Trabajo Fin de Grado
Ingeniería de Organización Industrial

Optimización sanitaria para equilibrado de cirujanos en una unidad quirúrgica mediante algoritmos evolutivos

Autor: Alejandro García Izquierdo
Tutor: Víctor Fernández-Viagas Escudero

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

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería de Organización Industrial

Optimización sanitaria para equilibrado de cirujanos en una unidad quirúrgica mediante algoritmos evolutivos

Autor:
Alejandro García Izquierdo

Tutor:
Víctor Fernández-Viagas Escudero

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

Proyecto Fin de Carrera: Optimización sanitaria para equilibrado de cirujanos en una unidad quirúrgica mediante algoritmos evolutivos

Autor: Alejandro García Izquierdo

Tutor: Víctor Fernández-Viagas Escudero

El tribunal nombrado para juzgar el Proyecto 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 maestros

Agradecimientos

Quisiera expresar mis agradecimientos a aquellas personas que me han acompañado durante la realización de este trabajo, así como el periodo de realización del grado y esta gratificante etapa universitaria. En primer lugar, agradecer a mi tutor Víctor Fernández-Viagas Escudero, por toda la información aportada, así como sus buenos consejos, durante la realización del trabajo, fruto de su experiencia en el ámbito de estudio abordado. También agradecer al resto de profesores, que gracias a su profesionalidad han permitido enriquecer mis conocimientos en ingeniería, así como otras áreas relacionadas con la misma, de igual forma, a mis compañeros, con quienes he compartido esta etapa académica y que han contribuido a hacer de ella una experiencia significativa en mi vida, y en definitiva, a la Escuela Técnica Superior de Ingeniería y a la Universidad de Sevilla, así como el resto de organizaciones relacionadas, las cuales han contribuido a mi formación, no solo a nivel académico, sino también a nivel personal.

De igual forma agradecer a mi familia y gente cercana, por el apoyo incondicional y el ánimo transmitido en todo momento, factores que me han aportado una mayor motivación y me han permitido sobrepasar momentos que han requerido de un esfuerzo mayor.

A todos ellos, mi más sincera gratitud.

Resumen

En el siguiente documento será abordado un problema de optimización quirúrgica que consiste en la asignación de pacientes de una lista de espera, a las salas de operaciones de un área hospitalaria y a los días de un horizonte de planificación en el que ser realizadas. Se trata de un problema de optimización matemática, formado por una función objetivo y una serie de restricciones y variables.

Dado que ha sido abordado un problema complejo, es decir, con un alto número de soluciones posibles para alguna de las instancias y el tiempo de computación para los métodos de resolución exacta sigue una exponencial en función de dicho número, se ha usado para su resolución una serie de métodos aproximados.

En primer lugar, se ha abordado el problema en cuestión, analizando el modelo ILP en el que se basa, así como el entorno al que está enfocado, para posteriormente realizar una codificación del mismo, trabajando a su vez con modificaciones e implementaciones sobre el modelo anterior relativas a la resolución aproximada.

Posteriormente, e impulsado a su vez por este tipo de resolución, se ha abordado la misma mediante la creación de una serie de modelos basados en los algoritmos memético, los cuales han incorporado operadores, tanto propios, ideadas siguiendo un determinado razonamiento, como otros populares de la literatura y adaptados al problema abordado.

Una vez implementados, se ha desarrollado un banco de pruebas amplio, con múltiples instancias y datos en las mismas generados de forma inteligente, para realizar un sistemático proceso de pruebas computacionales con el fin de comparar los modelos originales, y obtener una base para estudiar y seleccionar, aquel que mejores soluciones genera. Posteriormente ha sido calibrado mediante razonadas modificaciones en el mismo, relativas tanto a su marco general, operaciones concretas, y parámetros de entrada que recibe. Todo ello de una manera equitativa, bajo una condición de parada basada en el tiempo de ejecución y con el fin de la determinación un algoritmo eficaz para el problema tratado, respaldado por datos numéricos.

Finalmente, se presentan una serie de conclusiones del estudio realizado, así como propuestas e implementaciones de escalabilidad del mismo, para líneas futuras.

Relativo al software empleado para la realización del estudio, se ha destinado la programación del mismo a lenguaje Python, con uso limitado de librerías, para mejorar la eficiencia computacional y realizar los distintos algoritmos y métodos desde un inicio, con el fin de realizar la mejor adaptación posible al problema a tratar, así como a los distintos objetivos propuestos.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Actualmente existe un aumento en la presión sobre las organizaciones sanitarias, relativa a la posibilidad de prestar cirugías de calidad al menor coste posible, debido al efecto combinado de restricciones presupuestarias más estrictas, aparición de nuevas enfermedades e incremento de la población en edad avanzada (Fei, Meskens, & Chu, 2010). Por otro lado, los hospitales europeos han sufrido en los últimos años reformas exigidas por las autoridades para mejorar la calidad de vida de los ciudadanos, el cual, sumado a ese aumento de la complejidad del problema, provoca una limitación en las organizaciones sanitarias para la gestión de sus recursos humanos y materiales (Roland, Di Martinelly, Riane, & Pochet, 2010)

Atendiendo el objetivo conjunto de la reducción de costes y la maximización del nivel de satisfacción del cliente. Una unidad de gran interés para la toma de decisiones, con un gran impacto en el rendimiento del hospital en su conjunto es el quirófano, al ser el centro de costes e ingresos más grande del mismo. (Cardoen, Demeulemeester, & Beliën, 2010)

Por ello, son cada vez más frecuentes el uso de procedimientos de planificación y programación propios de métodos de optimización sobre investigación operativa, en centros sanitarios con grandes volúmenes de pacientes, similares a los usados en el sector industrial para la programación y secuenciación de trabajos a realizar en las distintas máquinas y líneas de producción; propios de un nivel de decisión operativa y conocidos como *Scheduling Problems*, permitiendo mejorar la eficiencia de la misma en base a distintos factores.

Este tipo de problemas llevan estudios previos a su implementación, con el fin de adaptar el problema a modelos matemáticos y decodificaciones computacionales, que permitan su posterior implementación mediante algún tipo de Software, así como su análisis durante el desarrollo de los mismos, para mejorar su eficiencia en el entorno considerado. Una vez implementados derivan en mejoras en la planificación y el control de los distintos procesos desarrollados, y en definitiva del impacto de las estrategias de gestión realizadas por el tomador de decisiones, ofreciendo mejoras significativas en términos de eficiencia, reducción de costes, así como la posibilidad de gestionar situaciones complejas, que si de una toma de decisiones, sin herramientas de apoyo, se tratase. Por tanto, el riesgo de una mala asignación de los recursos, así como retrasos y aumento de costes asociados a los mismos, con la implementación de dichas herramientas, sumado a su análisis detallado y sistemático de variables y restricciones, se ve reducido.

En resumen, sin una herramienta de optimización, la toma de decisiones se realiza de una manera más rápida e intuitiva, basándose en el juicio inmediato, y que en gran medida requiere un alto grado de experiencia, por lo que en muchos casos puede conducir a errores, derivando en la necesidad de corrección de estas decisiones apresuradas y en definitiva en problemas a largo a plazo, así como la insatisfacción de la demanda requerida.

1.1 Bases del estudio

Para una mejor comprensión de lo recién descrito, así como las bases del estudio realizado se presenta a continuación una serie de cuestiones relativas a su ámbito de procedencia y la resolución del mismo.

1.1.1 Programación de la producción

En primer lugar, se presenta el término programación de la producción, el cual es el área generalizada del problema tratado, así como de los problemas de optimización sobre secuenciación y planificación.

