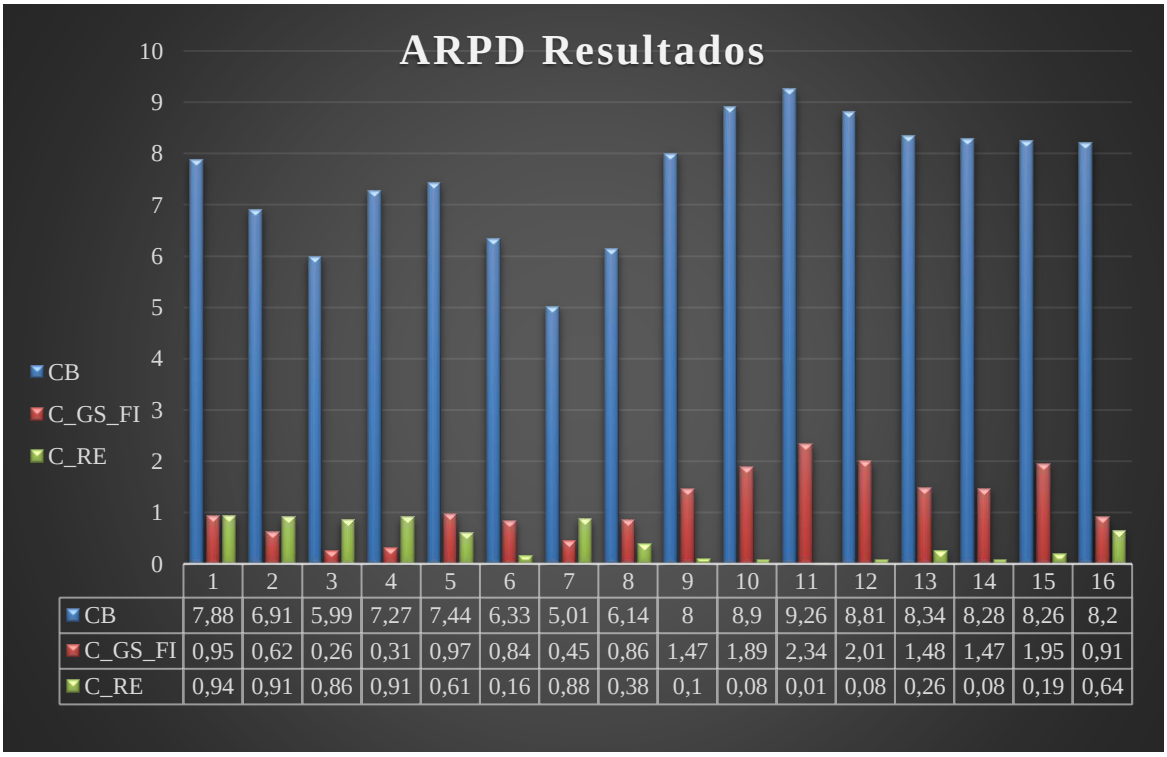
Para más detalles, consultar ***Decisión N_intercambios.***

6 ANÁLISIS DE LOS RESULTADOS

Una vez definido el procedimiento de asignación (*decoding* Overloaded), lo utilizaremos para obtener el valor de la función objetivo de las secuencias generadas mediante la *Búsqueda del Cuco*. Para decidir el número de iteraciones que evaluar, aplicamos el tiempo de parada definido al inicio del apartado ***5. Calibración***, que, mientras no se supere este umbral, se seguirán realizando iteraciones.

Tras generar la mejor solución con el procedimiento de *Cuckoo Search Basic (CB)*, aplicaremos el procedimiento *GeneralSwap_FI (GS_FI)* y conservaremos la solución con el mayor valor de la función objetivo. Luego, repetiremos este proceso con el procedimiento *Random Extraction (RE)*, aplicándo a la mejor solución obtenida por *Cuckoo Search* y mantendremos la mejor solución resultante de esta búsqueda local.

Después de generar una nueva batería de instancias de datos con los parámetros propuestos en ***4. Generación de instancias de datos*** y realizar los cálculos y asignaciones, se recogen las distintas soluciones obtenidas de los diferentes procedimientos para cada instancia y su valor de función objetivo. Con los valores obtenidos, calculamos el RPD (y ARPD) en cada instancia generada de las 3 variantes de *Cuckoo Search* que estamos estudiando, *CB*, *CB + GS_FI y CB+RE*. A partir de este documento, realizaremos la misma tabla resumen que para los apartados anteriores, que se muestran los ARPD de cada variante en cada configuración de datos, identificando así cual es la mejor combinación para aplicar la *Búsqueda de Cuco,* ***véase Ilustración 8.*** Resumen Resultados



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CB	7,88	6,91	5,99	7,27	7,44	6,33	5,01	6,14	8	8,9	9,26	8,81	8,34	8,28	8,26	8,2
C_GS_FI	0,95	0,62	0,26	0,31	0,97	0,84	0,45	0,86	1,47	1,89	2,34	2,01	1,48	1,47	1,95	0,91
C_RE	0,94	0,91	0,86	0,91	0,61	0,16	0,88	0,38	0,1	0,08	0,01	0,08	0,26	0,08	0,19	0,64

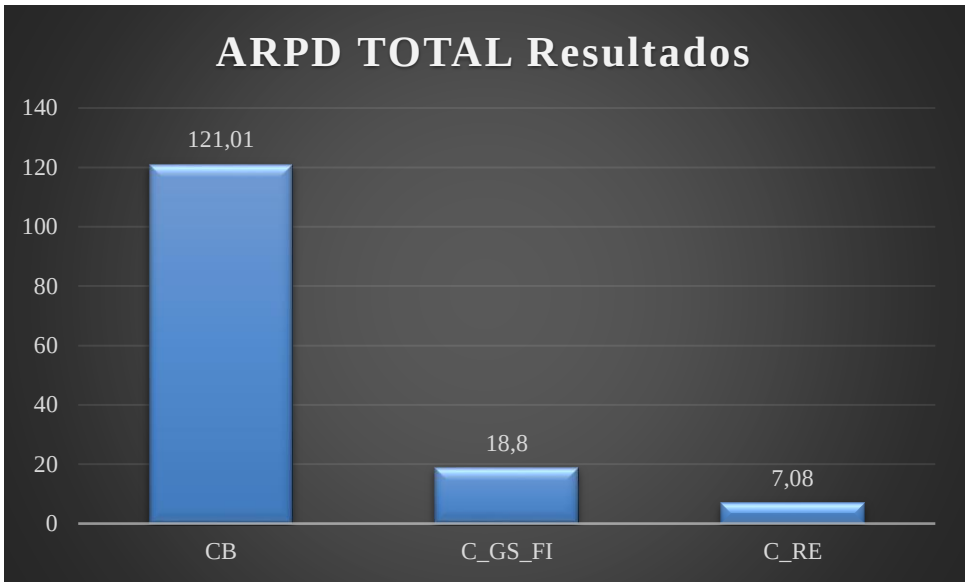


Ilustración 8. Resumen Resultados

Se puede observar los resultados detallados en minutos y RPD de cada instancia en ***7. Tabla de Resultados.***

6.1 Conclusión

Se ha realizado diferentes alternativas para implementar la metaheurística *Cuckoo Search* del modo más eficiente posible para tratar el problema de asignación de cirugías a fecha, quirófano y cirujano.

Se ha realizado diferentes baterías de problemas para la calibración de los parámetros y para el análisis de los resultados.

Una primera calibración para el procedimiento de asignación de las cirugías, *el decoding.* Con esta calibración, obtenemos entre los tres *decodings* planteados, *Underloaded, Overloaded y FirstAvailable*, continuar con el más prometedor, evitando tiempo computacional no provechoso. El *decoding* resultante es *Overloaded.*

Una vez definido el *decoding,* planteamos tres estrategias diferentes: *Realizar el Cuco Básico*, *realizar el Cuco Básico con General Swap First Improvement o realizar el procedimiento Cuco Básico con Random Extraction.*

Previamente a comparar los resultados, establecemos un tiempo computacional idéntico para cada procedimiento con el fin de evaluar en las mismas condiciones.

Para poder aplicar *Random Extraction*, se ha tenido que calibrar cuantas extracciones o intercambios se desean realizar. Con la batería de problemas, se ha calibrado el valor decidiendo entre dos posibles valores, realizar una única extracción o una extracción por cada veinte cirugías (un valor considerable para no modificar completamente las soluciones) que se deban asignar. Los resultados salen favorables para las extracciones por parejas.

Con el *decoding* seleccionado y el parámetro de intercambio para la *Random Extraction* definido, se realiza el procedimiento Cuco Básico. A partir de la solución que se ha obtenido, aplicamos las heurísticas de mejoras definidas previamente y se comparan. Observamos que las heurísticas de mejora dan un resultado considerablemente mejor (podemos observar que el *ARPD total* es bastante menor, ***véase Ilustración 8.*** Resumen Resultados por lo que sí debemos aplicar tras el procedimiento *Cuco Básico* una heurística de mejora.

Observando la ***Ilustración 8.*** Resumen Resultados**,** identificamos dos tendencias claras, desde la configuración de *Datos1* hasta *Datos8*, y desde *Datos9* hasta *Datos16*.

Si observamos la ***Tabla 2.*** Resumen instancias que diferencias hay entre estos dos grupos, observamos que son configuraciones con distintas dimensiones, correspondiendo el primer grupo a baterías de problemas más pequeños (con menor número de cirugías y cirujanos) que el segundo grupo. Por lo tanto, las conclusiones las podemos separar en dos entornos.

Para problemas pequeños, similares al primer grupo definido, observamos una pequeña dominancia de la variante con *General Swap First Improvement* respecto a la variante con *Random Extraction*. Sin embargo, para el segundo grupo de problemas, que se asemejan más a entornos reales con un mayor número de cirugías programadas, existe una dominancia notable de la variante del *Cuco Básico* con el procedimiento de mejora *Random Extraction* frente al *Cuco Básico* con *General Swap First Improvement.*

La pequeña diferencia entre *CB+RE* y *CB+GS_FI* en problemas de menor magnitud, y la significativa ventaja de *CB+RE* en problemas más grandes y realistas, nos lleva a concluir que la mejor combinación para aplicar la metaheurística *Cuckoo Search* a la asignación de cirugías a quirófanos es la regla de asignación *Overloaded* combinada con el procedimiento de mejora *Random Extraction*, generando vecinos intercambiando solo dos posiciones seleccionadas al azar. Esta combinación resuelve tanto problemas pequeños como grandes de manera efectiva, mejorando la asignación y utilización de recursos quirúrgicos.

La asignación *Overloaded* da preferencia a completar la capacidad de un quirófano antes de abrir otro, lo que nos asegura que se va a aprovechar al máximo la capacidad.

El procedimiento de la *Búsqueda del Cuco Básico* permite explorar con poca profundidad todo el espacio de soluciones y con mayor grado de detalles aquellas zonas con potencial de ser el óptimo del problema.

Y por último, con el procedimiento *Random Extraction*, permite buscar soluciones cercanas a la obtenida con el procedimiento *Búsqueda del Cuco Básico,* pero sin seguir una lógica lo que hace que esta búsqueda sea más rápida y haya cambios más bruscos de dirección que como lo hacía la *GeneralSwap_FI.*

7 ANEXO

7.1 Resumen detallado de las instancias

1	I	S	tiempo parada
DATOS1.1	49	7	9,19
DATOS1.2	45	7	8,44
DATOS1.3	45	7	8,44
DATOS1.4	49	7	9,19
DATOS1.5	52	7	9,75
DATOS1.6	46	7	8,63
DATOS1.7	48	7	9,00
DATOS1.8	52	7	9,75
DATOS1.9	50	7	9,38
DATOS1.10	48	7	9,00
DATOS1	48,4	7	9,08

2	I	S	tiempo parada
DATOS2.1	49	5	9,19
DATOS2.2	46	5	8,63
DATOS2.3	50	5	9,38
DATOS2.4	51	5	9,56
DATOS2.5	47	5	8,81
DATOS2.6	44	5	8,25
DATOS2.7	51	5	9,56
DATOS2.8	46	5	8,63
DATOS2.9	45	5	8,44
DATOS2.10	49	5	9,19
DATOS2	47,8	5	8,96

3	I	S	tiempo parada
DATOS3.1	50	10	9,38
DATOS3.2	45	10	8,44
DATOS3.3	47	10	8,81
DATOS3.4	50	10	9,38
DATOS3.5	49	10	9,19
DATOS3.6	53	10	9,94
DATOS3.7	52	10	9,75
DATOS3.8	43	10	8,06
DATOS3.9	55	10	10,31
DATOS3.10	47	10	8,81
DATOS3	49,1	10	9,21

4	I	S	tiempo parada
DATOS4.1	43	7	8,06
DATOS4.2	49	7	9,19
DATOS4.3	46	7	8,63
DATOS4.4	48	7	9,00
DATOS4.5	50	7	9,38
DATOS4.6	47	7	8,81
DATOS4.7	51	7	9,56
DATOS4.8	47	7	8,81
DATOS4.9	47	7	8,81

9	I	S	tiempo parada
DATOS9.1	131	22	24,56
DATOS9.2	145	22	27,19
DATOS9.3	147	22	27,56
DATOS9.4	140	22	26,25
DATOS9.5	147	22	27,56
DATOS9.6	128	22	24,00
DATOS9.7	142	22	26,63
DATOS9.8	142	22	26,63
DATOS9.9	152	22	28,50
DATOS9.10	139	22	26,06
DATOS9	141,3	22	79,48

10	I	S	tiempo parada
DATOS10.1	137	16	25,69
DATOS10.2	150	16	28,13
DATOS10.3	149	16	27,94
DATOS10.4	137	16	25,69
DATOS10.5	139	16	26,06
DATOS10.6	143	16	26,81
DATOS10.7	142	16	26,63
DATOS10.8	142	16	26,63
DATOS10.9	154	16	28,88
DATOS10.10	121	16	22,69
DATOS10	141,4	16	79,54

11	I	S	tiempo parada
DATOS11.1	151	30	28,31
DATOS11.2	141	30	26,44
DATOS11.3	150	30	28,13
DATOS11.4	150	30	28,13
DATOS11.5	150	30	28,13
DATOS11.6	147	30	27,56
DATOS11.7	146	30	27,38
DATOS11.8	143	30	26,81
DATOS11.9	142	30	26,63
DATOS11.10	145	30	27,19
DATOS11	146,5	30	82,41

12	I	S	tiempo parada
DATOS12.1	143	22	26,81
DATOS12.2	148	22	27,75
DATOS12.3	143	22	26,81
DATOS12.4	142	22	26,63
DATOS12.5	151	22	28,31
DATOS12.6	148	22	27,75
DATOS12.7	150	22	28,13
DATOS12.8	140	22	26,25
DATOS12.9	140	22	26,25

DATOS4.10	49	7	9,19
DATOS4	47,7	7	8,94

5	I	S	tiempo parada
DATOS5.1	52	7	9,75
DATOS5.2	50	7	9,38
DATOS5.3	53	7	9,94
DATOS5.4	44	7	8,25
DATOS5.5	49	7	9,19
DATOS5.6	48	7	9,00
DATOS5.7	49	7	9,19
DATOS5.8	48	7	9,00
DATOS5.9	45	7	8,44
DATOS5.10	44	7	8,25
DATOS5	48,2	7	9,04

6	I	S	tiempo parada
DATOS6.1	51	5	9,56
DATOS6.2	51	5	9,56
DATOS6.3	49	5	9,19
DATOS6.4	49	5	9,19
DATOS6.5	46	5	8,63
DATOS6.6	47	5	8,81
DATOS6.7	47	5	8,81
DATOS6.8	48	5	9,00
DATOS6.9	53	5	9,94
DATOS6.10	49	5	9,19
DATOS6	49	5	9,19

7	I	S	tiempo parada
DATOS7.1	53	10	9,94
DATOS7.2	49	10	9,19
DATOS7.3	52	10	9,75
DATOS7.4	47	10	8,81
DATOS7.5	48	10	9,00
DATOS7.6	47	10	8,81
DATOS7.7	51	10	9,56
DATOS7.8	53	10	9,94
DATOS7.9	45	10	8,44
DATOS7.10	51	10	9,56
DATOS7	49,6	10	9,30

8	I	S	tiempo parada
DATOS8.1	48	7	9,00
DATOS8.2	50	7	9,38
DATOS8.3	48	7	9,00
DATOS8.4	47	7	8,81
DATOS8.5	51	7	9,56
DATOS8.6	48	7	9,00
DATOS8.7	43	7	8,06
DATOS8.8	51	7	9,56
DATOS8.9	48	7	9,00
DATOS8.10	46	7	8,63
DATOS8	48	7	9,00

DATOS12.10	144	22	27,00
DATOS12	144,9	22	81,51

13	I	S	tiempo parada
DATOS13.1	138	22	25,88
DATOS13.2	139	22	26,06
DATOS13.3	137	22	25,69
DATOS13.4	152	22	28,50
DATOS13.5	146	22	27,38
DATOS13.6	150	22	28,13
DATOS13.7	140	22	26,25
DATOS13.8	143	22	26,81
DATOS13.9	145	22	27,19
DATOS13.10	142	22	26,63
DATOS13	143,2	22	80,55

14	I	S	tiempo parada
DATOS14.1	151	16	28,31
DATOS14.2	138	16	25,88
DATOS14.3	144	16	27,00
DATOS14.4	140	16	26,25
DATOS14.5	143	16	26,81
DATOS14.6	148	16	27,75
DATOS14.7	144	16	27,00
DATOS14.8	138	16	25,88
DATOS14.9	143	16	26,81
DATOS14.10	136	16	25,50
DATOS14	142,5	16	80,16

15	I	S	tiempo parada
DATOS15.1	141	30	26,44
DATOS15.2	145	30	27,19
DATOS15.3	138	30	25,88
DATOS15.4	155	30	29,06
DATOS15.5	164	30	30,75
DATOS15.6	146	30	27,38
DATOS15.7	153	30	28,69
DATOS15.8	150	30	28,13
DATOS15.9	142	30	26,63
DATOS15.10	143	30	26,81
DATOS15	147,7	30	83,08

16	I	S	tiempo parada
DATOS16.1	148	22	27,75
DATOS16.2	145	22	27,19
DATOS16.3	144	22	27,00
DATOS16.4	135	22	25,31
DATOS16.5	133	22	24,94
DATOS16.6	151	22	28,31
DATOS16.7	144	22	27,00
DATOS16.8	149	22	27,94
DATOS16.9	148	22	27,75
DATOS16.10	141	22	26,44
DATOS16	143,8	22	80,89

7.2 Comparaciones y Calibración

7.2.1 Decisión Decoding

RPD	Und	Ove	First
Datos1_1.txt	4.06	0.00	7.60
Datos1_2.txt	9.91	0.00	2.40
Datos1_3.txt	4.63	0.00	1.04
Datos1_4.txt	1.18	1.22	0.00
Datos1_5.txt	4.98	2.48	0.00
Datos1_6.txt	0.00	2.68	0.10
Datos1_7.txt	0.00	0.95	2.92
Datos1_8.txt	0.00	7.24	1.49
Datos1_9.txt	0.00	2.77	1.74
Datos1_10.txt	0.00	0.63	9.38
	2.48	1.80	2.67

RPD	Und	Ove	First
Datos2_1.txt	7.46	9.43	0.00
Datos2_2.txt	7.51	0.00	6.37
Datos2_3.txt	0.00	6.69	0.97
Datos2_4.txt	17.57	3.25	0.00
Datos2_5.txt	0.00	5.02	2.73
Datos2_6.txt	4.88	8.55	0.00
Datos2_7.txt	2.53	1.46	0.00
Datos2_8.txt	11.50	0.00	3.34
Datos2_9.txt	0.31	0.00	0.00
Datos2_10.txt	0.00	6.74	4.40
	5.17	4.11	1.78

RPD	Und	Ove	First
Datos3_1.txt	5.99	4.44	0.00
Datos3_2.txt	9.10	0.00	4.92
Datos3_3.txt	3.34	0.00	1.14
Datos3_4.txt	3.72	1.14	0.00
Datos3_5.txt	8.37	8.37	0.00
Datos3_6.txt	1.42	0.00	6.09
Datos3_7.txt	1.89	2.05	0.00
Datos3_8.txt	4.05	0.00	4.56
Datos3_9.txt	6.24	2.38	0.00
Datos3_10.txt	9.59	5.80	0.00
	5.37	2.42	1.67

RPD	Und	Ove	First
Datos4_1.txt	4.03	0.00	2.96
Datos4_2.txt	15.06	2.96	0.00
Datos4_3.txt	0.00	2.50	3.91
Datos4_4.txt	0.00	1.02	2.33
Datos4_5.txt	0.00	0.93	6.73
Datos4_6.txt	2.20	0.24	0.00
Datos4_7.txt	0.61	0.00	0.00
Datos4_8.txt	11.24	0.00	5.26
Datos4_9.txt	7.23	0.00	2.95
Datos4_10.txt	0.00	10.95	7.61
	4.04	1.86	3.18

RPD	Und	Ove	First
Datos5_1.txt	0.00	4.65	4.65
Datos5_2.txt	8.06	0.00	10.90
Datos5_3.txt	0.00	8.44	4.48
Datos5_4.txt	8.52	0.00	0.00
Datos5_5.txt	14.17	0.00	11.92
Datos5_6.txt	1.87	0.00	3.37
Datos5_7.txt	4.67	5.49	0.00
Datos5_8.txt	5.76	0.00	4.19
Datos5_9.txt	5.18	5.40	0.00
Datos5_10.txt	0.32	0.00	0.00
	4.85	2.40	3.95

RPD	Und	Ove	First
Datos6_1.txt	0.66	0.06	0.00
Datos6_2.txt	0.91	0.00	2.14
Datos6_3.txt	4.74	0.00	1.25
Datos6_4.txt	0.62	5.37	0.00
Datos6_5.txt	8.03	8.83	0.00
Datos6_6.txt	13.78	0.00	7.08
Datos6_7.txt	0.00	10.07	0.96
Datos6_8.txt	4.43	0.00	4.54
Datos6_9.txt	2.63	0.00	2.51
Datos6_10.txt	12.63	0.00	14.61
	4.84	2.43	3.31

RPD	Und	Ove	First
Datos7_1.txt	4.15	1.03	0.00
Datos7_2.txt	8.12	0.00	2.32
Datos7_3.txt	0.00	6.27	13.85
Datos7_4.txt	1.07	0.00	3.12
Datos7_5.txt	2.95	1.21	0.00
Datos7_6.txt	16.16	0.00	7.59
Datos7_7.txt	3.77	0.93	0.00
Datos7_8.txt	0.82	0.00	0.00
Datos7_9.txt	6.24	0.00	0.08
Datos7_10.txt	3.41	2.28	0.00
	4.67	1.17	2.70

RPD	Und	Ove	First
Datos8_1.txt	19.78	0.00	3.04
Datos8_2.txt	1.13	4.34	0.00
Datos8_3.txt	4.26	0.99	0.00
Datos8_4.txt	7.77	8.67	0.00
Datos8_5.txt	7.18	0.00	0.00
Datos8_6.txt	3.97	0.00	0.85
Datos8_7.txt	7.98	1.11	0.00
Datos8_8.txt	6.37	0.00	5.91
Datos8_9.txt	7.22	0.00	6.31
Datos8_10.txt	0.00	2.13	2.13
	6.57	1.72	1.82

RPD	Und	Ove	First
Datos9_1.txt	7.34	0.00	4.69
Datos9_2.txt	9.15	0.00	1.39
Datos9_3.txt	17.50	0.00	5.81
Datos9_4.txt	13.53	0.00	2.66
Datos9_5.txt	12.46	10.00	0.00
Datos9_6.txt	8.19	0.00	3.03
Datos9_7.txt	12.33	0.00	4.71
Datos9_8.txt	7.00	0.00	0.15
Datos9_9.txt	18.87	0.00	0.19
Datos9_10.txt	14.80	0.00	5.13
	12.12	1.00	2.78

RPD	Und	Ove	First
Datos10_1.txt	14.70	0.00	1.82
Datos10_2.txt	11.69	0.00	8.24
Datos10_3.txt	7.06	0.00	0.60
Datos10_4.txt	13.34	0.00	8.54
Datos10_5.txt	12.68	0.00	4.69
Datos10_6.txt	13.01	0.00	0.78
Datos10_7.txt	13.05	1.23	0.00
Datos10_8.txt	6.70	2.76	0.00
Datos10_9.txt	6.84	0.00	0.56
Datos10_10.txt	11.46	0.00	3.80
	11.05	0.40	2.90

RPD	Und	Ove	First
Datos11_1.txt	12.81	6.28	0.00
Datos11_2.txt	14.84	0.00	4.21
Datos11_3.txt	7.67	0.36	0.00
Datos11_4.txt	1.92	0.00	0.01
Datos11_5.txt	12.33	0.00	1.80
Datos11_6.txt	8.97	0.00	3.90
Datos11_7.txt	10.14	2.46	0.00
Datos11_8.txt	13.96	0.13	0.00
Datos11_9.txt	11.83	0.00	4.69
Datos11_10.txt	9.78	0.22	0.00
	10.42	0.94	1.46

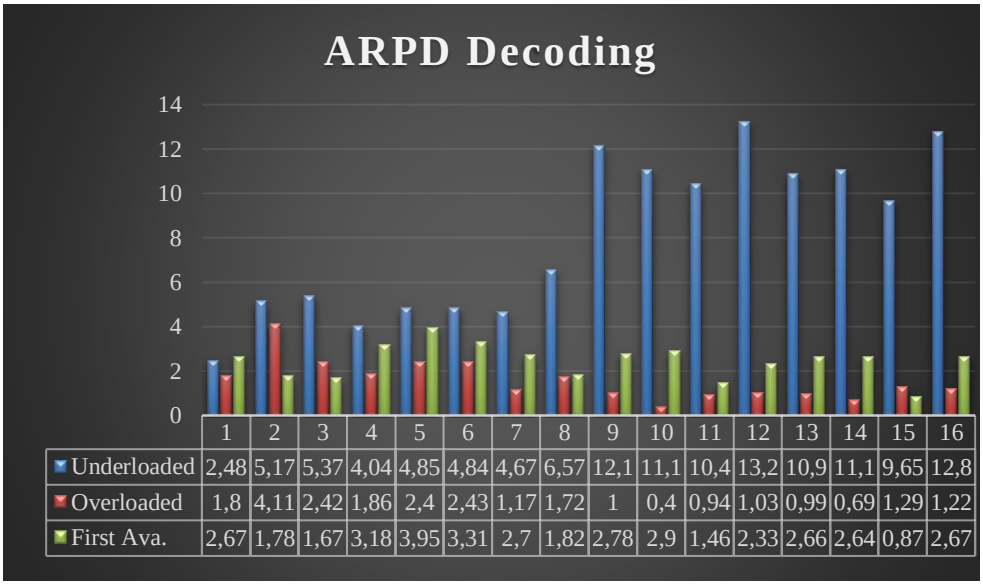
RPD	Und	Ove	First
Datos12_1.txt	15.47	0.00	2.33
Datos12_2.txt	5.32	0.00	3.80
Datos12_3.txt	16.03	0.00	6.40
Datos12_4.txt	14.57	3.96	0.00
Datos12_5.txt	13.67	0.00	4.89
Datos12_6.txt	18.72	0.00	2.25
Datos12_7.txt	11.90	2.26	0.00
Datos12_8.txt	5.43	4.04	0.00
Datos12_9.txt	13.62	0.00	3.27
Datos12_10.txt	17.31	0.00	0.35
	13.21	1.03	2.33

RPD	Und	Ove	First
Datos13_1.txt	10.31	0.00	5.16
Datos13_2.txt	13.42	0.00	4.11
Datos13_3.txt	5.40	0.00	3.20
Datos13_4.txt	14.54	0.00	3.13
Datos13_5.txt	10.30	1.84	0.00
Datos13_6.txt	9.60	0.00	5.19
Datos13_7.txt	9.93	0.00	2.63
Datos13_8.txt	14.11	0.00	3.13
Datos13_9.txt	7.49	1.36	0.00
Datos13_10.txt	13.62	6.66	0.00
	10.87	0.99	2.66

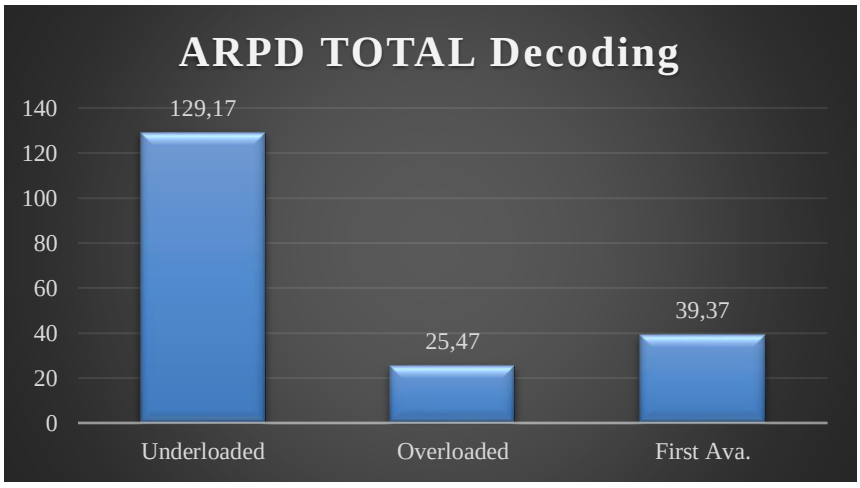
RPD	Und	Ove	First
Datos14_1.txt	11.87	0.00	1.95
Datos14_2.txt	13.54	0.00	0.95
Datos14_3.txt	14.55	0.00	9.11
Datos14_4.txt	10.88	3.37	0.00
Datos14_5.txt	8.93	0.00	1.49
Datos14_6.txt	9.78	3.53	0.00
Datos14_7.txt	9.14	0.00	5.34
Datos14_8.txt	9.31	0.00	3.40
Datos14_9.txt	15.07	0.00	0.70
Datos14_10.txt	7.62	0.00	3.47
	11.07	0.69	2.64

RPD	Und	Ove	First
Datos15_1.txt	9.11	2.10	0.00
Datos15_2.txt	10.57	0.77	0.00
Datos15_3.txt	15.44	0.77	0.00
Datos15_4.txt	12.48	6.21	0.00
Datos15_5.txt	4.96	0.00	4.26
Datos15_6.txt	6.30	0.00	0.23
Datos15_7.txt	6.40	2.89	0.00
Datos15_8.txt	10.46	0.00	3.83
Datos15_9.txt	8.02	0.20	0.00
Datos15_10.txt	12.77	0.00	0.34
	9.65	1.29	0.87

RPD	Und	Ove	First
Datos16_1.txt	18.21	0.34	0.00
Datos16_2.txt	8.38	0.00	5.57
Datos16_3.txt	17.18	0.00	6.26
Datos16_4.txt	15.15	0.00	2.24
Datos16_5.txt	11.76	4.43	0.00
Datos16_6.txt	9.96	0.00	6.14
Datos16_7.txt	11.42	0.11	0.00
Datos16_8.txt	9.68	4.29	0.00
Datos16_9.txt	11.73	0.00	6.47
Datos16_10.txt	14.42	3.00	0.00
	12.79	1.22	2.67



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Underloaded	2,48	5,17	5,37	4,04	4,85	4,84	4,67	6,57	12,1	11,1	10,4	13,2	10,9	11,1	9,65	12,8
Overloaded	1,8	4,11	2,42	1,86	2,4	2,43	1,17	1,72	1	0,4	0,94	1,03	0,99	0,69	1,29	1,22
First Ava.	2,67	1,78	1,67	3,18	3,95	3,31	2,7	1,82	2,78	2,9	1,46	2,33	2,66	2,64	0,87	2,67



7.2.2 Decisión N_intercambios

RPD	N=1	N=I/20
Datos1_1.txt	0.00	4.40
Datos1_2.txt	0.00	0.50
Datos1_3.txt	0.00	1.31
Datos1_4.txt	0.00	0.22
Datos1_5.txt	2.45	0.00
Datos1_6.txt	0.00	3.80
Datos1_7.txt	1.70	0.00
Datos1_8.txt	4.12	0.00
Datos1_9.txt	4.96	0.00
Datos1_10.txt	1.14	0.00
	1.44	1.02