La programación de la producción es una tarea frecuente en la gestión de la producción, propia de un nivel de decisión operativo, la cual es definida como "la asignación de recursos para la fabricación de un conjunto de productos" (Framinan, Leisten, & Ruiz, 2013). Donde los recursos se refieren a personas, máquinas, instalaciones, etc., que realizan las operaciones de procesamiento de los productos, y los productos, son la materia objeto de procesamiento en los recursos.

Como resultado de dicha programación se obtiene un programa de producción (*Production Schedule*), el cual contiene la información sobre los instantes de comienzo y finalización de cada uno de los productos a

procesar, así como en que recursos será realizado dicho procesamiento. “La obtención de cronogramas económicos y confiables constituye el núcleo de la excelencia en el servicio al cliente y la eficiencia en las operaciones de manufactura. Por tanto, la programación constituye un área de vital importancia para la competencia en las empresas manufactureras.” (Framinan, Leisten, & Ruiz, 2013).

Por ello, es frecuente para dicha resolución el uso de algoritmos y modelos matemáticos, los cuales permitan resolver el problema definido en base al contexto de su entorno y atendiendo a los objetivos definidos para el mismo, aportando una resolución eficiente y la generación de una solución admisible dentro del conjunto finito de las mismas.

1.1.2 Resolución exacta y resolución aproximada

La resolución de este tipo de problemas y en general de los problemas de optimización puede ser clasificada en dos grandes ramas en base a la metodología empleada para la misma. Los cuales se tratan de la resolución exacta y la resolución aproximada, donde la principal diferencia entre ambas es la garantía de la optimalidad y la elección de uso en base al tipo de problema tratado.

Por un lado, la resolución exacta garantiza el óptimo del problema, es decir, la mejor solución de entre todas las posibles al problema tratado, mientras que los algoritmos aproximados, no tienen la garantía de optimalidad, pero si generan una que se acerque al mismo, más en concreto, la mejor que hayan encontrado hasta el momento de su finalización.

Por otro lado, el motivo de su uso se debe al tipo de problema abordado, siendo clasificados en polinomiales (P) y no polinomiales (NP), según su tipo de modelado (Pinedo, 2005).

Para el caso de los polinomiales, pueden ser resueltos de forma exacta y en un tiempo razonable independientemente de su tamaño. Para su resolución destacan los algoritmos constructivos exactos como el Algoritmo Johnson, Algoritmo de Lawler o Algoritmo de Moore. Para el caso de los problemas no polinomiales, dicha resolución dependerá del tamaño del mismo, mientras un problema no polinomial con una instancia pequeña puede ser resuelto de forma exacta, mediante algún algoritmo enumerativo (Branch and Bound, Programación Matemática o Programación Dinámica), para el caso de instancias grandes (NP Hard), el tiempo de cómputo empleado mediante resolución exacta sería físicamente inaceptable, derivando en un problema complejo y motivo de implementación de algoritmos aproximado, como son las heurísticas constructivas y de mejora, o las metaheurísticas.

Dado que el problema tratado es complejo, causa de uso de instancias de mayor tamaño, han sido empleadas para su resolución heurísticas y metaheurísticas, las cuales serán introducidas y desarrolladas, más en detalle, en sus capítulos específicos.

1.2 Ámbito del estudio

Como ya ha siso introducido, el problema de optimización tratado, se encuentra en un entorno sanitario y se enfoca en la optimización quirúrgica. Dicho problema puede ser considerado como una aplicación concreta de los problemas de programación, donde los recursos son las salas de operaciones, y los trabajos, las distintas cirugías a realizar sobre los pacientes en la lista de espera. De igual forma el resultado que se pretende obtener es el Programa Quirúrgico, con las mismas características que el Programa de Producción, antes definido.

Debido a su creciente desarrollo, por los motivos ya destacados, sobre los presupuestos restrictivos, y la consecuente implementación de técnicas propias de investigación operativa para la toma de decisiones, así como los avances tecnológicos de los últimos años. Esta rama de la programación se encuentra muy generalizada y desarrollada, en la actualidad, contando con un gran número de aportaciones en la literatura.

De este modo, las decisiones relacionadas con la gestión de quirófanos se suelen descomponer en tres niveles jerárquicos (Cardoen, Demeulemeester, & Beliën, 2010), estratégico, táctico y operativo, con un enfoque similar al de un entorno productivo.

El nivel estratégico aborda el problema de Planificación de Combinación de Casos, mediante el uso de modelos que establezcan una asignación de recursos para un horizonte de planificación, y la evaluación de una serie de variables, con el objetivo de establecer el equilibrio de un determinado factor. Así, en (Ivo & Vissers,

2002) se evalúa la carga de afluencia de pacientes en base a dos variables de decisión referidas al número y tipo de las cirugías para un número de recursos requeridos, estableciendo así un perfil de ingreso o admisión para la especialidad.

El nivel táctico se produce una vez tomadas las decisiones del nivel estratégico, donde los recursos del quirófano se asignan en un horizonte de planificación de varias semanas, estableciendo el Programa Quirúrgico Maestro. Así, en (Blake & Donald, 2002), proponen un modelo de programación entera para la asignación del tiempo de quirófano en cinco divisiones de un hospital, objeto de su implementación; mediante el cual realizan la generación de un plan maestro quirúrgico equitativo, de forma rápida, optimizando la utilización de los recursos operatorios y logrando ahorros en costes administrativos.

Por último, fechas de inicio y finalización más exactas, las salas de operaciones asignadas y el tiempo de cada cirugía programada se determinan en el nivel operativo. Más en concreto, es resuelto en dos pasos (Magerlein & Martin, 1978), los cuales son denominados de la siguiente manera, propuesta en (Cardoen, Demeulemeester, & Beliën, 2010).

1º Programación anticipada, donde a cada cirugía de la lista de espera se le asigna un quirófano y un día para su intervención, realizando la programación antes de la fecha quirúrgica.

2º Programación de asignación, donde para cada día del horizonte de planificación, se obtiene una secuencia de las cirugías de cada quirófano, realizando la programación durante el día de la cirugía para aquellos pacientes disponibles.

Por otro lado, el nivel de decisión operativa, se divide en dos subniveles, uno fuera de línea, el cual es resuelto mediante los dos pasos anteriores, y otro en línea que consiste en emplear mecanismos de control para monitorear el proceso y reaccionar ante eventos imprevistos (Hans, van Houdenhoven, & Hulshof, 2011).

Dicho trabajo, se ha centrado, en el nivel operativo de la gestión de quirófanos fuera de línea, más en concreto, en el paso de programación anticipada, muy acorde a los objetivos planteados.

De esta manera, se observa como la gestión de decisiones en un entorno quirúrgico, es muy similar a un entorno productivo, donde análogo a los ejemplos mostrados, se realiza una estimación de la demanda desde un punto de vista estratégico, se elabora un Plan Maestro de Producción, donde se asigna recursos y tiempos, desde un punto de vista táctico y de manera más aproximada, y se realiza una programación, desde un punto de vista operativo, estableciendo fechas y tiempo para los trabajos, de manera más exacta.

1.3 Objeto del estudio

De este modo el objeto del estudio y como ya ha sido presentado con anterioridad se basa en la optimización en un ámbito sanitario, relativo a la planificación de cirugías de una lista de espera en una serie de quirófanos para un horizonte de planificación de días, con un enfoque operativo.

Más en detalle se ha centrado en la maximización del número de cirugías programadas y la minimización de los movimientos realizados por los cirujanos, como representación a la afirmación sobre prioridades en conflicto y preferencias de las partes interesadas, las cuales dificultan la gestión operativa en el hospital (Cardoen, Demeulemeester, & Beliën, 2010).

Para su resolución, al tratarse de un problema de optimización combinatoria; y, por ende, complejo, se ha empleado metaheurísticas centradas en los Algoritmos Meméticos, y objeto de mejora mediante un análisis exhaustivo de sus resultados obtenidos, con el fin de desarrollar modelos eficaces para la resolución del problema planteado.

2 DESCRIPCIÓN DEL PROBLEMA

Para la buena comprensión del trabajo en su totalidad, así como una eficiente investigación del mismo durante su desarrollo, se necesita principalmente un entendimiento más exhaustivo del problema bajo consideración. Por lo tanto, en este capítulo, se procederá a realizar una descripción más detallada del problema, atendiendo a las distintas características que lo forman, relativas a su entorno productivo, así como el modelo matemático generado a partir del mismo y formado por los distintos elementos, variables, restricciones y funciones objetivo. Todo ello según lo propuesto en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015)

2.1 Entorno productivo

En primer lugar, se explica el entorno productivo del problema, donde se nos presenta un área especializada de un hospital, la cual, dispone de un número determinado de quirófanos y cirujanos para realizar una serie de cirugías de una lista de espera, las cuáles a su vez deben ser programadas dentro de un horizonte de planificación.

Dichas cirugías tienen asignado un único cirujano responsable de su realización y se asignan a un único conjunto día – quirófano. Tanto el día – quirófano, como el cirujano sólo pueden realizar una cirugía durante el tiempo que dure la misma, siendo la relación de ambos con la cirugía de uno a uno.

De igual forma el entorno del problema tratado no considera de forma directa, etapas previas de preparación de los trabajos, como son las pruebas médicas; o posteriores, por ejemplo, los descansos postoperatorios, sino que los tiempos de estas son incluidos en las duraciones totales de la cirugía a programar, establecidos en el diseño del banco de pruebas.

Por último, comentar que los elementos del problema presentan datos distintos entre sí, como los relativos, en el caso de los trabajos, a tiempos de proceso, fechas de llegada y vencimiento, etc. O en el caso de los recursos sanitarios, relativos a capacidades, límite de trabajos a procesar en un día, etc. Lo mismo sucede con distintas especificaciones del problema como la relación uno a uno antes comentada. Afectando todo ello al modelo del problema, el cual será explicado en detalle, a continuación.

2.2 Elementos del problema a optimizar

Adicional a la resolución aproximada, se ha planteado inicialmente el problema con un enfoque de resolución exacta, estableciendo el modelo ILP asociado y formado por los distintos datos, elementos, variables, restricciones y función objetivo que presenta. Dicho modelo se elabora únicamente para una mejor comprensión del problema, sin llegar por tanto a ser usado en su resolución, pues para ello, se ha centrado el trabajo en el uso de métodos aproximados.

Relativo a los parámetros principales del problema, se presentan los siguientes conjuntos, los cuáles hacen referencia a los 4 elementos del problema, cirugías o pacientes de la lista de espera, quirófanos y cirujanos de la especialidad, y días del horizonte de planificación; denotados por I, J, S y H, respectivamente.

Por otro lado, los elementos del problema, presentan atributos tanto propios como compartidos, los cuales se presentan a continuación, junto a una explicación del motivo de su implementación. Adicionalmente en la Tabla 2–1. Resumen de datos y variables para el modelo ILP se muestra una lista con los datos y variables principales del problema, utilizadas en el modelo ILP y en los posteriores algoritmos aproximados, así como su notación usada en todo el trabajo.

Tabla 2–1. Resumen de datos y variables para el modelo ILP

Conjuntos, datos y variables utilizadas en modelo de decisión ILP	
Índices y conjuntos	
$h \in H$	Conjunto de días dentro del horizonte de planificación (índice y tamaño)
$i \in I$	Conjunto de pacientes (cirugías) en la lista de espera (índice y tamaño)
$j \in J$	Conjunto de salas de operaciones del área (índice y tamaño)
$s \in S$	Conjunto de cirujanos del área (índice y tamaño)
Parámetros	
r_{jh}	Capacidad regular de la sala de operaciones j en el día h (min.)
a_{sh}	Capacidad regular del cirujano s en el día h (min.)
u_s	Número entero no negativo de salas de operaciones en las que el cirujano s puede realizar cirugías en un mismo día
γ_i	Cirujano a cargo de la cirugía i
rd_i	Fecha de inicio para realizar la cirugía i
d_i	Fecha de vencimiento (límite) para realizar la cirugía i
δ_{ijh}	Parámetro binario que da como resultado 1 si la cirugía del paciente i se puede realizar en la sala de operaciones j en el día h; 0 en caso contrario
t_i	Tiempo esperado de la cirugía i (min.)
w_i	Peso clínico de la cirugía i
Variables	
X_{ijh}	1 si el paciente i es asignado a la sala de operaciones j en el día h; 0 en caso contrario
Z_{sjh}	1 si el cirujano s realiza cirugía en la sala de operaciones j en el día h; 0 en caso contrario

En el caso de las cirugías, éstas presentan una serie de atributos propios, relativos a las fechas de programación de las mismas, así como su duración y el cirujano responsable de llevarla a cabo.

La fecha más temprana en la que se puede realizar la cirugía, se debe imponer debido a los propios procedimientos llevados a cabo en un hospital, tratándose éstos de una serie de pruebas médicas previas a la realización de las mismas, por ende, cada cirugía i presenta un Release Date, el cual será denotado por rd_i.

De forma análoga, presentan una fecha más tardía, debido al tiempo máximo antes de su tratamiento, el cuál es establecido por el tipo de urgencia del paciente, definidos por los Servicios Nacionales de Salud en base a una serie de criterios clínicos y sociales, por ende, cada cirugía i presenta un Due Date, el cual será denotado por d_i.

Por otro lado, cada una de las cirugías tendrá establecido una duración, t_i, en minutos y un cirujano responsable de su realización, γ_i, cuyo valor será una de los valores del conjunto de cirujanos, es decir la etiqueta del mismo.

Para cada quirófano en cada día se nos presenta una capacidad relativa al tiempo que este recurso se encuentran disponible, el cual se ha establecido en 8 horas, es decir 480 minutos, para todos ellos, y es denotado por r_{jh}.

De igual forma, sucede con los cirujanos, lo cuales cada día tienen un tiempo disponible para la realización de cirugías. Debido a que este recurso es humano y éstos pueden presentar descansos o jornadas distintas establecidas, su valor puede ir de 0 a 480 minutos, el cuál será denotado por a_{sh}.

Relativo a la capacidad de los recursos, mencionar también que el resto de recursos perioperatorios humanos e instrumentales, así como las instalaciones de recuperación están disponibles cuando sea necesario y no representan cuellos de botella para el problema estudiado.

Para cada cirujano, se define también un número máximo de quirófanos donde podría ser asignado en un mismo día, denotado por u_s, y cuyo objetivo es el de reducir su tiempo de inactividad y la superposición de

cirugías consecutivas por parte del mismo.

Por último, se presenta un atributo compartido entre cirugías, quirófanos y días, el cual representa la compatibilidad de los mismos, estableciendo si es posible la realización de la cirugía i en el quirófano j y el día h. Por ende, es de cuantificación binaria, tomando valor 1 si es posible y 0 en caso contrario, y denotado por δ_{ijh}.

A modo de resumen de lo recién explicado, en la Tabla 2–2, se presenta los elementos y atributos del problema en consideración, atendiendo a las características de los mismos.