RPD	N=1	N=I/20
Datos2_1.txt	0.16	0.00
Datos2_2.txt	0.00	1.57
Datos2_3.txt	0.94	0.00
Datos2_4.txt	0.00	0.70
Datos2_5.txt	4.81	0.00
Datos2_6.txt	0.00	3.44
Datos2_7.txt	0.12	0.00
Datos2_8.txt	1.56	0.00
Datos2_9.txt	0.00	0.05
Datos2_10.txt	5.15	0.00
	1.27	0.57

RPD	N=1	N=I/20
Datos3_1.txt	4.11	0.00
Datos3_2.txt	1.77	0.00
Datos3_3.txt	0.10	0.00
Datos3_4.txt	1.05	0.00
Datos3_5.txt	0.00	1.59
Datos3_6.txt	0.00	2.14
Datos3_7.txt	0.00	0.06
Datos3_8.txt	0.28	0.00
Datos3_9.txt	3.16	0.00
Datos3_10.txt	0.37	0.00
	1.08	0.38

RPD	N=1	N=I/20
Datos4_1.txt	0.03	0.00
Datos4_2.txt	1.77	0.00
Datos4_3.txt	1.95	0.00
Datos4_4.txt	0.00	5.69
Datos4_5.txt	3.19	0.00
Datos4_6.txt	3.59	0.00
Datos4_7.txt	0.73	0.00
Datos4_8.txt	0.89	0.00
Datos4_9.txt	1.81	0.00
Datos4_10.txt	0.00	0.26
	1.40	0.59

RPD	N=1	N=I/20
Datos5_1.txt	0.00	0.27
Datos5_2.txt	1.43	0.00
Datos5_3.txt	0.00	0.07
Datos5_4.txt	0.66	0.00
Datos5_5.txt	0.00	3.55
Datos5_6.txt	0.00	0.37
Datos5_7.txt	1.97	0.00
Datos5_8.txt	1.15	0.00
Datos5_9.txt	0.00	1.82
Datos5_10.txt	0.00	2.74
	0.52	0.88

RPD	N=1	N=I/20
Datos6_1.txt	0.00	0.05
Datos6_2.txt	0.00	2.22
Datos6_3.txt	1.47	0.00
Datos6_4.txt	0.00	0.71
Datos6_5.txt	0.00	1.87
Datos6_6.txt	0.00	0.69
Datos6_7.txt	1.87	0.00
Datos6_8.txt	1.64	0.00
Datos6_9.txt	0.00	5.55
Datos6_10.txt	0.00	2.64
	0.50	1.37

RPD	N=1	N=I/20
Datos7_1.txt	0.00	1.28
Datos7_2.txt	0.00	1.96
Datos7_3.txt	0.00	1.49
Datos7_4.txt	1.23	0.00
Datos7_5.txt	4.49	0.00
Datos7_6.txt	2.48	0.00
Datos7_7.txt	0.00	0.87
Datos7_8.txt	0.90	0.00
Datos7_9.txt	0.00	2.44
Datos7_10.txt	0.00	1.51
	0.91	0.96

RPD	N=1	N=I/20
Datos8_1.txt	0.00	4.02
Datos8_2.txt	0.00	2.53
Datos8_3.txt	0.00	2.16
Datos8_4.txt	0.00	1.28
Datos8_5.txt	1.52	0.00
Datos8_6.txt	0.00	1.08
Datos8_7.txt	2.95	0.00
Datos8_8.txt	6.97	0.00
Datos8_9.txt	0.24	0.00
Datos8_10.txt	0.00	1.86
	1.17	1.29

RPD	N=1	N=I/20
Datos9_1.txt	0.00	1.34
Datos9_2.txt	2.78	0.00
Datos9_3.txt	0.00	1.62
Datos9_4.txt	0.00	0.09
Datos9_5.txt	0.91	0.00
Datos9_6.txt	0.00	1.91
Datos9_7.txt	0.00	2.72
Datos9_8.txt	0.00	1.05
Datos9_9.txt	1.69	0.00
Datos9_10.txt	0.00	0.85
	0.54	0.96

RPD	N=1	N=I/20
Datos10_1.txt	0.00	0.06
Datos10_2.txt	2.33	0.00
Datos10_3.txt	0.00	2.41
Datos10_4.txt	0.00	2.01
Datos10_5.txt	0.00	2.32
Datos10_6.txt	3.06	0.00
Datos10_7.txt	0.00	0.01
Datos10_8.txt	2.13	0.00
Datos10_9.txt	0.00	2.33
Datos10_10.txt	0.00	4.32
	0.75	1.35

RPD	N=1	N=I/20
Datos11_1.txt	0.57	0.00
Datos11_2.txt	1.88	0.00
Datos11_3.txt	0.00	3.39
Datos11_4.txt	0.00	1.82
Datos11_5.txt	0.00	1.76
Datos11_6.txt	0.00	2.51
Datos11_7.txt	0.00	1.84
Datos11_8.txt	1.10	0.00
Datos11_9.txt	0.00	2.41
Datos11_10.txt	0.00	2.29
	0.36	1.60

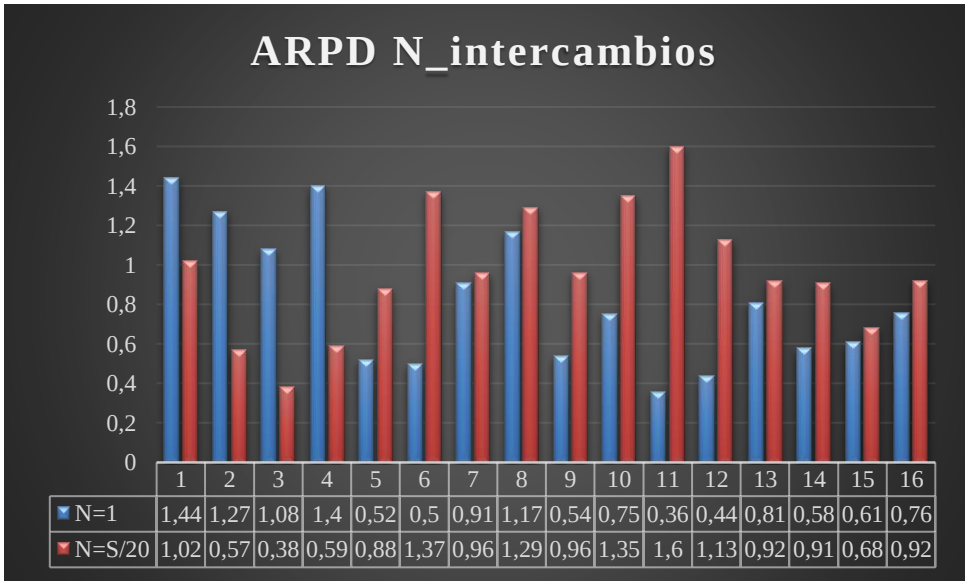
RPD	N=1	N=I/20
Datos12_1.txt	0.00	1.68
Datos12_2.txt	0.00	2.86
Datos12_3.txt	0.00	0.96
Datos12_4.txt	0.00	2.06
Datos12_5.txt	2.13	0.00
Datos12_6.txt	0.00	0.36
Datos12_7.txt	0.00	0.78
Datos12_8.txt	1.88	0.00
Datos12_9.txt	0.00	2.56
Datos12_10.txt	0.37	0.00
	0.44	1.13

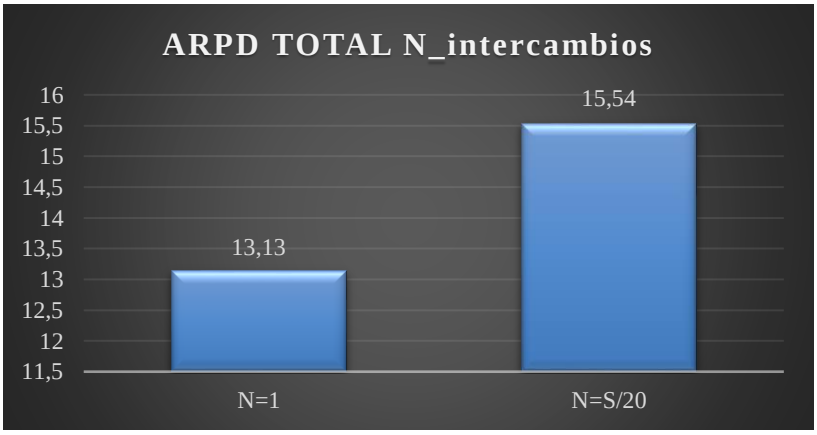
RPD	N=1	N=I/20
Datos13_1.txt	3.49	0.00
Datos13_2.txt	0.00	0.55
Datos13_3.txt	0.00	0.97
Datos13_4.txt	0.00	0.27
Datos13_5.txt	0.00	1.73
Datos13_6.txt	0.00	1.84
Datos13_7.txt	2.19	0.00
Datos13_8.txt	0.00	3.87
Datos13_9.txt	1.24	0.00
Datos13_10.txt	1.20	0.00
	0.81	0.92

RPD	N=1	N=I/20
Datos14_1.txt	0.00	1.73
Datos14_2.txt	1.29	0.00
Datos14_3.txt	0.64	0.00
Datos14_4.txt	0.00	1.42
Datos14_5.txt	0.00	2.10
Datos14_6.txt	0.00	1.16
Datos14_7.txt	0.77	0.00
Datos14_8.txt	0.00	2.73
Datos14_9.txt	2.58	0.00
Datos14_10.txt	0.51	0.00
	0.58	0.91

RPD	N=1	N=I/20
Datos15_1.txt	1.56	0.00
Datos15_2.txt	0.00	0.17
Datos15_3.txt	2.46	0.00
Datos15_4.txt	0.00	0.77
Datos15_5.txt	0.00	0.47
Datos15_6.txt	0.00	1.69
Datos15_7.txt	1.47	0.00
Datos15_8.txt	0.00	0.59
Datos15_9.txt	0.59	0.00
Datos15_10.txt	0.00	3.11
	0.61	0.68

RPD	N=1	N=I/20
Datos16_1.txt	2.93	0.00
Datos16_2.txt	0.00	0.55
Datos16_3.txt	0.00	0.97
Datos16_4.txt	0.00	0.27
Datos16_5.txt	0.00	1.73
Datos16_6.txt	0.00	1.84
Datos16_7.txt	2.19	0.00
Datos16_8.txt	0.00	3.87
Datos16_9.txt	1.24	0.00
Datos16_10.txt	1.20	0.00
	0.76	0.92





7.3 Tabla de Resultados

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos1_1.txt	40 de 49	5923	14	45 de 49	6685	15	44 de 49	6437	15	C_GS_FI
Datos1_2.txt	40 de 45	6270	15	40 de 45	6724	15	42 de 45	6839	15	C_RE
Datos1_3.txt	39 de 45	6333	15	41 de 45	6730	15	42 de 45	6806	15	C_RE
Datos1_4.txt	42 de 49	6235	15	45 de 49	6465	15	45 de 49	6733	15	C_RE
Datos1_5.txt	42 de 52	5940	15	46 de 52	6475	15	46 de 52	6556	15	C_RE
Datos1_6.txt	41 de 46	6464	15	43 de 46	6819	15	44 de 46	6919	15	C_RE
Datos1_7.txt	43 de 48	6330	15	44 de 48	6704	15	44 de 48	6632	15	C_GS_FI
Datos1_8.txt	43 de 52	6096	15	46 de 52	6600	15	45 de 52	6397	15	C_GS_FI
Datos1_9.txt	44 de 50	6328	15	44 de 50	6490	15	45 de 50	6476	15	C_GS_FI
Datos1_10.txt	39 de 48	5781	15	43 de 48	6651	15	45 de 48	6566	15	C_GS_FI
		61700			66343			66361		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos2_1.txt	41 de 49	5997	15	43 de 49	6444	15	42 de 49	6414	15	C_GS_FI
Datos2_2.txt	41 de 46	6340	15	43 de 46	6652	15	42 de 46	6896	15	C_RE
Datos2_3.txt	45 de 50	6333	15	47 de 50	6726	15	46 de 50	6563	15	C_GS_FI
Datos2_4.txt	44 de 51	6169	15	48 de 51	6759	15	45 de 51	6729	15	C_GS_FI
Datos2_5.txt	40 de 47	5923	15	44 de 47	6889	15	43 de 47	6597	15	C_GS_FI
Datos2_6.txt	37 de 44	6583	15	40 de 44	6728	15	39 de 44	6839	15	C_RE
Datos2_7.txt	45 de 51	6360	15	49 de 51	6840	15	49 de 51	6905	15	C_RE
Datos2_8.txt	38 de 46	6052	15	41 de 46	6444	15	40 de 46	6453	15	C_RE
Datos2_9.txt	40 de 45	6420	15	41 de 45	6644	15	41 de 45	6644	15	C_RE
Datos2_10.txt	41 de 49	6206	15	43 de 49	6487	15	42 de 49	6391	15	C_GS_FI
		62383			66613			66431		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos3_1.txt	42 de 50	6205	15	44 de 50	6302	15	43 de 50	6274	15	C_GS_FI
Datos3_2.txt	39 de 45	6274	15	41 de 45	6700	15	41 de 45	6598	15	C_GS_FI
Datos3_3.txt	43 de 47	6558	15	45 de 47	6912	15	43 de 47	6791	15	C_GS_FI
Datos3_4.txt	42 de 50	5875	15	44 de 50	6609	15	46 de 50	6504	15	C_GS_FI
Datos3_5.txt	42 de 49	6361	15	45 de 49	6661	15	46 de 49	6774	15	C_RE
Datos3_6.txt	48 de 53	6534	15	50 de 53	6972	15	49 de 53	6829	15	C_GS_FI
Datos3_7.txt	47 de 52	6610	15	49 de 52	6928	15	47 de 52	6881	15	C_GS_FI
Datos3_8.txt	38 de 43	6315	15	39 de 43	6731	15	39 de 43	6729	15	C_GS_FI
Datos3_9.txt	50 de 55	6302	15	50 de 55	6728	15	51 de 55	6689	15	C_GS_FI
Datos3_10.txt	42 de 47	6299	15	44 de 47	6655	15	44 de 47	6719	15	C_RE
		63333			67198			66788		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos4_1.txt	39 de 43	6549	15	40 de 43	6873	15	40 de 43	6925	15	C_RE
Datos4_2.txt	40 de 49	5939	15	44 de 49	6559	15	45 de 49	6654	15	C_RE
Datos4_3.txt	42 de 46	6456	15	43 de 46	6760	15	43 de 46	6637	15	C_GS_FI
Datos4_4.txt	41 de 48	6145	15	44 de 48	6650	15	44 de 48	6543	15	C_GS_FI
Datos4_5.txt	44 de 50	6320	15	45 de 50	6725	15	45 de 50	6730	15	C_RE
Datos4_6.txt	40 de 47	5960	15	43 de 47	6625	15	43 de 47	6558	15	C_GS_FI
Datos4_7.txt	46 de 51	6361	15	48 de 51	6858	15	47 de 51	6798	15	C_GS_FI
Datos4_8.txt	41 de 47	6478	15	43 de 47	6709	15	44 de 47	6764	15	C_RE
Datos4_9.txt	42 de 47	6368	15	45 de 47	6900	15	44 de 47	6708	15	C_GS_FI
Datos4_10.txt	42 de 49	6099	15	46 de 49	6710	15	45 de 49	6645	15	C_GS_FI
		62675			67369			66962		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos5_1.txt	40 de 52	5810	15	46 de 52	6783	15	46 de 52	6664	15	C_GS_FI
Datos5_2.txt	44 de 50	6522	15	46 de 50	6954	15	46 de 50	6768	15	C_GS_FI
Datos5_3.txt	42 de 53	6196	15	43 de 53	6500	15	47 de 53	6725	15	C_RE
Datos5_4.txt	38 de 44	6103	15	39 de 44	6344	15	39 de 44	6477	15	C_RE
Datos5_5.txt	43 de 49	6316	15	45 de 49	6700	15	45 de 49	6767	15	C_RE
Datos5_6.txt	40 de 48	6245	15	43 de 48	6437	15	42 de 48	6443	15	C_RE
Datos5_7.txt	42 de 49	6215	15	44 de 49	6517	15	44 de 49	6502	15	C_GS_FI
Datos5_8.txt	43 de 48	6363	15	44 de 48	6683	15	44 de 48	6686	15	C_RE
Datos5_9.txt	38 de 45	6148	15	40 de 45	6451	15	41 de 45	6663	15	C_RE
Datos5_10.txt	38 de 44	5949	15	40 de 44	6857	15	42 de 44	6757	15	C_GS_FI
		61867			66226			66452		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos6_1.txt	42 de 51	5888	14	43 de 51	6171	14	43 de 51	6158	14	C_GS_FI
Datos6_2.txt	45 de 51	6137	15	47 de 51	6528	15	46 de 51	6491	15	C_GS_FI
Datos6_3.txt	42 de 49	6106	15	45 de 49	6760	15	46 de 49	6767	15	C_RE
Datos6_4.txt	43 de 49	6543	15	46 de 49	6903	15	47 de 49	7002	15	C_RE
Datos6_5.txt	39 de 46	5924	15	40 de 46	6252	15	41 de 46	6428	15	C_RE
Datos6_6.txt	41 de 47	6164	15	43 de 47	6721	15	44 de 47	6690	15	C_GS_FI
Datos6_7.txt	39 de 47	6325	15	41 de 47	6446	15	41 de 47	6568	15	C_RE
Datos6_8.txt	44 de 48	6481	15	45 de 48	6611	15	45 de 48	6659	15	C_RE
Datos6_9.txt	45 de 53	6218	15	48 de 53	6794	15	49 de 53	6901	15	C_RE
Datos6_10.txt	43 de 49	6306	15	46 de 49	6566	15	45 de 49	6542	15	C_GS_FI
		62092			65752			66206		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos7_1.txt	48 de 53	6643	15	51 de 53	7045	15	49 de 53	6851	15	C_GS_FI
Datos7_2.txt	45 de 49	6501	15	46 de 49	6920	15	47 de 49	6990	15	C_RE
Datos7_3.txt	42 de 52	6291	15	48 de 52	6636	15	47 de 52	6576	15	C_GS_FI
Datos7_4.txt	38 de 47	5891	14	40 de 47	6160	14	40 de 47	6196	15	C_RE
Datos7_5.txt	42 de 48	6228	15	43 de 48	6697	15	42 de 48	6547	15	C_GS_FI
Datos7_6.txt	42 de 47	6303	15	44 de 47	6817	15	44 de 47	6720	15	C_GS_FI
Datos7_7.txt	46 de 51	6785	15	46 de 51	6856	15	46 de 51	6902	15	C_RE
Datos7_8.txt	50 de 53	6841	15	50 de 53	6921	15	50 de 53	6921	15	C_RE
Datos7_9.txt	39 de 45	6656	15	43 de 45	6904	15	41 de 45	6805	15	C_GS_FI
Datos7_10.txt	44 de 51	6281	15	46 de 51	6550	15	46 de 51	6703	15	C_RE
		64420			67506			67211		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	Surgeries_	W_assigne	ORs_used	MEJOR DECODING
Datos8_1.txt	43 de 48	6538	15	45 de 48	6936	15	45 de 48	6884	15	C_GS_FI
Datos8_2.txt	42 de 50	5892	15	44 de 50	6504	15	46 de 50	6689	15	C_RE
Datos8_3.txt	44 de 48	6586	15	46 de 48	6875	15	45 de 48	6838	15	C_GS_FI
Datos8_4.txt	44 de 47	6639	15	44 de 47	6885	15	45 de 47	6951	15	C_RE
Datos8_5.txt	44 de 51	6331	15	46 de 51	6654	15	47 de 51	6813	15	C_RE
Datos8_6.txt	41 de 48	6093	15	43 de 48	6394	15	42 de 48	6460	15	C_RE
Datos8_7.txt	34 de 43	5576	13	36 de 43	5916	13	35 de 43	5956	13	C_RE
Datos8_8.txt	43 de 51	5990	15	45 de 51	6388	15	44 de 51	6282	15	C_GS_FI
Datos8_9.txt	43 de 48	6532	15	45 de 48	6894	15	46 de 48	6953	15	C_RE
Datos8_10.txt	43 de 46	6605	15	44 de 46	6853	15	43 de 46	6792	15	C_GS_FI
		62782			66299			66618		
Capacidad ORs	7200									

	CB			C_GS_FI			C_RE			
	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	MEJOR DECODING
Datos9_1.txt	103 de 131	18072	45	119 de 131	20194	45	122 de 131	20585	45	C_RE
Datos9_2.txt	126 de 145	18264	45	130 de 145	19335	45	132 de 145	19412	45	C_RE
Datos9_3.txt	102 de 147	18277	45	123 de 147	19706	45	126 de 147	19723	45	C_RE
Datos9_4.txt	122 de 140	17999	45	129 de 140	19419	45	127 de 140	19476	45	C_RE
Datos9_5.txt	124 de 147	18238	45	136 de 147	19628	45	133 de 147	19433	45	C_GS_FI
Datos9_6.txt	97 de 128	18521	45	118 de 128	20313	45	120 de 128	20472	45	C_RE
Datos9_7.txt	123 de 142	18390	45	126 de 142	19138	45	131 de 142	19987	45	C_RE
Datos9_8.txt	101 de 142	18691	45	118 de 142	19845	45	121 de 142	19978	45	C_RE
Datos9_9.txt	131 de 152	18564	45	136 de 152	19175	45	140 de 152	19947	45	C_RE
Datos9_10.txt	100 de 139	18570	45	115 de 139	19887	45	123 de 139	20398	45	C_RE
		183586			196640			199411		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	MEJOR DECODING
Datos10_1.txt	95 de 137	18233	45	120 de 137	20221	45	118 de 137	20072	45	C_GS_FI
Datos10_2.txt	120 de 150	17186	45	134 de 150	19146	45	133 de 150	19280	45	C_RE
Datos10_3.txt	121 de 149	17384	45	131 de 149	18985	45	139 de 149	20081	45	C_RE
Datos10_4.txt	117 de 137	18219	45	120 de 137	19016	45	122 de 137	19907	45	C_RE
Datos10_5.txt	104 de 139	18743	45	123 de 139	19931	45	126 de 139	20214	45	C_RE
Datos10_6.txt	118 de 143	17654	45	126 de 143	19006	45	126 de 143	19038	45	C_RE
Datos10_7.txt	120 de 142	18302	45	126 de 142	19123	45	129 de 142	19693	45	C_RE
Datos10_8.txt	116 de 142	17731	45	127 de 142	19435	45	131 de 142	19415	45	C_GS_FI
Datos10_9.txt	111 de 153	18548	45	124 de 153	19815	45	133 de 153	19916	45	C_RE
Datos10_10.txt	108 de 141	18658	45	120 de 141	19851	45	130 de 141	20532	45	C_RE
		180658			194529			198148		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	MEJOR DECODING
Datos11_1.txt	131 de 151	18262	45	136 de 151	19278	45	139 de 151	19602	45	C_RE
Datos11_2.txt	118 de 141	17834	45	130 de 141	19803	45	129 de 141	19895	45	C_RE
Datos11_3.txt	121 de 150	17102	45	133 de 150	19268	45	141 de 150	20624	45	C_RE
Datos11_4.txt	127 de 150	18017	45	140 de 150	19775	45	139 de 150	19848	45	C_RE
Datos11_5.txt	128 de 150	18724	45	131 de 150	19335	45	139 de 150	20269	45	C_RE
Datos11_6.txt	127 de 147	18194	45	134 de 147	19441	45	137 de 147	20223	45	C_RE
Datos11_7.txt	123 de 146	18178	45	130 de 146	19427	45	131 de 146	19845	45	C_RE
Datos11_8.txt	121 de 143	18318	45	134 de 143	19988	45	133 de 143	19973	45	C_GS_FI
Datos11_9.txt	126 de 142	18682	45	129 de 142	19582	45	131 de 142	19914	45	C_RE
Datos11_10.txt	106 de 145	18564	45	122 de 145	19866	45	129 de 145	20297	45	C_RE
		181875			195763			200490		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_a	W_assigne	ORs_used	Surgeries_a	W_assigne	ORs_used	Surgeries_a	W_assigne	ORs_used	MEJOR DECODING
Datos12_1.txt	114 de 143	19187	45	126 de 143	20243	45	130 de 143	20519	45	C_RE
Datos12_2.txt	125 de 148	18347	45	135 de 148	19548	45	139 de 148	20425	45	C_RE
Datos12_3.txt	124 de 143	18373	45	131 de 143	19721	45	132 de 143	20331	45	C_RE
Datos12_4.txt	121 de 142	18708	45	129 de 142	19971	45	132 de 142	20328	45	C_RE
Datos12_5.txt	126 de 151	18013	45	137 de 151	19675	45	137 de 151	19697	45	C_RE
Datos12_6.txt	125 de 148	17635	45	134 de 148	19361	45	137 de 148	20215	45	C_RE
Datos12_7.txt	129 de 150	18456	45	135 de 150	19498	45	135 de 150	19907	45	C_RE
Datos12_8.txt	125 de 140	18555	45	133 de 140	20264	45	131 de 140	20094	45	C_GS_FI
Datos12_9.txt	120 de 140	18602	45	130 de 140	20316	45	133 de 140	20545	45	C_RE
Datos12_10.txt	126 de 144	18512	45	131 de 144	19531	45	133 de 144	19980	45	C_RE
		184388			198128			202041		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_a	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	MEJOR DECODING
Datos13_1.txt	99 de 138	18563	45	117 de 138	20184	45	122 de 138	20022	45	C_GS_FI
Datos13_2.txt	123 de 139	18653	45	129 de 139	19795	45	131 de 139	20088	45	C_RE
Datos13_3.txt	117 de 137	18458	45	126 de 137	19811	45	128 de 137	20095	45	C_RE
Datos13_4.txt	127 de 152	17759	45	135 de 152	19189	45	140 de 152	19694	45	C_RE
Datos13_5.txt	122 de 146	18012	45	131 de 146	19322	45	134 de 146	19740	45	C_RE
Datos13_6.txt	130 de 150	18275	45	136 de 150	19259	45	139 de 150	20002	45	C_RE
Datos13_7.txt	120 de 140	18329	45	125 de 140	19297	45	126 de 140	19441	45	C_RE
Datos13_8.txt	118 de 143	17589	45	132 de 143	19811	45	133 de 143	20070	45	C_RE
Datos13_9.txt	126 de 145	18421	45	128 de 145	19506	45	133 de 145	19800	45	C_RE
Datos13_10.txt	120 de 142	18126	45	129 de 142	19667	45	126 de 142	19314	45	C_GS_FI
		182185			195841			198266		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_a	W_assigne	ORs_used	Surgeries_a	W_assigne	ORs_used	Surgeries_as	W_assigne	ORs_used	MEJOR DECODING
Datos14_1.txt	109 de 151	18588	45	128 de 151	19732	45	133 de 151	20120	45	C_RE
Datos14_2.txt	108 de 138	18020	45	121 de 138	19572	45	122 de 138	19714	45	C_RE
Datos14_3.txt	120 de 144	17769	45	126 de 144	18893	45	132 de 144	19807	45	C_RE
Datos14_4.txt	122 de 140	18426	44	129 de 140	19767	45	129 de 140	19979	45	C_RE
Datos14_5.txt	121 de 143	18507	45	126 de 143	19347	45	127 de 143	20238	45	C_RE
Datos14_6.txt	105 de 148	18433	45	127 de 148	20180	45	130 de 148	20315	45	C_RE
Datos14_7.txt	100 de 144	18151	45	120 de 144	19904	45	124 de 144	19753	45	C_GS_FI
Datos14_8.txt	116 de 138	18259	45	127 de 138	19857	45	126 de 138	19849	45	C_GS_FI
Datos14_9.txt	119 de 143	17771	45	130 de 143	19401	45	129 de 143	19621	45	C_RE
Datos14_10.txt	102 de 136	18935	45	115 de 136	19777	45	118 de 136	19808	45	C_RE
		182859			196430			199204		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_a	W_assigne	ORs_used	Surgeries_a	W_assigne	ORs_used	Surgeries_a	W_assigne	ORs_used	MEJOR DECODING
Datos15_1.txt	119 de 141	17966	45	129 de 141	19589	45	129 de 141	19790	45	C_RE
Datos15_2.txt	124 de 145	17945	45	133 de 145	19497	45	130 de 145	19775	45	C_RE
Datos15_3.txt	119 de 138	18296	45	123 de 138	19016	45	127 de 138	19744	45	C_RE
Datos15_4.txt	127 de 155	18114	45	136 de 155	19019	45	137 de 155	19629	45	C_RE
Datos15_5.txt	111 de 164	17875	45	130 de 164	19486	45	137 de 164	19800	45	C_RE
Datos15_6.txt	109 de 146	18620	45	128 de 146	20189	45	133 de 146	20461	45	C_RE
Datos15_7.txt	113 de 153	18531	45	129 de 153	20146	45	127 de 153	19773	45	C_GS_FI
Datos15_8.txt	130 de 150	18690	45	135 de 150	19421	45	139 de 150	20069	45	C_RE
Datos15_9.txt	123 de 142	18703	45	128 de 142	19301	45	131 de 142	19990	45	C_RE
Datos15_10.txt	107 de 143	18763	45	125 de 143	20477	45	132 de 143	20627	45	C_RE
		183503			196141			199658		
Capacidad ORs	21600									

	CB			C_GS_FI			C_RE			
	Surgeries_as	W_assigne	ORs_used	Surgeries_a	W_assigne	ORs_used	Surgeries_a:	W_assigne	ORs_used	MEJOR DECODING
Datos16_1.txt	127 de 148	18521	45	132 de 148	19528	45	137 de 148	19877	45	C_RE
Datos16_2.txt	103 de 145	18538	45	117 de 145	19744	45	126 de 145	20344	45	C_RE
Datos16_3.txt	124 de 144	18305	45	134 de 144	20202	45	131 de 144	19709	45	C_GS_FI
Datos16_4.txt	114 de 136	17800	45	124 de 136	19583	45	122 de 136	19778	45	C_RE
Datos16_5.txt	98 de 133	18610	45	117 de 133	19961	45	120 de 133	20084	45	C_RE
Datos16_6.txt	110 de 151	18662	45	132 de 151	20236	45	129 de 151	20030	45	C_GS_FI
Datos16_7.txt	123 de 144	18324	45	132 de 144	19975	45	131 de 144	20110	45	C_RE
Datos16_8.txt	112 de 149	18843	45	127 de 149	20345	45	126 de 149	19747	45	C_GS_FI
Datos16_9.txt	125 de 148	18448	45	135 de 148	19843	45	138 de 148	20154	45	C_RE
Datos16_10.txt	118 de 141	18353	45	126 de 141	19625	45	126 de 141	19747	45	C_RE
		184404			199042			199580		
Capacidad ORs	21600									

7.4 Resultados RPD

RPD	CB	C_GS_FI	C_RE
Datos1_1.txt	11.40	0.00	3.71
Datos1_2.txt	8.32	1.68	0.00
Datos1_3.txt	6.95	1.12	0.00
Datos1_4.txt	7.40	3.98	0.00
Datos1_5.txt	9.40	1.24	0.00
Datos1_6.txt	6.58	1.45	0.00
Datos1_7.txt	5.58	0.00	1.07
Datos1_8.txt	7.64	0.00	3.08
Datos1_9.txt	2.50	0.00	0.22
Datos1_10.txt	13.08	0.00	1.28
TOTAL D1	7.88	0.95	0.94

RPD	CB	C_GS_FI	C_RE
Datos2_1.txt	6.94	0.00	0.47
Datos2_2.txt	8.06	3.54	0.00
Datos2_3.txt	5.84	0.00	2.42
Datos2_4.txt	8.73	0.00	0.44
Datos2_5.txt	14.02	0.00	4.24
Datos2_6.txt	3.74	1.62	0.00
Datos2_7.txt	7.89	0.94	0.00
Datos2_8.txt	6.21	0.14	0.00
Datos2_9.txt	3.37	0.00	0.00
Datos2_10.txt	4.33	0.00	1.48
TOTAL D2	6.91	0.62	0.91

RPD	CB	C_GS_FI	C_RE
Datos3_1.txt	1.54	0.00	0.44
Datos3_2.txt	6.36	0.00	1.52
Datos3_3.txt	5.12	0.00	1.75
Datos3_4.txt	11.11	0.00	1.59
Datos3_5.txt	6.10	1.67	0.00
Datos3_6.txt	6.28	0.00	2.05
Datos3_7.txt	4.59	0.00	0.68
Datos3_8.txt	6.18	0.00	0.03
Datos3_9.txt	6.33	0.00	0.58
Datos3_10.txt	6.25	0.95	0.00
TOTAL D3	5.99	0.26	0.86