Tabla 2–2. Elementos del problema de asignación de cirugías a día-quirófano

ELEM	CJTO	NC	DATOS				
			Nombre	Parámetro	TV	P	Valor
Cirugía	$i = 1 \dots I$	I_u	Fecha más temprana	rd_i	E	P	S6
			Fecha más tardía	d_i	E	P	S6
			Duración	t_i	E	P	S6
			Cirujano responsable	γ_i	O	P	S6
			Compatibilidad	δ_{ijh}	B	C	S6
Quirófano	$j = 1 \dots J$	I_u	Capacidad quirófano – día	r_{jh}	E	C	480
			Compatibilidad	δ_{ijh}	B	C	S6
Cirujano	$s = 1 \dots S$	I_u	Capacidad cirujano – día	a_{sh}	E	C	{0,480}
			Máximo de quirófanos	u_s	E	P	S6

En la Tabla 2–3, se muestra una leyenda de las anteriores notaciones para una mejor comprensión de las mismas.

Tabla 2–3. Leyenda para tabla de elementos

ELEM	Elemento
CJTO	Conjunto
NC	Naturaleza cuantitativa
TV	Tipo de valor
P	Pertenencia
I_u	Individual Unitario
E	Entero
O	Ordinal
B	Binario
P	Propio
C	Compartido
S6	Valores introducidos en 6. Banco de pruebas

2.3 Variables del modelo ILP

Adicional a los elementos del problema se nos presenta una serie de variables, resultado de modelar las actividades de decisión del problema, las cuales se muestran a continuación.

1º Asignar cirugía a día – quirófano.

$\forall i, \forall j, \forall h: X_{ij} =$ 1 si el paciente i es asignado a la sala de operaciones j en el día h; 0 en caso contrario. $(\forall i \in I, \forall j \in J, \forall h \in H: X_{ijh} \in \{0,1\})$

2º Asignar cirujano responsable de la cirugía a día – quirófano.

$\forall s, \forall j, \forall h: Z_{sjh} =$ 1 si el cirujano s realiza cirugía en la sala de operaciones j en el día h: 0 en caso contrario. $(\forall s \in S, \forall j \in J, \forall h \in H: Z_{ijh} \geq 0)$

2.4 Restricciones y función objetivo del modelo aproximado

Por último, se presenta la función objetivo y las restricciones del modelo, generadas a partir del objetivo propuesto y las especificaciones impuestas del problema, respectivamente. Para ello, a continuación, se muestra el modelo junto a una serie de explicaciones, para una mejor comprensión del mismo.

Tabla 2–4. Modelo ILP

$$LEX\ (\max f_1(x), \min f_2(x)) \tag{2–1}$$

Subject to:

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

$$\sum_{j \in J} \sum_{h=1}^{rd_i - 1} X_{ijh} = 0 \quad (\forall i \in I) \tag{2–3}$$

$$\sum_{j \in J} \sum_{\substack{h \in H \\ h \leq d_i}} X_{ijh} = 1 \quad (\forall i \in I | d_i \leq |H|) \tag{2–4}$$

$$\sum_{i \in I} t_i X_{ijh} \leq r_{jh} \quad (\forall j \in J, \forall h \in H) \tag{2–5}$$

$$\sum_{j \in J} \sum_{\substack{i \in I \\ \gamma_i = s}} t_i X_{ijh} \leq a_{sh} \quad (\forall s \in S, \forall h \in H) \tag{2–6}$$

$$\sum_{j \in J} Z_{sjh} \leq u_s \quad (\forall s \in S, \forall h \in H) \tag{2–7}$$

$$\sum_{\substack{i \in I \\ \gamma_i = s}} t_i X_{ijh} \leq r_{jh} Z_{sjh} \quad (\forall s \in S, \forall j \in J, \forall h \in H) \tag{2–8}$$

$$\sum_{\substack{i \in I \\ \gamma_i = s}} t_i X_{ijh} \geq Z_{sjh} \quad (\forall s \in S, \forall j \in J, \forall h \in H) \tag{2–9}$$

$$X_{ijh} = 0 \quad (\forall s \in S, \forall j \in J, \forall h \in H | \delta_{ijh} = 0) \tag{2–10}$$

$$X_{ijh} \in \{0,1\} \quad (\forall i \in I, \forall j \in J, \forall h \in H) \tag{2–11}$$

$$Z_{sjh} \in \{0,1\} \quad (\forall i \in I, \forall j \in J, \forall h \in H) \tag{2–12}$$

La función objetivo (2–1) maximiza el número de cirugías programadas ponderado por el peso de las mismas y minimiza el número de movimientos totales realizados por los cirujanos, correspondiente al cambio de quirófano en un mismo día para realizar las cirugías asignadas. Dado que dicho estudio no profundiza en la

optimización de problemas multiobjetivo, se define el objetivo del total de cirugías programadas ponderado como el principal, y se acudirá al secundario, relativo al total de movimientos realizados por los cirujanos en caso de empate del anterior.

La restricción (2–2) garantiza que cada cirugía sea programada como máximo una vez durante todo el horizonte de planificación y en único quirófano.

Las restricciones (2–3) y (2–4) garantizan el complimiento de la fecha más temprana y más tardía respectivamente en la que el paciente i puede ser programado. Prohibiendo la (2–3) que la cirugía del paciente se programe en un día h anterior al de la fecha de inicio (rd_i), y la (2–4) que tenga lugar, anterior a su fecha de vencimiento (d_i).

La restricción (2–5) prohíbe que la duración total de las cirugías asignadas a un día-quirófano supere su capacidad regular (r_{jh}).

La restricción (2–6) prohíbe que la duración total de las cirugías asignadas a un cirujano en un día, sea mayor que su capacidad regular (a_{sh}).

La restricción (2–7) limita la cantidad de quirófanos asignados a un cirujano, en un mismo día, al número máximo de cirugías que éste puede realizar en un mismo día (u_s).

Las restricciones (2–8) y (2–9) definen si un cirujano es asignado a un día-quirófano asegurando la no superposición de cirugías realizadas por el mismo.

La restricción (2–10) asegura que cada cirugía se realice en un día – quirófano adecuado, de acuerdo a la admisibilidad de los mismos (δ_{ij}).

Por último, las restricciones (2–11) y (2–12) son relativas a tipo de valor de las variables de asignación, (X_{ijh} y Z_{sjh}), antes explicadas.

Una vez detallado el problema objeto de estudio, así como el modelo ILP generado a partir del mismo, siendo este la base del estudio; se procede a explicar en los siguientes capítulos la resolución del mismo, la cual, al ser aproximada, repercute en criterios adicionales a considerar.

3 DECODIFICACIÓN DE LA SOLUCIÓN

Para el desarrollo del estudio de optimización con métodos aproximados, es abordado, en primer lugar, la decodificación de las soluciones al problema, la cual se trata del conjunto de variables y restricciones antes introducidas establecidas de forma codificada, permitiendo, por un lado, evaluar las distintas configuraciones de los valores que dichas variables toman e interpretando a su vez las restricciones impuestas. Todo ello con el fin de aportar soluciones admisibles al problema junto a los valores de las funciones objetivo que estas proporcionan, para medir la calidad de la misma.

Dicha fase es de crucial importancia, ya que aparte de garantizar la admisibilidad de las soluciones generadas, afecta en gran medida a la eficiencia computacional de los algoritmos propuestos, dado que es la etapa del mismo que más se repite, es decir, su nivel más interno.

Para ello, se ha elaborado el código a través de Python, el cual debe admitir el acceso a los datos, la interpretación de las restricciones y la correcta actualización de las variables del modelo antes presentado, así como de otras adicionales, implementadas debido al tipo de resolución aproximada, así como su programación realizada.

La estructura del capítulo se detalla a continuación.

En el subcapítulo 3.1 se presenta los datos y variables antes definidos en el modelo para resolución exacta atendiendo a sus características para resolución aproximada, así como la definición de nuevas variables y parámetros implementados.

En el subcapítulo 3.2 se presentan las distintas decodificaciones implementadas, así como sus principales características y estructuras, junto a pasos y cálculos de importancia durante las mismas.