RPD	CB	C_GS_FI	C_RE
Datos4_1.txt	5.43	0.75	0.00
Datos4_2.txt	10.75	1.43	0.00
Datos4_3.txt	4.50	0.00	1.82
Datos4_4.txt	7.59	0.00	1.61
Datos4_5.txt	6.09	0.07	0.00
Datos4_6.txt	10.04	0.00	1.01
Datos4_7.txt	7.25	0.00	0.87
Datos4_8.txt	4.23	0.81	0.00
Datos4_9.txt	7.71	0.00	2.78
Datos4_10.txt	9.11	0.00	0.97
TOTAL D4	7.27	0.31	0.91

RPD	CB	C_GS_FI	C_RE
Datos5_1.txt	14.34	0.00	1.75
Datos5_2.txt	6.21	0.00	2.67
Datos5_3.txt	7.87	3.35	0.00
Datos5_4.txt	5.77	2.05	0.00
Datos5_5.txt	6.66	0.99	0.00
Datos5_6.txt	3.07	0.09	0.00
Datos5_7.txt	4.63	0.00	0.23
Datos5_8.txt	4.83	0.04	0.00
Datos5_9.txt	7.73	3.18	0.00
Datos5_10.txt	13.24	0.00	1.46
TOTAL D5	7.44	0.97	0.61

RPD	CB	C_GS_FI	C_RE
Datos6_1.txt	4.59	0.00	0.21
Datos6_2.txt	5.99	0.00	0.57
Datos6_3.txt	9.77	0.10	0.00
Datos6_4.txt	6.56	1.41	0.00
Datos6_5.txt	7.84	2.74	0.00
Datos6_6.txt	8.29	0.00	0.46
Datos6_7.txt	3.70	1.86	0.00
Datos6_8.txt	2.67	0.72	0.00
Datos6_9.txt	9.90	1.55	0.00
Datos6_10.txt	3.96	0.00	0.37
TOTAL D6	6.33	0.84	0.16

RPD	CB	C_GS_FI	C_RE
Datos7_1.txt	5.71	0.00	2.75
Datos7_2.txt	7.00	1.00	0.00
Datos7_3.txt	5.20	0.00	0.90
Datos7_4.txt	4.92	0.58	0.00
Datos7_5.txt	7.00	0.00	2.24
Datos7_6.txt	7.54	0.00	1.42
Datos7_7.txt	1.70	0.67	0.00
Datos7_8.txt	1.16	0.00	0.00
Datos7_9.txt	3.59	0.00	1.43
Datos7_10.txt	6.30	2.28	0.00
TOTAL D7	5.01	0.45	0.88

RPD	CB	C_GS_FI	C_RE
Datos8_1.txt	5.74	0.00	0.75
Datos8_2.txt	11.92	2.77	0.00
Datos8_3.txt	4.20	0.00	0.54
Datos8_4.txt	4.49	0.95	0.00
Datos8_5.txt	7.07	2.33	0.00
Datos8_6.txt	5.68	1.02	0.00
Datos8_7.txt	6.38	0.67	0.00
Datos8_8.txt	6.23	0.00	1.66
Datos8_9.txt	6.05	0.85	0.00
Datos8_10.txt	3.62	0.00	0.89
TOTAL D8	6.14	0.86	0.38

RPD	CB	C_GS_FI	C_RE
Datos9_1.txt	12.21	1.90	0.00
Datos9_2.txt	5.91	0.40	0.00
Datos9_3.txt	7.33	0.09	0.00
Datos9_4.txt	7.58	0.29	0.00
Datos9_5.txt	7.08	0.00	0.99
Datos9_6.txt	9.53	0.78	0.00
Datos9_7.txt	7.99	4.25	0.00
Datos9_8.txt	6.44	0.67	0.00
Datos9_9.txt	6.93	3.87	0.00
Datos9_10.txt	8.96	2.51	0.00
TOTAL D9	8.00	1.47	0.10

RPD	CB	C_GS_FI	C_RE
Datos10_1.txt	9.83	0.00	0.74
Datos10_2.txt	10.86	0.70	0.00
Datos10_3.txt	13.43	5.46	0.00
Datos10_4.txt	8.48	4.48	0.00
Datos10_5.txt	7.28	1.40	0.00
Datos10_6.txt	7.27	0.17	0.00
Datos10_7.txt	7.06	2.89	0.00
Datos10_8.txt	8.77	0.00	0.10
Datos10_9.txt	6.87	0.51	0.00
Datos10_10.txt	9.13	3.32	0.00
TOTAL D10	8.90	1.89	0.08

RPD	CB	C_GS_FI	C_RE
Datos11_1.txt	6.84	1.65	0.00
Datos11_2.txt	10.36	0.46	0.00
Datos11_3.txt	17.08	6.57	0.00
Datos11_4.txt	9.23	0.37	0.00
Datos11_5.txt	7.62	4.61	0.00
Datos11_6.txt	10.03	3.87	0.00
Datos11_7.txt	8.40	2.11	0.00
Datos11_8.txt	8.36	0.00	0.08
Datos11_9.txt	6.19	1.67	0.00
Datos11_10.txt	8.54	2.12	0.00
TOTAL D11	9.26	2.34	0.01

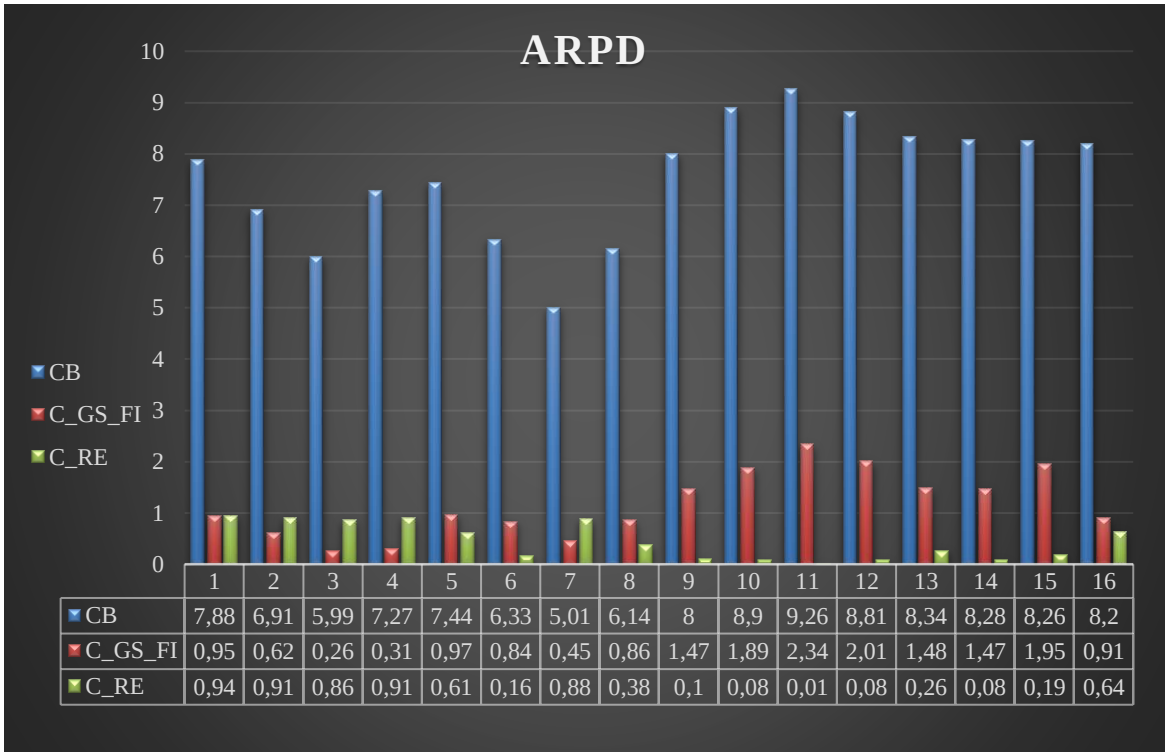
RPD	CB	C_GS_FI	C_RE
Datos12_1.txt	6.49	1.35	0.00
Datos12_2.txt	10.17	4.29	0.00
Datos12_3.txt	9.63	3.00	0.00
Datos12_4.txt	7.97	1.76	0.00
Datos12_5.txt	8.55	0.11	0.00
Datos12_6.txt	12.76	4.22	0.00
Datos12_7.txt	7.29	2.05	0.00
Datos12_8.txt	8.43	0.00	0.84
Datos12_9.txt	9.46	1.11	0.00
Datos12_10.txt	7.35	2.25	0.00
TOTAL D12	8.81	2.01	0.08

RPD	CB	C_GS_FI	C_RE
Datos13_1.txt	8.03	0.00	0.80
Datos13_2.txt	7.14	1.46	0.00
Datos13_3.txt	8.15	1.41	0.00
Datos13_4.txt	9.83	2.56	0.00
Datos13_5.txt	8.75	2.12	0.00
Datos13_6.txt	8.63	3.71	0.00
Datos13_7.txt	5.72	0.74	0.00
Datos13_8.txt	12.36	1.29	0.00
Datos13_9.txt	6.96	1.48	0.00
Datos13_10.txt	7.84	0.00	1.79
TOTAL D13	8.34	1.48	0.26

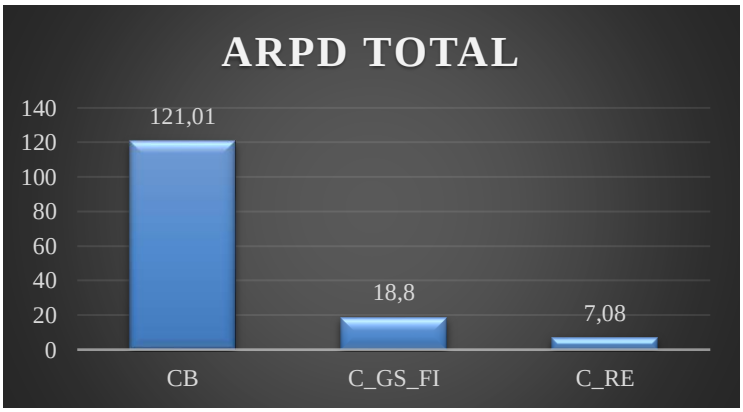
RPD	CB	C_GS_FI	C_RE
Datos14_1.txt	7.61	1.93	0.00
Datos14_2.txt	8.59	0.72	0.00
Datos14_3.txt	10.29	4.61	0.00
Datos14_4.txt	7.77	1.06	0.00
Datos14_5.txt	8.55	4.40	0.00
Datos14_6.txt	9.26	0.66	0.00
Datos14_7.txt	8.81	0.00	0.76
Datos14_8.txt	8.05	0.00	0.04
Datos14_9.txt	9.43	1.12	0.00
Datos14_10.txt	4.41	0.16	0.00
TOTAL D14	8.28	1.47	0.08

RPD	CB	C_GS_FI	C_RE
Datos15_1.txt	9.22	1.02	0.00
Datos15_2.txt	9.25	1.41	0.00
Datos15_3.txt	7.33	3.69	0.00
Datos15_4.txt	7.72	3.11	0.00
Datos15_5.txt	9.72	1.59	0.00
Datos15_6.txt	9.00	1.33	0.00
Datos15_7.txt	8.02	0.00	1.85
Datos15_8.txt	6.87	3.23	0.00
Datos15_9.txt	6.44	3.45	0.00
Datos15_10.txt	9.04	0.73	0.00
TOTAL D15	8.26	1.95	0.19

RPD	CB	C_GS_FI	C_RE
Datos16_1.txt	6.82	1.76	0.00
Datos16_2.txt	8.88	2.95	0.00
Datos16_3.txt	9.39	0.00	2.44
Datos16_4.txt	10.00	0.99	0.00
Datos16_5.txt	7.34	0.61	0.00
Datos16_6.txt	7.78	0.00	1.02
Datos16_7.txt	8.88	0.67	0.00
Datos16_8.txt	7.38	0.00	2.94
Datos16_9.txt	8.46	1.54	0.00
Datos16_10.txt	7.06	0.62	0.00
TOTAL D16	8.20	0.91	0.64



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CB	7,88	6,91	5,99	7,27	7,44	6,33	5,01	6,14	8	8,9	9,26	8,81	8,34	8,28	8,26	8,2
C_GS_FI	0,95	0,62	0,26	0,31	0,97	0,84	0,45	0,86	1,47	1,89	2,34	2,01	1,48	1,47	1,95	0,91
C_RE	0,94	0,91	0,86	0,91	0,61	0,16	0,88	0,38	0,1	0,08	0,01	0,08	0,26	0,08	0,19	0,64



7.5 PYTHON. Generador de Instancias

```
# -*- coding: utf-8 -*-
"""
TFG - OPTIMIZACION QUIRURGICA MAXIMIZANDO LA UTILIDAD DE LOS RECURSOS
    MEDIANTE EL ALGORITMO CUCKOO SEARCH

@author: AlejandroPina
"""
import math
import random

random.seed(123)



#Número de días
H=5

#Nº de ORs
J= [3, 9]

#Porcentaje de sobre tiempo
beta=[100, 125]

#Porcentaje para cirujanos
alpha= [1.50, 2.00]

#t_surgeries
mu=[60, 120, 180, 240]
sigma=[0.1, 0.2, 0.3, 0.4, 0.5]

#Disponibilidad del cirujano
av=8*60 #8 horas
l=1 #Semanas de horizonte
mds=[3,4]
r=av
#rjh=[[a for h in range H] for j in range J]
```

```
problema=1
for j in J:
    for b in beta:
        for p in alpha:
            for m in mds:
                for z in range(10):
                    #Calculamos nº de Surgeons
                    ttotal=r*j*H
                    Surgeons=math.floor((p*ttotal/(l*av*m)))

                    #Límite de movimientos de quirófano por cirujano
                    U=[]
                    for s in range (Surgeons):
                        U.append(j)

                    #Surgeries
                    t_surgeries=[]
                    while (sum(t_surgeries)<ttotal):
                        #Debemos realizar una conversión de los datos a escala logaritmica
                        mu_i=random.choice(mu)
                        sigma_i=random.choice(sigma)*mu_i
                        log_mu_i= math.log(mu_i)
                        CV= sigma_i/mu_i
                        log_sigma_i= math.log(math.sqrt((CV**2)+1))
                        t_surgeries.append(int(random.lognormvariate(log_mu_i, log_sigma_i)))
                    Surgeries= len(t_surgeries)

                    archivo_dato=f'Datoss{problema}_{z+1}.txt'
                    with open (archivo_dato,'w') as file:

                        file.write(f'[Horizon={H}]\n')
                        file.write(f'[ORs={j}]\n')
                        file.write(f'[Cap_OR={r}]\n')
                        file.write(f'[Surgeries={Surgeries}]\n')
                        file.write('[t_surgeries='+','.join(str(x) for x in t_surgeries)+ ']\n')
                        file.write(f'[Surgeons={Surgeons}]\n')
```

```
file.write(f'[Cap_S={av}]\n')
file.write(f'[mds={m}]\n')
file.write('[Lim_OR='+','.join(str(x) for x in U)+ ']\n')

#Peso de la operación
w=[]
for i in range (Surgeries):
   w.append(random.randint(1, 5))
file.write('[W='+','.join(str(x) for x in w) + "]\n")

#Cirujano encargado de la cirugía 'i'
gamma=[]



for i in range (Surgeries):
   maxim=random.randint(1,Surgeons)
   aux=[0 for s in range(Surgeons)]     #vector auxiliar que nos dice el cirujano de un paciente
   for _ in range(maxim):
      a=random.randint(0, Surgeons-1)         #nos dice la posición (por eso -1)
      while(aux[a]==1):
         a=random.randint(0, Surgeons-1)
      aux[a]=1
   gamma.append(aux)

gamma_str = ''.join(','.join(map(str, sublist)) + ';' for sublist in gamma)  # Genera la cadena incluyendo todos los puntos y comas
gamma_str = gamma_str[:-1]  #Me genera un ; final que no quiero, lo elimino quedándome hasta el penúltimo
file.write('[S_encargado=' + gamma_str + ']\n')

#Release y due date
rd=[]
dd=[]
for i in range (Surgeries):
   rd.append(random.randint(0, int(0.6*H)))
   dd.append(random.randint(int(0.5*H),int(0.75*H)))
file.write('[rd='+','.join(str(x) for x in rd)+ ']\n')
file.write('[dd='+','.join(str(x) for x in dd)+ ']\n')
```

```
        problema+=1
```