Finalmente, en el subcapítulo 3.3 se muestran ejemplos ilustrativos de la representación de la solución planteada, para una mejor interpretación de la misma.

3.1 Declaración de datos y variables

En este subcapítulo se introducen otros datos adicionales necesarios, por el hecho de usar para la resolución del problema métodos aproximados, habiendo factores adicionales a considerar con respecto a la anterior descripción y modelo ILP, correspondiente a una resolución exacta. Por otro lado, se les dedica un apartado propio, y referido a su declaración, usada para su inicialización previa al uso de las decodificaciones y formalizada en los distintos modelos implementados.

Se determina un índice a modo de resumen en Tabla 2–1, con una breve descripción de las nuevas variables y parámetros implementados, así como su notación formalizada en todo el documento

Tabla 3–1 Resumen de datos y variables adicionales para los modelos aproximados

Conjuntos, datos y variables adicionales implementados en los algoritmos aproximados	
Índices y conjuntos	
$h \in H$	Índice y conjunto de los días dentro del horizonte de planificación
$i \in I$	Índice y conjunto de los pacientes en la lista de espera
$j \in J$	Índice y conjunto de las salas de operaciones del área
$s \in S$	Índice y conjunto de los cirujanos del área
t, n	Índice y tamaño de los intervalos en la lista de intervalos libres
Parámetros	
π	Secuencia de cirugías
OF1	Valor objetivo de la función objetivo 1
OF2	Valor objetivo de la función objetivo 2
ϕ	Una solución en forma de tupla formada por una secuencia y dos valores objetivo
u	Un intervalo de la lista de intervalos libres
Variables	
X_{ijh}	Variable binaria que toma 1 si el paciente i es asignado a la sala de operaciones j en el día h; 0 en caso contrario
Z_{sjh}	Variable binaria que toma 1 el cirujano s realiza cirugía en la sala de operaciones j en el día h; 0 en caso contrario
c_{jh}	*Completion Time* de la sala de operaciones j en el día h
c_{sh}	*Completion Time* de cirujano s en el día h
c_{sjh}	*Completion Time* de cirujano s en la sala de operaciones j en el día h
r_{sjh}	*Release Date* de cirujano s en la sala de operaciones j en el día h
d_{sjh}	*Due Date* de cirujano s en la sala de operaciones j en el día h
c_{ijh}	*Completion Time* de cirugía i en la sala de operaciones j en el día h
r_{ijh}	*Release Date* de cirugía i en la sala de operaciones j en el día h
d_{ijh}	*Due Date* de cirugía i en la sala de operaciones j en el día h
$intervals$	Matriz con listas de intervalos libres para cada cirujano s y cada día h
$movements$	Matriz con movimientos totales realizados por cada cirujano s en cada día h
$last\_OR$	Matriz con índices de los últimos quirófanos asignados a cada cirujano s en cada día h
JH	Lista de tuplas formadas por todas las combinaciones posibles de j y h
$available\_time$	Matriz con tiempos disponible en cada sala j para cada día h
$available\_tuple$	Lista de tuplas formadas por el tiempo disponible de cada conjunto día – quirófano y sus respectivos índices.

3.1.1 Datos para el algoritmo aproximado

Para el caso de los datos, no se da ninguno adicional, siendo todos ellos coincidentes con los introducidos anteriormente. Por ello se dedica el apartado para tratar cuestiones relativas a su representación, valor y características enfocadas a la resolución aproximada.

Debido a que los valores que toman son arbitrarios en general, exceptuando los que imponga el problema real, como es el caso del número de quirófanos del hospital, el número de cirujanos o la capacidad de los mismos, y

que a su vez toman múltiples valores en la etapa de pruebas, la idea es buscar una declaración que aparte de garantizar una solución factible, nos de flexibilidad en su uso. Tratándose estas principalmente de vectores y matrices de valores y conjuntos de valores, dada su facilidad de representación e interpretación en el lenguaje de programación empleado, gracias a objetos como las listas y las tuplas.

Para el caso del problema planteado, se nos presentan una serie de datos cuyo valor representa un entorno realista, asemejándose a artículos similares de la literatura. Por ello algunos datos serán por un lado deterministas como puede ser la cantidad de quirófanos, y otros totalmente estocásticos, como el caso del número de pacientes en la lista de espera o la duración de las cirugías, factor que respalda aún más la necesidad de generar múltiples instancias y hecho tratado más en detalle en 6. Banco de pruebas.

Relativo a la implementación en Python y de forma resumida, para la inicialización de los conjuntos (I, J, H y S) y sus atributos recién explicados se ha hecho uso del método *__init__*, el cual permite su llamada de forma automática al momento de generar una nueva instancia de la clase relativa al modelo y construcción de la misma inicializando los atributos del nuevo objeto generado. De este modo, se da la seguridad, de que el objeto relativo al modelo, siempre posea las mismas características, es decir mismo tipo de atributos, para cada una de las instancias generadas. De igual forma, los datos son inicializados como objeto *self*, para asegurar que el resto del código trabaje posteriormente con los definidos en la instancia actual, simplificando a su vez su acceso, al ser manejados de forma interna por Python, evitando ser llamados de forma individual uno a uno en el momento de usarlos.

Adicional a los datos anteriormente definidos, se generan una serie de variables adicionales, las cuales almacenen mediante una copia profunda en memoria, las listas de cada uno de los conjuntos (*I_original*, *J_original*, etc.). Su propósito es el de proporcionar en todo momento la lista original inicializada, para su recuperación, en caso de modificación de las normales en alguna parte del algoritmo.

3.1.2 Variables de asignación para el algoritmo aproximado

A diferencia de los datos de entrada; los cuales toman valores fijos durante la optimización, en el caso de las variables, conocidas también como elementos de decisión, se tiene la posibilidad de ajustar dicho valor. Esto es necesario para aportar control sobre la optimización, permitiendo realizar la programación y generación de la solución, y, por ende, encontrar posteriormente los valores que maximicen y minimicen las funciones objetivo del problema, es decir, aquellas que más se acerquen al óptimo.

A partir de dicho subcapítulo, se introducen las variables de asignación, antes mencionadas, variables de decisión del modelo, así como otras necesarias para la optimización a través de métodos aproximados, como son los *Completion Time*, las almacenadoras de valores para el correcto cálculo de la función objetivo y la solución generada, así como otras para garantizar el correcto funcionamiento de una decodificación concreta.

Relativo a las variables de decisión, denominadas *Variables de asignación*, por el contexto de las mismas, se encuentra:

- Variable de asignación de cirugías a conjuntos día – quirófano (X_{ijh}): Como ya ha sido comentado, se trata de una variable de decisión de cuantificación binaria, es decir, toma valor 1 si la cirugía i se asigna al quirófano j y al día h y 0 en caso contrario. Por lo que se trata de una matriz tridimensional de tamaño $|I| * |J| * |H|$.
  Su implementación radica en dar posibilidad durante la programación de cirugías, de contabilizar las asignaciones producidas de las mismas, permitiendo el cumplimiento iterativo de especificaciones relativas a la realización de la cirugía únicamente en un día – quirófano, evitando así soluciones inadmisibles, como realizar una misma cirugía en varios días o quirófanos distintos. Por ello su uso se da en la correspondiente restricción en cuestión, condicional en la decodificación; y la actualización de su valor, tras programar con éxito la cirugía planificada, es decir, cumplir el conjunto de condicionales por parte de la misma.
- Variable de asignación de cirujanos a conjuntos día – quirófano (Z_{sjh}): De forma análoga a la anterior, se implementa una variable para la contabilización, en este caso, de las asignaciones de los

cirujanos a conjuntos días – quirófanos. La representación de dicha variable es por tanto una matriz tridimensional de tamaño $|S| * |J| * |H|$.
Su implementación radica, en contabilizar dichas asignaciones, y es necesaria debido a especificaciones relativas a la no superposición de las cirugías que los cirujanos realizan, es decir, imposibilitar soluciones donde el cirujano realice dos cirugías al mismo tiempo, así como la relativa al número máximo de quirófanos en los que un cirujano puede operar en un mismo día. Su implementación por tanto se da en las restricciones en cuestión y su actualización al igual que la anterior variable, tras realizar una programación.