7.6 PYTHON. Funciones

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 25 18:05:22 2024

@author: AlejandroPina
"""

import random
import math
from scheptk.util import read_tag
import numpy as np
import matplotlib.pyplot as plt
import copy
from scheptk.util import sorted_index_asc, sorted_index_desc



#3.exponente en funcion objetivo político
#excel de resultado (CUANDO ACABE EL TRABAJO FINAL)
#levy flight
def Leer_datos(filename):
    Horizon = read_tag(filename, 'Horizon')
    ORs = read_tag(filename, 'ORs')
    Surgeries = read_tag(filename, 'Surgeries')
    Surgeons = read_tag(filename, 'Surgeons')
    mds = read_tag(filename, 'mds')
    Cap_S = read_tag(filename, 'Cap_S')
    Cap_OR = read_tag(filename, 'Cap_OR')
    Lim_OR = read_tag(filename, 'Lim_OR')
    t_surgeries = read_tag(filename, 't_surgeries')
    W = read_tag(filename, 'W')
    gamma = read_tag(filename, 'S_encargado')
    rd = read_tag(filename, 'rd')
    dd = read_tag(filename, 'dd')
```

```
    return Horizon, ORs, Surgeries, Surgeons, mds, Cap_S, Cap_OR, Lim_OR, t_surgeries, W, gamma, rd, dd


#Me dice cuantas cirugías han sido asignadas
def Surgeries_assigned(Surgeries, ORs, Horizon, Xijh):
    valor=0
    for i in range(len(Xijh)):
        for j in range(ORs):
            for h in range (Horizon):
                if(Xijh[i][j][h]==1):
                    valor+=1
    return valor


#Me genera una solución aleatoria
def random_solution(Surgeries):
    PI=[x for x in range (Surgeries)]
    random.shuffle(PI)
    return PI


#
==========================================================================================
# DECODING
#
==========================================================================================


#Función que asigna hora, dia, quirofano y cirujano en el primero que quepa
def assign_first (Horizon, ORs, Surgeons, PI, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds):
    Xijh = [[[0 for _ in range(Horizon)] for _ in range(ORs)] for _ in range(len(PI))]     #1 si i es operado en j el día h
    Yis = [[0 for _ in range(Surgeons)] for _ in range(len(PI))]                           #1 si el la cirugía i la hace el cirujano s
    Zsjh = [[[0 for _ in range(Horizon)] for _ in range(ORs)] for _ in range(Surgeons)]    #1 si el s opera en j el día h
#Para controlar mds, no tener siempre que recorrer Xijh, si no una variable auxiliar que requiera menos cálculos
    OpDays_s = [[0 for _ in range(Horizon)]for _ in range(Surgeons)]
#Para llevar el control de los quirófanos que ha operado tengo que llevar el conteo de quirofanos operado y los distintos
    ORs_Surgeon_shj = [[[0 for _ in range (ORs)] for _ in range (Horizon)] for _ in range(Surgeons)]
```

```
Last_OR_shj = [[[0 for _ in range (ORs)] for _ in range (Horizon)] for _ in range (Surgeons)]

#3.1 Declaración de completion time
   #Completion time de quirófano
ct_ORs = [[0 for _ in range(Horizon)] for _ in range(ORs)]
   #Completion time de Surgeon
ct_surgeons = [[0 for _ in range(Horizon)] for _ in range(Surgeons)]
   #Hora final de la cirugía i en el quirofano j el día h
ct_surgeries = [0 for _ in range(len(PI))]



"""El cirujano no puede más de 6 horas, y el quirofano hasta 10"""
#3.3 Asignación al principio
#3.3.1 Asignación de la primera operación
asignacion_hecha=False #Para salir del bucle

for h in range(rd[PI[0]], Horizon):
   for j in range(ORs):
      if (delta_ijh[PI[0]][j][h] == 1):    #Mira que se pueda realizar la cirugía
         for s in range(Surgeons):
            if gamma[PI[0]][s] == 1 and sum(OpDays_s[s]) < mds and
sum(ORs_Surgeon_shj[s][h])<Lim_OR[s]:  #Mira que el cirujano sea apto y que no haya operado más de los
días
               #Asigno
               Xijh[PI[0]][j][h] = 1
               Yis[PI[0]][s] = 1
               Zsjh[s][j][h] = 1

               #Control de LimOR por día (u)
               ORs_Surgeon_shj[s][h][j] = 1
               Last_OR_shj[s][h][j] = 1
               #Control de MDS
               OpDays_s[s][h]=1

               ct_ORs[j][h] = t_surgeries[PI[0]]
               ct_surgeons[s][h] = t_surgeries[PI[0]]
               ct_surgeries[PI[0]] = t_surgeries[PI[0]]
```

```
                break

            asignacion_hecha = True
            break
        if asignacion_hecha:
            break

    #3.3.2 Asignación de las siguientes operaciones
    for i in range(1, len(PI)):
        asignacion_hecha = False  #Para salir del bucle
        for h in range(rd[PI[i]], Horizon):
            for j in range(ORs):
                if (delta_ijh[PI[i]][j][h] == 1 and ct_ORs[j][h] + t_surgeries[PI[i]] <= Cap_OR):  #Comprueba que se puede ese dia, or  y que no excede la capacidad
                    for s in range(Surgeons):
                        if (gamma[PI[i]][s] == 1 and (sum(OpDays_s[s]) < mds or OpDays_s[s][h]==1) and (sum(ORs_Surgeon_shj[s][h])<Lim_OR[s] or Last_OR_shj[s][h][j]==1)):  #Se contempla el lim de dias y cambios del cirujano
                            if (t_surgeries[PI[i]] + ct_surgeons[s][h] <= Cap_S):  #Compruebo que no excede la capacidad
                                Xijh[PI[i]][j][h] = 1   #Asigno a este día y quirofano y actualizo fecha
                                Yis[PI[i]][s] = 1
                                Zsjh[s][j][h] = 1

                                if(ORs_Surgeon_shj[s][h][j] == 0): #Opera en un nuevo quirófano // Si opera en un mismo quirófano, no hace falta
                                    ORs_Surgeon_shj[s][h][j] = 1    #Control de nº de quirófanos
                                    for k in range(ORs):
                                        Last_OR_shj[s][h][k] = 0    #Pone en 0 el último quirófano asignado
                                    Last_OR_shj[s][h][j]=1          #Pone en 1 el quirófano nuevo

                                if(OpDays_s[s][h]==0): #Se asigna que ese día opera
                                    OpDays_s[s][h]=1

                                asignacion_hecha = True
                                if (ct_ORs[j][h] >= ct_surgeons[s][h]):
                                    ct_ORs[j][h] += t_surgeries[PI[i]]
                                    ct_surgeons[s][h] = ct_ORs[j][h]
                                    ct_surgeries[PI[i]] = ct_ORs[j][h]
```

```
                else:
                    ct_surgeons[s][h] += t_surgeries[PI[i]]
                    ct_ORs[j][h] = ct_surgeons[s][h]
                    ct_surgeries[PI[i]] = ct_surgeons[s][h]


                break

        if asignacion_hecha:
            break
    if asignacion_hecha:
        break


return Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries


def assign_overloaded (Horizon, ORs, Surgeons, PI, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds):
    Xijh = [[[0 for _ in range(Horizon)] for _ in range(ORs)] for _ in range(len(PI))]     #1 si i es operado en j el día h
    Yis = [[0 for _ in range(Surgeons)] for _ in range(len(PI))]                           #1 si el la cirugía i la hace el cirujano s
    Zsjh = [[[0 for _ in range(Horizon)] for _ in range(ORs)] for _ in range(Surgeons)]    #1 si el s opera en j el día h
#Para controlar mds, no tener siempre que recorrer Xijh, si no una variable auxiliar que requiera menos cálculos
    OpDays_s = [[0 for _ in range(Horizon)]for _ in range(Surgeons)]
#Para llevar el control de los quirófanos que ha operado tengo que llevar el conteo de quirofanos operado y los distintos
    ORs_Surgeon_shj = [[[0 for _ in range (ORs)] for _ in range (Horizon)] for _ in range(Surgeons)]
    Last_OR_shj = [[[0 for _ in range (ORs)] for _ in range (Horizon)] for _ in range (Surgeons)]


    #3.1 Declaración de completion time
        #Completion time de quirófano
    ct_ORs = [[0 for _ in range(Horizon)] for _ in range(ORs)]
        #Completion time de Surgeon
    ct_surgeons = [[0 for _ in range(Horizon)] for _ in range(Surgeons)]
        #Hora final de la cirugía i en el quirofano j el día h
```

```
ct_surgeries = [0 for _ in range(len(PI))]
    #Tiempo del quirófano (establecerá el orden en el que se deban asignar)
OR_time=[[0 for j in range (ORs)] for h in range (Horizon)]



"""El cirujano no puede más de 6 horas, y el quirofano hasta 10"""
#3.3 Asignación al principio
#3.3.1 Asignación de la primera operación
asignacion_hecha=False #Para salir del bucle

for h in range(rd[PI[0]], Horizon):
    for j in range(ORs):
        if (delta_ijh[PI[0]][j][h] == 1):
            for s in range(Surgeons):
                if gamma[PI[0]][s] == 1 and sum(OpDays_s[s]) < mds and
sum(ORs_Surgeon_shj[s][h])<Lim_OR[s]:
                    Xijh[PI[0]][j][h] = 1
                    Yis[PI[0]][s] = 1
                    Zsjh[s][j][h] = 1

                    #Control de LimOR por día (u)
                    ORs_Surgeon_shj[s][h][j] = 1
                    Last_OR_shj[s][h][j] = 1
                    #Control de MDS
                    OpDays_s[s][h]=1

                    ct_ORs[j][h]= t_surgeries[PI[0]]
                    ct_surgeons[s][h]= t_surgeries[PI[0]]
                    ct_surgeries[PI[0]]= t_surgeries[PI[0]]
                    OR_time[h][j]= t_surgeries[PI[0]]
                    break

            asignacion_hecha = True
            break
        if asignacion_hecha:
            break
```

```
    #3.3.2 Asignación de las siguientes operaciones donde más tiempo de quirófano haya
  for i in range(1, len(PI)):
    asignacion_hecha = False  #Para salir del bucle
    for h in range(rd[PI[i]], Horizon):
      OR_order_time=sorted_index_desc(OR_time[h])
      for j in (OR_order_time):
        if (delta_ijh[PI[i]][j][h] == 1 and ct_ORs[j][h] + t_surgeries[PI[i]] <= Cap_OR):  #Comprueba que se puede ese dia, or  y que no excede la capacidad
          for s in range(Surgeons):
            if (gamma[PI[i]][s] == 1 and (sum(OpDays_s[s]) < mds or OpDays_s[s][h]==1) and (sum(ORs_Surgeon_shj[s][h])<Lim_OR[s] or Last_OR_shj[s][h][j]==1)):  #Se contempla el lim de dias y cambios del cirujano
              if (t_surgeries[PI[i]] + ct_surgeons[s][h] <= Cap_S):  #Compruebo que no excede la capacidad
                Xijh[PI[i]][j][h] = 1   #Asigno a este día y quirofano y actualizo fecha
                Yis[PI[i]][s] = 1
                Zsjh[s][j][h] = 1

                if(ORs_Surgeon_shj[s][h][j] == 0): #Opera en un nuevo quirófano // Si opera en un mismo quirófano, no hace falta
                  ORs_Surgeon_shj[s][h][j] = 1    #Control de nº de quirófanos
                  for k in range(ORs):
                    Last_OR_shj[s][h][k] = 0    #Pone en 0 el último quirófano asignado
                  Last_OR_shj[s][h][j]=1         #Pone en 1 el quirófano nuevo

                if(OpDays_s[s][h]==0): #Se asigna que ese día opera
                  OpDays_s[s][h]=1

                asignacion_hecha = True
                if (ct_ORs[j][h] >= ct_surgeons[s][h]):
                  ct_ORs[j][h] += t_surgeries[PI[i]]
                  ct_surgeons[s][h] = ct_ORs[j][h]
                  ct_surgeries[PI[i]] = ct_ORs[j][h]
                  OR_time[h][j]=ct_ORs[j][h]

                else:
                  ct_surgeons[s][h] += t_surgeries[PI[i]]
                  ct_ORs[j][h] = ct_surgeons[s][h]
                  ct_surgeries[PI[i]] = ct_surgeons[s][h]
```

```
                        OR_time[h][j]=ct_surgeons[s][h]

                    break

            if asignacion_hecha:
                break
        if asignacion_hecha:
            break

    return Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries


def assign_underloaded (Horizon, ORs, Surgeons, PI, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds):
    Xijh = [[[0 for _ in range(Horizon)] for _ in range(ORs)] for _ in range(len(PI))]     #1 si i es operado en j el día h
    Yis = [[0 for _ in range(Surgeons)] for _ in range(len(PI))]                           #1 si el la cirugía i la hace el cirujano s
    Zsjh = [[[0 for _ in range(Horizon)] for _ in range(ORs)] for _ in range(Surgeons)]    #1 si el s opera en j el día h
#Para controlar mds, no tener siempre que recorrer Xijh, si no una variable auxiliar que requiera menos cálculos
    OpDays_s = [[0 for _ in range(Horizon)]for _ in range(Surgeons)]
#Para llevar el control de los quirófanos que ha operado tengo que llevar el conteo de quirofanos operado y los distintos
    ORs_Surgeon_shj = [[[0 for _ in range (ORs)] for _ in range (Horizon)] for _ in range(Surgeons)]
    Last_OR_shj = [[[0 for _ in range (ORs)] for _ in range (Horizon)] for _ in range (Surgeons)]


    #3.1 Declaración de completion time
        #Completion time de quirófano
    ct_ORs = [[0 for _ in range(Horizon)] for _ in range(ORs)]
        #Completion time de Surgeon
    ct_surgeons = [[0 for _ in range(Horizon)] for _ in range(Surgeons)]
        #Hora final de la cirugía i en el quirofano j el día h
    ct_surgeries = [0 for _ in range(len(PI))]
        #Tiempo del quirófano (establecerá el orden en el que se deban asignar)
    OR_time=[[0 for j in range (ORs)] for h in range (Horizon)]
```

```
    """El cirujano no puede más de 6 horas, y el quirofano hasta 10"""
    #3.3 Asignación al principio
    #3.3.1 Asignación de la primera operación
    asignacion_hecha=False #Para salir del bucle

    for h in range(rd[PI[0]], Horizon):
        for j in range(ORs):
            if (delta_ijh[PI[0]][j][h] == 1):
                for s in range(Surgeons):
                    if gamma[PI[0]][s] == 1 and sum(OpDays_s[s]) < mds and
sum(ORs_Surgeon_shj[s][h])<Lim_OR[s]:
                        Xijh[PI[0]][j][h] = 1
                        Yis[PI[0]][s] = 1
                        Zsjh[s][j][h] = 1

                        #Control de LimOR por día (u)
                        ORs_Surgeon_shj[s][h][j] = 1
                        Last_OR_shj[s][h][j] = 1
                        #Control de MDS
                        OpDays_s[s][h]=1

                        ct_ORs[j][h]= t_surgeries[PI[0]]
                        ct_surgeons[s][h]= t_surgeries[PI[0]]
                        ct_surgeries[PI[0]]= t_surgeries[PI[0]]
                        OR_time[h][j]= t_surgeries[PI[0]] #Controla el tiempo del OR
                        break

                asignacion_hecha = True
                break
        if asignacion_hecha:
            break

        #3.3.2 Asignación de las siguientes operaciones donde más tiempo de quirófano haya
    for i in range(1, len(PI)):
        asignacion_hecha = False  #Para salir del bucle
        for h in range(rd[PI[i]], Horizon):
```

```
OR_order_time=sorted_index_asc(OR_time[h])
for j in (OR_order_time):
  if (delta_ijh[PI[i]][j][h] == 1 and ct_ORs[j][h] + t_surgeries[PI[i]] <= Cap_OR):  #Comprueba que se puede ese dia, or  y que no excede la capacidad
    for s in range(Surgeons):
      if (gamma[PI[i]][s] == 1 and (sum(OpDays_s[s]) < mds or OpDays_s[s][h]==1) and (sum(ORs_Surgeon_shj[s][h])<Lim_OR[s] or Last_OR_shj[s][h][j]==1)):  #Se contempla el lim de dias y cambios del cirujano
        if (t_surgeries[PI[i]] + ct_surgeons[s][h] <= Cap_S):  #Compruebo que no excede la capacidad
          Xijh[PI[i]][j][h] = 1  #Asigno a este día y quirofano y actualizo fecha
          Yis[PI[i]][s] = 1
          Zsjh[s][j][h] = 1

          if(ORs_Surgeon_shj[s][h][j] == 0): #Opera en un nuevo quirófano // Si opera en un mismo quirófano, no hace falta
            ORs_Surgeon_shj[s][h][j] = 1   #Control de nº de quirófanos
            for k in range(ORs):
              Last_OR_shj[s][h][k] = 0   #Pone en 0 el último quirófano asignado
            Last_OR_shj[s][h][j]=1        #Pone en 1 el quirófano nuevo

          if(OpDays_s[s][h]==0): #Se asigna que ese día opera
            OpDays_s[s][h]=1

          asignacion_hecha = True
          if (ct_ORs[j][h] >= ct_surgeons[s][h]):
            ct_ORs[j][h] += t_surgeries[PI[i]]
            ct_surgeons[s][h] = ct_ORs[j][h]
            ct_surgeries[PI[i]] = ct_ORs[j][h]
            OR_time[h][j]=ct_ORs[j][h]

          else:
            ct_surgeons[s][h] += t_surgeries[PI[i]]
            ct_ORs[j][h] = ct_surgeons[s][h]
            ct_surgeries[PI[i]] = ct_surgeons[s][h]
            OR_time[h][j]=ct_surgeons[s][h]

          break

  if asignacion_hecha:
```

```
                break
            if asignacion_hecha:
                break


    return Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries


#
# FUNCION OBJETIVO
#

#Peso de las cirugías asignadas // l=1 política de eliminar cola , l=5 política de prioridad



def ORs_assigned(Surgeries, ORs, Horizon, Xijh):
    ORs_used=[[0 for h in range (Horizon)] for j in range (ORs)]
    for h in range (Horizon):
        for j in range (ORs):
            if(ORs_used[j][h]==0):      #Me ahorro tiempo computacional, no tengo que ver todas las cirugías
                for i in range (Surgeries):
                    if(Xijh[i][j][h]==1):
                        ORs_used[j][h]=1

    valor=0
    for j in range(ORs):
        valor+=sum(ORs_used[j])

    return valor


def FO(Surgeries, ORs, Horizon, Xijh, t_surgeries):
    valor=0
    for i in range(len(Xijh)):
```

```
        for j in range(ORs):
            for h in range (Horizon):
                if(Xijh[i][j][h]==1):
                    valor+=t_surgeries[i]
    return valor

#
=============================================================================
#    No me sirve de nada minimizar los ORs_used si no realizo cirugias
#
=============================================================================



def CuckooSearck(Surgeries):
    beta_CS=1.5
    S=[]
    for i in range (Surgeries):
        num= math.gamma(1+beta_CS)*math.sin(math.pi*beta_CS/2)
        denom= math.gamma((1+beta_CS)/2)*beta_CS*2**((beta_CS-1)/2)
        sigma_u= (num/denom)**(1/beta_CS)
        sigma_v= 1
        u= random.gauss(0, 1)*sigma_u
        v= random.gauss(0, sigma_v)
        S.append(u/(abs(v)**(1/beta_CS)))
    return S

def Intercambiar(seq,pos1,pos2):
    vecino=copy.deepcopy(seq)

    vecino[pos1]=seq[pos2]
    vecino[pos2]=seq[pos1]
    return vecino



#
=============================================================================
# ESCRITURA DE RESULTADOS POR PANTALLA
#
=============================================================================
```

```
#Escribir por pantalla los resultados (quirofano, paciente, dia, cirujano):
def timetable_OR(ORs, Horizon, Surgeries, Surgeons, Xijh, Yis):
    for j in range(ORs):
        print(f'\n En el quirófano {j} se realizan las siguientes cirugías:')
        for h in range(Horizon):
            for i in range(Surgeries):
                if (Xijh[i][j][h]==1):
                    for s in range (Surgeons):
                        if (Yis[i][s]==1):
                            print(f'   El paciente {i+1} se opera el día {h+1} por el cirujano {s+1}')
    return


#Imprime el cirujano 1 opera... El cirujano 2...
def timetable_Surgeons(ORs, Horizon, Surgeries, Surgeons, Xijh, Yis, ct_surgeries, t_surgeries):
    for s in range (Surgeons):
        print(f'\nEl cirujano {s+1} opera')
        for h in range (Horizon):
            for i in range(Surgeries):
                if(Yis[i][s]==1):
                    for j in range(ORs):
                        if (Xijh[i][j][h]==1):
                            print(f'En el quirófano {j} el día {h+1} a las {ct_surgeries[i]-t_surgeries[i]} -
{ct_surgeries[i]}')
    return


#Imprime para "En el quirofano 1: el dia 1... el dia 2..."
def timetable_OR_Horizon(ORs, Horizon, Surgeries, Xijh, ct_surgeries, t_surgeries):
    for j in range (ORs):
        print(f'\nEn el quirófano {j}')
        for h in range(Horizon):
            print(f'   En el día {h+1}')
            for i in range(Surgeries):
                if (Xijh[i][j][h]==1):
                    print(f'   El paciente {i+1} se opera de {ct_surgeries[i]-t_surgeries[i]}-{ct_surgeries[i]}')
            print('\n')
```

```
    return


#
# ============================================================================
# DIAGRAMA DE GANTT
#
# ============================================================================

def crear_gantt(H, dias, quirofanos, duracion_turno, ht):

    hbar = 40
    tticks = 40
    nor = len(quirofanos)

    fig, gantt = plt.subplots()

    diagrama = {
        "fig": fig,
        "ax": gantt,
        "hbar": hbar,
        "tticks": tticks,
        "maquinas": quirofanos,
        "ht": ht
    }

    # Etiquetas de los ejes:
    gantt.set_xlabel("Días")
    gantt.set_ylabel("Quirófanos")

    # Límites de los ejes:
    gantt.set_xlim(0, ht)
    gantt.set_ylim(0, nor*hbar)

    # Divisiones del eje de tiempo:
    gantt.set_xticks(range(duracion_turno, ht+1, duracion_turno),  minor=True)
    gantt.grid(True, axis='x', which='both')
```

```
    # Divisiones del eje de quirófanos:
    gantt.set_yticks(range(hbar, nor*hbar, hbar), minor=True)
    gantt.grid(True, axis='y', which='minor')



    # Etiquetas de dias:
    gantt.set_xticks(np.arange(duracion_turno, ht+1,duracion_turno))
    gantt.set_xticklabels(dias)

    # Etiquetas de quirófanos:
    gantt.set_yticks(np.arange(hbar/2, hbar*nor - hbar/2 + hbar,
                        hbar))
    gantt.set_yticklabels(quirofanos)



    return diagrama

# Función para armar tareas:
def agregar_tarea(diagrama, t0, d, maq, nombre, color=None):
    maquinas = diagrama["maquinas"]
    hbar = diagrama["hbar"]
    gantt = diagrama["ax"]
    ht = diagrama["ht"]

    # Chequeos:
    assert t0 + d <= ht, "La tarea debe ser menor al horizonte temporal."
    assert t0 >= 0, "El t0 no puede ser negativo."
    assert d > 0, "La duración d debe ser positiva."

    # Índice de la máquina:
    imaq = maquinas.index(maq)
    # Posición de la barra:
    gantt.broken_barh([(t0, d)], (hbar*imaq, hbar),
                      facecolors=(color))
    # Posición del texto:
    gantt.text(x=(t0 + d/2), y=(hbar*imaq + hbar/2),
               s=f"{nombre} ({d})", va='center', ha='center', color='black')
    return
```

```
def mostrar():
    plt.show()
    return

def colores (Surgeries, Surgeons, Yis, r_surgeon, g_surgeon, b_surgeon, r_color, g_color, b_color, rg):
    for s in range(Surgeons):
        r_surgeon[s]= rg.random()
        g_surgeon[s]= rg.random()
        b_surgeon[s]= rg.random()

    for i in range (Surgeries):
        for s in range(Surgeons):
            if(Yis[i][s]==1):
                r_color[i] = r_surgeon[s]
                g_color[i] = g_surgeon[s]
                b_color[i] = b_surgeon[s]
    return

#Dibuja el diagrama
def Draw_plan(Horizon, ORs, Surgeries, Surgeons, Cap_OR, PI, Xijh, Yis, ct_surgeries, t_surgeries):
    #Datos del problema que se está resolviendo
    H=Horizon #Número de días
    J=ORs #Númro de quirófanos
    duracion_turno= Cap_OR
    ht=duracion_turno*H

    #Objeto generador de los números aleatorios
    rg = np.random.default_rng(33)



    # Generación de una solución. Esto tiene que ser salida del decoding
    #Pacientes en cada quirófano y día. MATRIZ
    programacion_dj=np.full((H,J,10), fill_value=99)

    for d in range(H):
        for j in range(J):
            cont=0
```

```
        for i in range(Surgeries):
            if(Xijh[PI[i]][j][d]==1):
                programacion_dj[d,j, cont]=PI[i]
                cont+=1



CT_i=ct_surgeries
t_i=t_surgeries



#Nombre etiquetas quirófanos y días
nombre_quirofano="OR"
nombre_dia="D"
quirofanos=[]
dias_horizonte=[]

#Creamos las etiquetas de los días y los quirófanos
for h in range(H):
    dias_horizonte.append(nombre_dia+str(h+1))

for j in range(J):
    quirofanos.append(nombre_quirofano+str(j+1))

#Generación de colores por paciente
r_color=np.zeros(Surgeries)
b_color=np.zeros(Surgeries)
g_color=np.zeros(Surgeries)

r_surgeon=np.zeros(Surgeons)
b_surgeon=np.zeros(Surgeons)
g_surgeon=np.zeros(Surgeons)



colores(Surgeries, Surgeons,Yis, r_surgeon, g_surgeon, b_surgeon, r_color, g_color, b_color, rg)

#Creación del objeto diagrama
diagrama=crear_gantt(H, dias_horizonte, quirofanos, duracion_turno, ht)
```

```
    #Agregar intervenciones al diagrama
    for h in range(H):
        for j in range(J):
            for index_col in range(10):
                if (programacion_dj[h,j,index_col]!=99):
                    color=(r_color[programacion_dj[h,j,index_col]], b_color[programacion_dj[h,j,index_col]], g_color[programacion_dj[h,j,index_col]])
                    agregar_tarea(diagrama,CT_i[programacion_dj[h,j,index_col]]+duracion_turno*h-t_i[programacion_dj[h,j,index_col]],t_i[programacion_dj[h,j,index_col]],quirofanos[j],programacion_dj[h,j,index_col], color)
    mostrar()
    return
```

7.7 PYTHON. Elección de Decoding

```
# -*- coding: utf-8 -*-
"""
TFG - OPTIMIZACION QUIRURGICA MAXIMIZANDO LA UTILIDAD DE LOS RECURSOS
    MEDIANTE EL ALGORsITMO CUCKOO SEARCH

Created on Sun Feb 25 16:42:13 2024

@author: AlejandroPina
"""



# IMPORTACIÓN DE LIBRERIAS
from scheptk.util import read_tag, edit_tag, write_tag
from scheptk.util import sorted_index_asc, sorted_index_desc, sorted_value_asc, sorted_value_desc
import copy
from scheptk.util import find_index_min, find_index_max
from scheptk.scheptk import Model, Schedule, Task
import random
import math
from openpyxl import Workbook
```

```
import time
import sys
import Funciones
import numpy as np
import matplotlib.pyplot as plt



random.seed(49131791)

#
================================================================================
# 1. Generador de datos  (archivo aparte)
#
================================================================================

wb = Workbook()
#
================================================================================
# #2. Lectura de datos (Funcion aparte)
#
================================================================================
for y in range(16):
    ws = wb.create_sheet(title=f'Sheet_{y+1}')

    ws.cell(row=1, column=3).value = 'Underloaded'
    ws.cell(row=2, column=2).value = 'Surgeries_assigned'
    ws.cell(row=2, column=3).value = 'W_assigned'
    ws.cell(row=2, column=4).value = 'ORs_used'

    ws.cell(row=1, column=7).value = 'Overloaded'
    ws.cell(row=2, column=6).value = 'Surgeries_assigned'
    ws.cell(row=2, column=7).value = 'W_assigned'
    ws.cell(row=2, column=8).value = 'ORs_used'

    ws.cell(row=1, column=11).value = 'First_Available'
    ws.cell(row=2, column=10).value = 'Surgeries_assigned'
    ws.cell(row=2, column=11).value = 'W_assigned'
    ws.cell(row=2, column=12).value = 'ORs_used'
```

```
    ws.cell(row=2, column=16).value = 'MEJOR DECODING'

    for h in range(10):
        Datos = f'Datos{y+1}_{h+1}.txt'
        print(f'Leyendo {Datos}')
        ws.cell(row=h+3, column=1).value = Datos

        #EXCEL fila 3  =SI(Y(G3>C3;G3>K3); $G$1; SI(C3>K3;$C$1;$K$1)) y arrastro, o lo incluyo desde py
        ws.cell(row=h+3, column=16).value = f'=IF(AND(G{h+3}>C{h+3}, G{h+3}>K{h+3}), G1, IF(C{h+3}>K{h+3}, C1, K1))'

        Horizon, ORs, Surgeries, Surgeons, mds, Cap_S, Cap_OR, Lim_OR, t_surgeries, W, gamma, rd, dd = Funciones.Leer_datos(
            Datos)
        delta_ijh = [[[random.randint(0, 1) for h in range(Horizon)]
                       for j in range(ORs)] for i in range(Surgeries)]


        best_seq=Funciones.random_solution(Surgeries)
        for v in range(3):
            if(v==0):
                Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_underloaded(
                    Horizon, ORs, Surgeons, best_seq, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)

                W_assigned= Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries)
                assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh)
                ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh)


            elif(v==1):
                Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
                    Horizon, ORs, Surgeons, best_seq, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)

                W_assigned= Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries)
```

```
            assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh)
            ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh)




        else:
            Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_first(
                Horizon, ORs, Surgeons, best_seq, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR,
mds)

            W_assigned= Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries)
            assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh)
            ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh)




        if(v==0):
            ws.cell(row=h+3, column=2).value = f'{assigned} de {Surgeries}'
            ws.cell(row=h+3, column=3).value = f'{W_assigned}'
            ws.cell(row=h+3, column=4).value = f'{ORs_used}'

        elif(v==1):
            ws.cell(row=h+3, column=6).value = f'{assigned} de {Surgeries}'
            ws.cell(row=h+3, column=7).value = f'{W_assigned}'
            ws.cell(row=h+3, column=8).value = f'{ORs_used}'

        else:
            ws.cell(row=h+3, column=10).value = f'{assigned} de {Surgeries}'
            ws.cell(row=h+3, column=11).value = f'{W_assigned}'
            ws.cell(row=h+3, column=12).value = f'{ORs_used}'

#           Funciones.Draw_plan(Horizon, ORs, Surgeries, Surgeons, Cap_OR,best_seq, Xijh, Yis, ct_surgeries,
t_surgeries)



    ws.cell(row=14, column=1).value = 'Capacidad ORs'
    ws.cell(row=14, column=2).value = f'{Cap_OR*ORs*Horizon}'
```

```
#                print(f'Se han asignado {assigned} cirugías con peso de {W_assigned} puntos en un total de {ORs_used} quirófanos')



wb.save('TablaResultados_aleatorio.xlsx')
print('ASIGNACIÓN REALIZADA')
```

7.8 PYTHON. Codigo principal

7.8.1 Lectura

```
# -*- coding: utf-8 -*-
"""
TFG - OPTIMIZACION QUIRURGICA MAXIMIZANDO LA UTILIDAD DE LOS RECURSOS
    MEDIANTE EL ALGORS-ITMO CUCKOO SEARCH