Relativo a sus declaraciones y debido al funcionamiento de la posterior decodificación, se inicializan con valor 0 en cada una de sus posiciones.

3.1.3 Variables relativas a los tiempos de finalización

Por otro lado, y como ya ha sido comentado, ha sido necesario implementar una serie de variables relativas a los *Completion Time* o tiempos de finalización. Dichos *Completion Time* se tratan de un concepto clave y frecuente en problemas de optimización sobre programación de tareas (*Scheduling Problems*), los cuales se refieren al tiempo de finalización de las mismas, y se usan para minimizar tiempos totales, de retraso, etc. Y, por ende, mejorar la productividad del sistema estudiado. En el caso de la optimización quirúrgica, dichas tareas son la realización de las distintas cirugías, y su implementación se realiza para lograr el cumplimiento de algunas de las restricciones del modelo, dado que los objetivos tratados no hacen referencia a ninguna métrica temporal.

Antes de proceder a su explicación de forma individual, comentar que, para el modelo original basado en el modelo ILP presentado, se ha trabajado en paralelo con una ampliación del mismo, la cual incorpora una heurística interna en la propia decodificación, relacionada con el tipo de asignación y programación. Dicha versión además ha implicado, la modificación de una serie de restricciones y variables relacionadas con los *Completion Time*, así como la incorporación de alguna propia para garantizar su correcto funcionamiento, motivo por el que se han clasificado las siguientes variables según el modelo donde se implementan

En primer lugar, y relativo a las versiones originales del modelo, los *Completion Time* definidos han sido enfocados al elemento *Cirujano*. Dichos Completion time son:

- *Completion Time* de cirujano – día (c_{sh}): Variable que almacena el tiempo de finalización de cada conjunto día – cirujano, y es usada para el cumplimiento de las restricciones relativas a la no superposición de cirugías realizadas por el mismo.
- *Completion Time* de quirófano – día (c_{jh}): Variable que almacena el tiempo de finalización de cada conjunto quirófano – día y es usada para el cumplimiento de las restricciones relativas a su capacidad.
- *Completion Time* de cirujano – quirófano – día (c_{sjh}): Implementado en el código, ya que con una serie de modificaciones en la restricción es equivalente a los dos anteriores, reduciendo así el número de variables con las que trabajar.

Las fórmulas (3–1) y (3–2) muestran la equivalencia mencionada.

$$\forall s \in S, \forall\, h \in H: c_{sh} = \max_{j \in J} c_{sjh} \tag{3–1}$$

$$\forall j \in J, \forall h \in H: c_{jh} = \max_{s \in S} c_{sjh} \tag{3–2}$$

Dichas variables relativas a los Completion Time son actualizados sumando de forma acumulada el tiempo de la cirugía programada e implementados en cada una de los condicionales motivos de su implementación. Por otro lado, y necesario para el correcto cálculo de los mismos, es necesaria la implementación de otra serie de variables, las cuales almacenan los tiempos de inicio y finalización de las cirugías realizadas por los distintos cirujanos.

- *Release Date* de cirujano – quirófano – día (r_{sjh}): Variable que almacena el instante de inicio de la última cirugía realizada por cada cirujano para cada conjunto quirófano – día.
- *Due Date* de cirujano – quirófano – día (d_{sjh}): Variable que almacena el instante de finalización de la última cirugía realizada por cada cirujano para cada conjunto quirófano – día.

Para las otras versiones del modelo, relativas al uso de la implementación antes mencionada, se ha enfocado los *Completion Time* al elemento *Cirugía*. A su vez ha involucrado modificaciones en los condicionales, relativos a las restricciones antes mencionadas.

- *Completion time* de cirugía – quirófano – día (c_{ijh}): Variable que va almacenado los tiempos de finalización de cada cirugía en cada quirófano – día. El motivo de su uso en dicha versión, es el hecho de que el anterior almacena el último instante de finalización del cirujano para un conjunto día – quirófano concreto, imposibilitando la asignación de cirugías en intervalos anteriores a dicho Completion Time, que han podido quedar libres y disponibles para la programación de cirugías para un determinado cirujano, posible limitación implementada y analizada mediante las versiones del modelo correspondientes a la misma.

De forma análoga al anterior, es necesario para su correcto cálculo el uso de los tiempos de inicio y finalización de cada una de las cirugías.

- *Release date* de cirugía – quirófano – día (r_{ijh}): Variable que almacena el instante de inicio de cada cirugía para cada uno de los conjuntos quirófano – día.
- *Due date* de cirugía – quirófano – día (d_{ijh}): Variable que almacena el instante de finalización de cada cirugía en cada uno de los días quirófano – día.

A continuación, se presenta una serie de pasos relativos a las principales características de la actualización de las variables recién presentadas para el caso del modelo original, relativo a los *Completion time* en base al elemento *Cirujano*.

Paso 1: Programación con éxito de la cirugía i relizada por el cirujano s en el día j y el día h, y, por ende, almacenada en la secuencia de cirugías programadas π, donde el cirujano s es igual al cirujano responsable definido en γ_i.

Paso 2: Cálculo del *Release Date*, es decir, tiempo de inicio de dicha cirugía i en el día h y quirófano j, en base a los *Completion Time*, de otras cirugías ya programadas, para la construcción de un plan factible, según se muestra en la fórmula (3–3), donde el tiempo de inicio es el máximo entre el máximo *Completion Time* de todos los cirujanos para el quirófano j y día h, seleccionados y el máximo de los *Completion Time* para el cirujano s en el día h en cuestión. De este modo se tiene en cuenta tanto cirugías ya programadas en ese día – quirófano, como cirugías ya programadas para el cirujano responsable durante ese día, evitando superposición de cualquier tipo.

$$\forall s \in S, \forall h \in H, \forall j \in J : r_{sjh} = \max(\max_{s \in S} c_{sjh}, \max_{j \in J} c_{sjh}) = \max(c_{sh}, c_{jh}) \quad (3\text{–}3)$$

Paso 3: Actualización de las variables relativas a los *Due Date* y los *Completion Time*, para siguientes iteraciones, sumando al *Release Date*, el tiempo de duración de la cirugía programada.

3.1.4 Variables específicas de decodificaciones

En dicho subcapítulo se presentan una serie de variables específicas para ciertas decodificaciones implementadas, de necesario uso para lograr el funcionamiento esperado de las mismas. Más en concreto se trata de las pertenecientes a la decodificación especializada en aprovechar los intervalos que van quedando libres durante la programación para los cirujanos en un cierto día del horizonte de planificación, y las relativas a dos tipos de asignaciones, basadas en asignar cirugías a los conjuntos día – quirófano, con más y menos tiempo disponible. Respectivamente. El funcionamiento de dichas decodificaciones ha sido tratado, en más detalle, en el subcapítulo, 3.2 Decodificaciones.

Relativo a la nueva implementación antes mencionada, se define una variable denominada $intervals$, cuyo objetivo es el de ir almacenando los intervalos de tiempo que van quedando libres para cada cirujano (s) de la unidad en cada día (h) del horizonte de planificación, con el objetivo de programar cirugías en dichos intervalos de tiempo, buscando solucionar ese posible desaprovechamiento de recursos de los modelos de la versión original. A continuación, se introduce dicha variable, así como una descripción de su funcionamiento para una mejor comprensión de la misma.