Created on Sun Feb 25 16:42:13 2024

@author: AlejandroPina
"""



# IMPORTACIÓN DE LIBRERIAS
from scheptk.util import read_tag, edit_tag, write_tag
from scheptk.util import sorted_index_asc, sorted_index_desc, sorted_value_asc, sorted_value_desc
import copy
from scheptk.util import find_index_min, find_index_max
from scheptk.scheptk import Model, Schedule, Task
import random
import math
from openpyxl import Workbook
import time
import sys
import Funciones
```

```
import numpy as np
import matplotlib.pyplot as plt
import time

random.seed(49131791)

#
=============================================================================
# 1. Generador de datos  (archivo aparte)
#
=============================================================================

wb = Workbook()
#
=============================================================================
# #2. Lectura de datos (Funcion aparte)
#
=============================================================================
for y in range(16):
    ws = wb.create_sheet(title=f'Sheet_{y+1}')

    ws.cell(row=1, column=3).value = 'CB' #Cuckoo Basico
    ws.cell(row=2, column=2).value = 'Surgeries_assigned'
    ws.cell(row=2, column=3).value = 'W_assigned'
    ws.cell(row=2, column=4).value = 'ORs_used'

    ws.cell(row=1, column=7).value = 'C_GS_FI' #Cuckoo GeneralSwap_FirstImprovement
    ws.cell(row=2, column=6).value = 'Surgeries_assigned'
    ws.cell(row=2, column=7).value = 'W_assigned'
    ws.cell(row=2, column=8).value = 'ORs_used'

    ws.cell(row=1, column=11).value = 'C_RE' #Cuckoo Random_Extraction
    ws.cell(row=2, column=10).value = 'Surgeries_assigned'
    ws.cell(row=2, column=11).value = 'W_assigned'
    ws.cell(row=2, column=12).value = 'ORs_used'



    ws.cell(row=2, column=16).value = 'MEJOR DECODING'
```

```
for h in range(10):
    Datos = f'Datos{y+1}_{h+1}.txt'
    print(f'Leyendo {Datos}')
    ws.cell(row=h+3, column=1).value = Datos


    #EXCEL fila 3  =SI(Y(G3>C3;G3>K3); $G$1; SI(C3>K3;$C$1;$K$1)) y arrastro, o lo incluyo desde py
    ws.cell(row=h+3, column=16).value = f'=IF(AND(G{h+3}>C{h+3}, G{h+3}>K{h+3}), G1, IF(C{h+3}>K{h+3}, C1, K1))'


    Horizon, ORs, Surgeries, Surgeons, mds, Cap_S, Cap_OR, Lim_OR, t_surgeries, W, gamma, rd, dd = Funciones.Leer_datos(
        Datos)
    delta_ijh = [[[random.randint(0, 1) for h in range(Horizon)]
                   for j in range(ORs)] for i in range(Surgeries)]
```

7.8.2 Cuckoo Básico

```
    #DATOS CUCKOO
    eggs = 5  # numero soluciones
    xmax = 100
    xmin = 0
    iteracion=1
    #iteracion = con el tiempo   -> Nº de poblaciones que generamos
    p=0.1 #probabilidad de detección
    iteracion=1


    #
================================================================================
    # #Verificación que la secuencia hace lo que quiero
    # print(PI)
    # print(X_ni[0][0])
    #
    #  #Verificación inversa
    # aux=sorted_index_desc(X_ni[0][0])
    # seq=[0 for i in range (Surgeries)]
    # for i in range(Surgeries):
    #     seq[aux[i]]=i
    # print(PI)
```

```
    # print(seq)
    #
================================================================================


    #Generamos los datos iniciales




    X_ni = [[[0 for i in range(Surgeries)] for n in range(eggs)] for z in range(iteracion)]
    FOX_n = []
    FOX_n.append([])

    PI = sorted_index_desc(t_surgeries)
    for i in range(Surgeries):
       X_ni[0][0][i] = xmax-(PI[i])*(xmax-xmin)/Surgeries

    PI = sorted_index_asc(t_surgeries)
    for i in range(Surgeries):
       X_ni[0][1][i] = xmax-(PI[i])*(xmax-xmin)/Surgeries

    for n in range(2, eggs):
       for i in range(Surgeries):
          r = random.uniform(0, 1)
          X_ni[0][n][i] = (xmax-xmin)*r+xmin




    #0-CB  1-GS_FI  2-RandomExtraction
    # Conversión de continuo a Secuencia de todos los nidos
    seqs = []
    seq=[0 for i in range (Surgeries)]
    seqs.append([])
    for n in range(eggs):
       aux=sorted_index_desc(X_ni[0][n])
       for i in range(Surgeries):
          seq[aux[i]]=i
       seqs[0].append(seq)
```

```
        Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
            Horizon, ORs, Surgeons, seqs[0][n], delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)
        FOX_n[0].append(Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries))



    X_best=[]
    FO_best=[]
    orddesc_by_FO=sorted_index_desc(FOX_n[0])
    X_best.append(X_ni[0][orddesc_by_FO[0]])
    FO_best.append(max(FOX_n[0]))



    t_ini= time.time()
    factor_tiempo= 0.0125 #Se proponen 2 factores concluyendo en el artículo Planificación que este factor es efectivo
    tiempo_limite= Surgeries*Horizon*ORs*factor_tiempo


    while(time.time()-t_ini < tiempo_limite):
        iteracion+=1
        X_ni.append([[0 for i in range(Surgeries)] for n in range (eggs)])
        z=iteracion-1 #Indice actual
        seqs.append([])
        FOX_n.append([])
        for n in range(eggs):  # Genero las nuevas soluciones
            S= Funciones.CuckooSearck(Surgeries)
            for i in range(Surgeries):  # Genero la secuencia en continuo
                X_ni[z][n][i] = X_ni[z-1][n][i] + random.gauss(0, 1)*0.01*S[i]*(X_ni[z-1][n][i]-X_best[z-1][i])
                #X_ni[z+1][n][i] = random.uniform(0,100)  #X_ni[z][n][i] +0.01*(X_ni[z][n][i]-X_best[z+1][i])



#PUEDO EVALUAR EN LOS 3 DECODING Y QUE ME ELIJA LA MEJOR DE LOS 3 DECODING


#FALTA IMPLEMENTAR LA PROBABILIDAD DE QUE ABANDONE O NO ESE NIDO Y ELEGIR OTRO


            #Detección del Cuckoo
            r=random.uniform(0, 1)
```

```
            if(r<p): #Es detectada
                ran= random.uniform(0, 1)
                d1= random.randint(0, eggs-1)
                d2= random.randint(0, eggs-1)
                while(d1==d2):
                    d2= random.randint(0, eggs-1)

                Xd1= copy.deepcopy(X_ni[z-1][d1])
                Xd2= copy.deepcopy(X_ni[z-1][d2])
                for i in range(len(Xd1)):
                    #Xnew=X+rand(Xd1-Xd2)
                    X_ni[z][n][i]= X_ni[z][n][i]+ran*(Xd1[i]-Xd2[i])




            #Evaluación del nuevo huevo
            aux=sorted_index_desc(X_ni[z][n])
            for l in range (Surgeries):
                seq[aux[l]]=l
            seqs[z].append(seq)



            Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
                Horizon, ORs, Surgeons, seqs[z][n], delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR,
mds)
            FOX_n[z].append(Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries))

            #Mejoramos la solución de partida con  GS_FI
            if(r<p):
                secuencia=copy.deepcopy(seqs[z][n])

                improvement= False
                for i in range(len(secuencia)-1):
                    for j in range(i+1,len(secuencia)):
                        vecino=Funciones.Intercambiar(secuencia,i ,j)
```

```
                Xijh_vecino, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
                    Horizon, ORs, Surgeons, vecino, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd,
Lim_OR, mds)

                obj_vecino=Funciones.FO(Surgeries, ORs, Horizon, Xijh_vecino, t_surgeries)
                if (FOX_n[z][n]<obj_vecino):
                    improvement=True
                    seqs[z][n]=copy.deepcopy(vecino)
                    FOX_n[z][n]=obj_vecino
                    break
            if(improvement==True):
                break


        #Me quedo el anterior si no mejora
        if(FOX_n[z-1][n] >= FOX_n[z][n]):
            seqs[z][n]=copy.deepcopy(seqs[z-1][n])
            FOX_n[z][n]=FOX_n[z-1][n]


    orddesc_by_FO=sorted_index_desc(FOX_n[z])
    X_best.append(X_ni[z][orddesc_by_FO[0]])
    FO_best.append(max(FOX_n[z]))


#print(seqs[iteracion-1])
#print(FOX_n[iteracion-1])

# La mejor solución una vez acabada las iteraciones
orddesc_by_FO= sorted_index_desc(FO_best)
best_solution= X_best[orddesc_by_FO[0]]
aux=sorted_index_desc(best_solution)
for i in range (Surgeries):
    seq[aux[i]]=i
best_seq= copy.deepcopy(seq)
FO_bs= max(FO_best)
#print(FO_bs)
```

7.8.3 Evaluación de las variantes

```
for v in range(3):  #Evalúo las 3 variantes
    #factor_tiempo= 0.0125 #Se proponen 2 factores concluyendo en el artículo Planificación que este factor es efectivo
    tiempo_limite= Surgeries*Horizon*ORs*factor_tiempo
    best_seq_partida= copy.deepcopy(best_seq)
    FO_bs_partida= FO_bs

    if(v==0): #CB
      Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
        Horizon, ORs, Surgeons, best_seq, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)
      W_assigned= Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries)
      assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh)
      ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh)

    elif(v==1): #C_GS_FI
      Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
        Horizon, ORs, Surgeons, best_seq_partida, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)
      W_assigned= Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries)
      assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh)
      ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh)

      tiempo_inicio= time.time()
      tiempo_act=time.time()
      buscar_nuevo=True #Me marca si hay vecinos que puedan mejorar
      best_seq_GS=copy.deepcopy(best_seq_partida)

      while(buscar_nuevo==True and tiempo_act-tiempo_inicio < tiempo_limite):
        improvement= False
        secuencia=copy.deepcopy(best_seq_GS)
        for i in range(len(secuencia)-1):
```

```
            for j in range(i+1,len(secuencia)):
                vecino=Funciones.Intercambiar(secuencia,i ,j)

                Xijh_vecino, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(Horizon, ORs, Surgeons, vecino, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)
                obj_vecino=Funciones.FO(Surgeries, ORs, Horizon, Xijh_vecino, t_surgeries)
                if (W_assigned<obj_vecino):
                    improvement=True
                    best_seq_GS=copy.deepcopy(vecino)
                    W_assigned=obj_vecino
                    assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh_vecino)
                    ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh_vecino)

                    buscar_nuevo=True #Sirve para volver a aplicar LS a la nueva solución.

                    break
                if (improvement==True):
                    break
            if (improvement== False):
                buscar_nuevo= False
            tiempo_act= time.time()

        else:  #C_RE
            Xijh, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(
                Horizon, ORs, Surgeons, best_seq_partida, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)

            W_assigned= Funciones.FO(Surgeries, ORs, Horizon, Xijh, t_surgeries)
            assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh)
            ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh)

            #Realizo intercambio por pareja  veces, cada 20 cirugías, 1 cambio
            N_intercambios=1 #math.ceil(Surgeries/20)
```

```
tiempo_inicio= time.time()
tiempo_act= time.time()
best_seq_RE=copy.deepcopy(best_seq_partida)

while(tiempo_act-tiempo_inicio < tiempo_limite):
   secuencia=copy.deepcopy(best_seq_RE)
   for _ in range (N_intercambios):
      a=random.randint(0, Surgeries-1)
      b=random.randint(0, Surgeries-1)
      while(a==b):
         b=random.randint(0, Surgeries-1)
      vecino=Funciones.Intercambiar(secuencia, a, b)



      Xijh_vecino, Yis, Zsjh, ct_ORs, ct_surgeons, ct_surgeries=Funciones.assign_overloaded(Horizon,
ORs, Surgeons, vecino, delta_ijh, gamma, t_surgeries, Cap_OR, Cap_S, rd, Lim_OR, mds)
      obj_vecino=Funciones.FO(Surgeries, ORs, Horizon, Xijh_vecino, t_surgeries)
      if (W_assigned<obj_vecino):
         improvement=True
         best_seq_RE=copy.deepcopy(vecino)
         W_assigned=obj_vecino
         assigned= Funciones.Surgeries_assigned(Surgeries, ORs, Horizon, Xijh_vecino)
         ORs_used= Funciones.ORs_assigned(Surgeries, ORs, Horizon, Xijh_vecino)

      tiempo_act= time.time()
```

7.8.4 Escritura de resultados

```
if(v==0):
   ws.cell(row=h+3, column=2).value = f'{assigned} de {Surgeries}'
   ws.cell(row=h+3, column=3).value = f'{W_assigned}'
   ws.cell(row=h+3, column=4).value = f'{ORs_used}'
```

```
        elif(v==1):
            ws.cell(row=h+3, column=6).value = f'{assigned} de {Surgeries}'
            ws.cell(row=h+3, column=7).value = f'{W_assigned}'
            ws.cell(row=h+3, column=8).value = f'{ORs_used}'


        else:
            ws.cell(row=h+3, column=10).value = f'{assigned} de {Surgeries}'
            ws.cell(row=h+3, column=11).value = f'{W_assigned}'
            ws.cell(row=h+3, column=12).value = f'{ORs_used}'

#           Funciones.Draw_plan(Horizon, ORs, Surgeries, Surgeons, Cap_OR,best_seq, Xijh, Yis, ct_surgeries, t_surgeries)



    ws.cell(row=14, column=1).value = 'Capacidad ORs'
    ws.cell(row=14, column=2).value = f'{Cap_OR*ORs*Horizon}'



#               print(f'Se han asignado {assigned} cirugías con peso de {W_assigned} puntos en un total de {ORs_used} quirófanos')




wb.save('EJEMPLO.xlsx')
print('ASIGNACIÓN REALIZADA')
```

Referencias

[1] B. d. E. LAB-ES, «Niveles de decisiones,» [En línea]. Available: https://labes-unizar.es/tipos-de-objetivos-estrategicos-tacticos-y-operativos/?expand_article=1.

[2] UNED, «Optimización heurística y aplicaciones,» [En línea]. Available: https://portal.uned.es/EadmonGuiasWeb/htdocs/abrir_fichero/abrir_fichero.jsp?idGuia=43158.

[3] Gamco.es, «Metaheurística concepto y definición.,» [En línea]. Available: https://gamco.es/glosario/metaheuristicas/.

[4] U. A. d. Barcelona, «UAB Divulga,» [En línea]. Available: https://www.uab.cat/web?cid=1096481466568&pagename=UABDivulga%2FPage%2FTemplatePageDetallArticleInvestigar&param1=1258357797938#:~:text=El%20concepto%20de%20%C3%B3ptimo%20local,el%20concepto%20de%20%C3%B3ptimo%20global..

[5] J. M. Molina Pariente, E. W. Hans, J. M. Framiñán Torres y T. Gomez Cia, «New heuristics for planning operating rooms.,» Sevilla, 2015.

[6] X.-S. Y. y. S. Deb, «Cuckoo Search,» 2009.

[7] X. L. &. M. Yin, «A hybrid cuckoo search via Lévy flights for the,» 2013.

[8] D. H. Garg, *https://www.youtube.com/watch?v=8sHkQ8kGEr8g,* 2020.

[9] FasterCapital, «Fundamentos teóricos de las distribuciones de cola pesada,» [En línea]. Available: https://fastercapital.com/es/tema/fundamentos-te%C3%B3ricos-de-las-distribuciones-de-cola-pesada.html#:~:text=Las%20distribuciones%20de%20cola%20pesada%20tienen%20una%20curtosis%20m%C3%A1s%20alta,agudo%20y%20colas%20m%C3%A1s%20gorras..

[10] P. Dasgupta y S. Das, «A Discrete Inter-Species Cuckoo Search for flowshop scheduling problem,» 2015.

[11] J. M. Molina Pariente, V. Fernández Viagas y J. M. Framiñán Torres, «Integrated operating room planning and scheduling problem with assistant surgeon dependent surgery duration,» 2015.

[1] P. Aguilera Bonet y J. Payán Somet, «Formato de Publicación de la Escuela Técnica Superior de Ingeniería de Sevilla».

2
]