- Intervalos libres de cirujano s en día h ($intervals_{sh}$): Dicha variable se trata de una matriz de tamaño $S * H$, la cual contiene en cada una de sus posiciones listas de intervalos, formados por el instante de inicio y finalización, referidos a tiempos que quedan libres de asignaciones durante la programación de las cirugías en los distintos conjuntos día – quirófano.

Para una mejor comprensión de dicha variable se procede a explicar su funcionamiento, así como los cálculos principales que lleva asociada, en una serie de pasos. Adicionalmente se muestra los valores que va tomando dicha variable en las sucesivas iteraciones durante la programación de una instancia dada, en **Ejemplo 3–1.** *Funcionamiento de la variable intervals dentro de la decodificación*

Paso 1: La variable es inicializada con un único intervalo en cada una de sus posiciones, con valores el instante inicial de la programación para el extremo inferior (0) y el definido por la capacidad máxima para todos los quirófanos, del día (h), en cuestión, como se muestra en (3–4).

$$\forall s \in S, \forall h \in H: intervals_{sh} = [0, \max_{j \in J}(r_{jh})] \tag{3–4}$$

Es decir, si la capacidad máxima permitida es de 480 (instante límite de la jornada para la programación de cirugías), el hueco inicial para esa jornada (h) y el cirujano (s) en cuestión será igual a $[0{,}480]$.

Paso 2: Al momento de ir programando cirugías, la matriz $huecos$ irá tomando la forma de $\left[[[u_1, u_2, \dots, u_t, u_n], [u_1, u_2, \dots, u_t, u_n]], \dots, [[u_1, u_2, \dots, u_t, u_n]]\right]$ donde t es el índice del intervalo y n el número total de intervalos registrados para el cirujano responsable en el día en cuestión, con su correspondiente tiempo de inicio u_{t0} y finalización u_{t1}, $u_t = [u_{t0}, u_{t1}]$, motivo que provoca que la variable se transforme en una matriz cuatridimensional. Esto es debido a que la actualización incluye la modificación de uno de los intervalos y la inserción de uno nuevo, posterior al mismo. De este modo la lista de intervalos resultante de secuenciar la cirugía i con cirujano responsable s en el día h, $intervals_{sh/s=\gamma_i} = [u_1, u_2, \dots, u_n]$ pasa a ser $intervals'_{sh/s=\gamma_i} = [u'_1, u'_2, \dots, u'_{n+1}]$, donde:

$$u_t{}' = \begin{cases} u_t \;\; si \;\; 1 \le t < k \\ [u_{t0}, r_{ijh}] \;\; si \;\; t = k \\ [d_{ijh}, u_{t1}] \;\; si \;\; t = k+1 \\ u_{t-1} \;\; si \;\; k+1 < t \le n+1 \end{cases} \tag{3–5}$$

Siendo k la posicion del primer intervalo u de la lista de intervalos en cuestión, cuya diferencia entre su extremo superior e inferior es mayor al tiempo de duración de la cirugía programada.

Ejemplo 3–1. *Funcionamiento de la variable intervals dentro de la decodificación*

El entorno de la instancia generada se trata de la programación de una lista de espera formada por 20 cirugías, teniendo como recursos 5 quirófano y 2 cirujanos, y definiendo el horizonte de planificación en 5 días. Por otro lado, la capacidad de los quirófanos y de los cirujanos es coincidente y de valor 390 minutos (8:30 – 14:00), y datos como el *Release Date* de las cirugías (rd_i) o la admisibilidad entre cirugía – quirófano – día (δ_{ij}) se definen lo menos restrictivas posibles, así como, de igual valor para todas sus combinaciones. Todo ello para generar una instancia que permite una mejor visualización de lo planteado. De este modo los datos de entrada resultantes son los siguientes.

$I = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$J = [1, 2, 3, 4, 5]$

$H = [1, 2, 3, 4, 5]$

$S = [1, 2]$

$t_i = [157, 112, 93, 83, 184, 78, 143, 60, 131, 187]$

$\gamma_i = [1, 2, 1, 2, 1, 2, 1, 2, 1, 2]$

A continuación, en Tabla 3–2 se representa la programación iterativa realizada, donde para cada iteración, es decir, planificación de la cirugía, se muestra la modificación que conlleva su programación, así como la modificación de la variable $intervals_{sh}$ resultante, marcado de color rojo en la representación de la misma. Todo ello con el objetivo de representar el proceso que se realiza de forma interna en las decodificaciones. Dicha modificación irá acompañada de una ilustración representativa del programa quirúrgico elaborado, para una mayor comprensión de la misma.

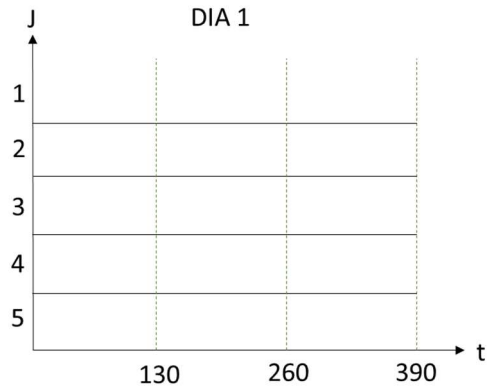


Ilustración 3–1. Representación del programa quirúrgico para un día genérico

Como se observa en Ilustración 3–1, el diagrama utilizado, se caracteriza por representar en el eje horizontal, el tiempo de la jornada (t) en minutos, y en el eje vertical, cada uno de los quirófanos donde serán programadas las cirugías. Adicionalmente se visualiza como el instante límite es el de la capacidad detallada. Mencionar que hay tantos diagramas de Gantt, como días tenga el horizonte de planificación.

Ilustración 3–2. Representación genérica de una lista de espera de pacientes

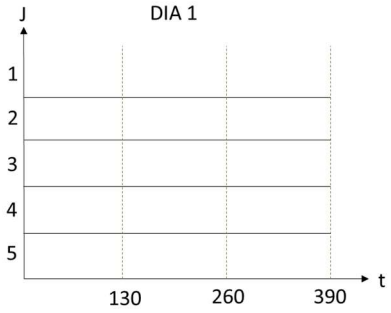
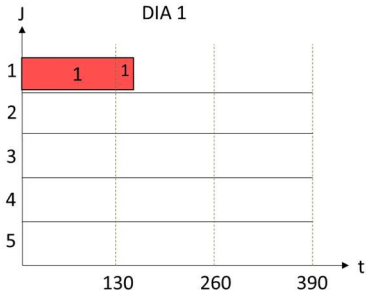
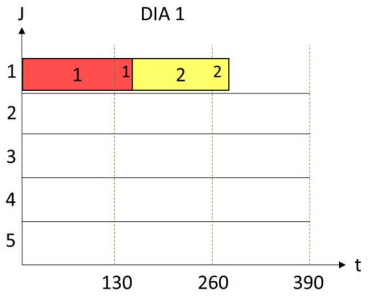
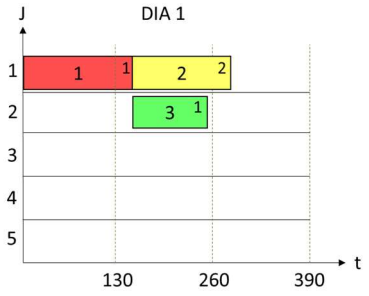
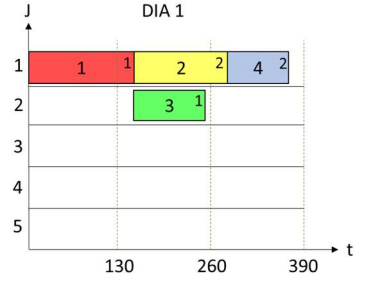
Como se observa en Ilustración 3–2, cada uno de los rectángulos representan una cirugía, marcado con su etiqueta en el centro, y con la del cirujano responsable en su esquina superior derecha. En la posterior representación de la programación iterativa (), se ve reflejada su duración en la longitud de su base.

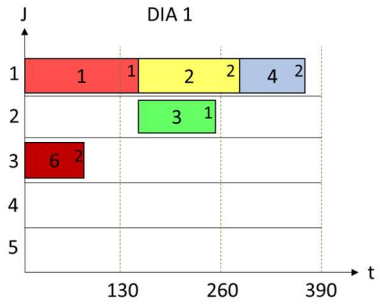
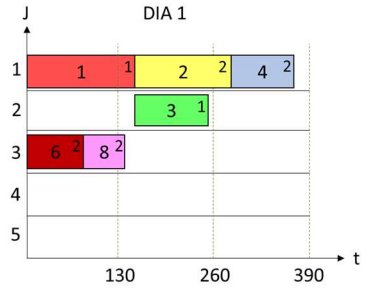
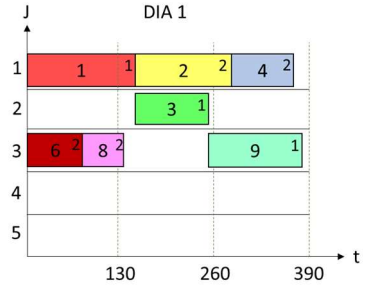
$$intervals_{sh} =$$
$$\text{(s = 1)}\ [[[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]],$$
$$\text{(s = 2)}\ [[[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]$$
$$\text{(h = 1)} \quad \text{(h = 2)} \quad \text{(h = 3)} \quad \text{(h = 4)} \quad \text{(h = 5)}$$

Ilustración 3–3. Representación de variable *intervals* en el momento de su inicialización

Por último, en Ilustración 3–3, se presenta la matriz correspondiente a la variable $intervals_{sh}$, en formato de lista de listas, similar a su representación en la consola del entorno de Python usado. Como puede verse reflejado es de tamaño $S * H$, y en cada una de sus posiciones hay una lista con un intervalo en su interior, es decir, la lista de huecos antes explicada en su forma inicializada. Dicho intervalo se va modificando y la lista que lo contiene almacenando los distintos intervalos generados durante las iteraciones de la programación. A continuación, se muestra la tabla para la solución obtenida para dicho ejemplo, donde cada fila es una iteración o paso (η), y las distintas columnas que contiene, la cirugía planificada (i), su duración (t_i), la modificación del diagrama y la modificación de la variable que está siendo estudiada ($intervals_{sh}$), tras su programación.

Tabla 3–2. Representación de la programación en base a la matriz *intervals* de forma iterativa

η	i	t_i	Diagrama resultante	Actualización de $intervals_{sh}$
#	-	-		$intervals_{sh} =$ (s = 1) [[[[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], (s = 2) [[[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]] (h = 1) (h = 2) (h = 3) (h = 4) (h = 5)
1	1	157		$intervlas_{sh} =$ [[[[0, 0], [157, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]
2	2	112		$intervals_{sh} =$ [[[[0, 0], [157, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 157], [269, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]
3	3	93		$intervals_{sh} =$ [[[[0, 0], [157, 157], [250, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 157], [269, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]]
4	4	83		$intervals_{sh} =$ [[[[0, 0], [157, 157], [250, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 157], [269, 269], [352, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]]

<table>
<tr><td>5</td><td>5</td><td>184</td><td colspan="2">Ningún intervalo admisible: u de mayor duración en $intervals_{11} < t_5$ $(140 < 184)$ *</td></tr>
<tr><td>6</td><td>6</td><td>78</td><td> 
</td><td>$intervals_{sh} =$
[[[[0, 0], [157, 157], [250, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 0], [78, 157], [269, 269], [352, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]]</td></tr>
<tr><td>7</td><td>7</td><td>143</td><td colspan="2">Ningún intervalo admisible: u de mayor duración en $intervals_{11} < t_7$ $(140 < 143)$ *</td></tr>
<tr><td>8</td><td>8</td><td>60</td><td> 
</td><td>$intervals_{sh} =$
[[[[0, 0], [157, 157], [250, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 0], [78, 78], [138, 157], [269, 269], [352, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]]</td></tr>
<tr><td>9</td><td>9</td><td>131</td><td> 
</td><td>$intervals_{sh} =$
[[[[0, 0], [157, 157], [250, 250], [381, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]], [[[0, 0], [78, 78], [138, 157], [269, 269], [352, 390]], [[0, 390]], [[0, 390]], [[0, 390]], [[0, 390]]]]</td></tr>
<tr><td>10</td><td>10</td><td>187</td><td colspan="2">Ningún intervalo admisible: u de mayor duración en $intervals_{21} < t_{10}$ $(38 < 187)$ *</td></tr>
<tr><td colspan="5">*Dichas cirugías, en realidad, sí están siendo programadas en el día 2 del horizonte de planificación, modificándose por tanto la segunda lista para cada uno de los cirujanos de la variable $intervals_{sh}$. En dicho ejemplo no está siendo reflejado, pues basta centrarlo en el día 1 para analizar lo relativo a la actualización de la variable, siendo similar para el resto de días.</td></tr>
</table>

Como puede ser observado, la implementación de dicha variable y de la nueva versión de la decodificación, no sólo cumple con la restricción relativa a evitar la superposición de cirugías para un mismo cirujano en cada día del horizonte; papel que ya permite tanto el c_{sj} de las versiones originales, como el c_{ij} usado en la actual, sino también, permitir al algoritmo programar en intervalos libres que puedan irse generando para un día en concreto, situación que únicamente con los *Completion Time* no sería posible, pues a la hora de implementarlos en sus correspondientes restricciones y actualizarlos dentro de la decodificación se evalúa el máximo de éstos, descartando la posibilidad de aprovechar alguno de estos intervalos generados.

A continuación, se muestra una representación para dos ejemplos, que permiten visualizar y comparar los resultados de usar o no dicha implementación, tanto para un caso donde es favorable su implementación mostrado en, **Ejemplo 3–2.** *Caso favorable de la decodificación que implementa la variable relativa a los intervalos libres de los cirujanos*. Como para otro donde es desfavorable, mostrado en **Ejemplo 3–3.** *Caso desfavorable de la decodificación que implementa la variable relativa a los intervalos libres de los cirujanos.*

Ejemplo 3–2. *Caso favorable de la decodificación que implementa la variable relativa a los intervalos libres de los cirujanos.*

El entorno de la instancia generada se trata de una lista de pacientes ordenados en sentido ascendente de tamaño 10, para cuya programación se dispone de 5 quirófanos, 2 cirujanos y 1 día como horizonte de planificación. Los valores de las variables relativas a la capacidad (r_{sh} y a_{sh}) toman valor 480 min (8:30 - 14:00) y otros datos, como la admisibilidad entre cirugía – quirófano – día o los relativos a los tiempos más tempranos y tardíos son lo mínimo restrictivos posible. Por tanto, los datos a considerar son los siguientes.

$I = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$J = [1, 2, 3, 4, 5]$

$H = [1]$

$S = [1, 2]$

$t_i = [120, 120, 120, 120, 120, 120, 120, 120, 120, 120]$

$\gamma_i = [1, 2, 1, 2, 1, 2, 1, 2, 1, 2]$

De esta forma obtenemos un resultado distinto para el caso anterior a la nueva implementación, frente al caso actual donde si se hace uso de la misma, mostrado en Ilustración 3–4.

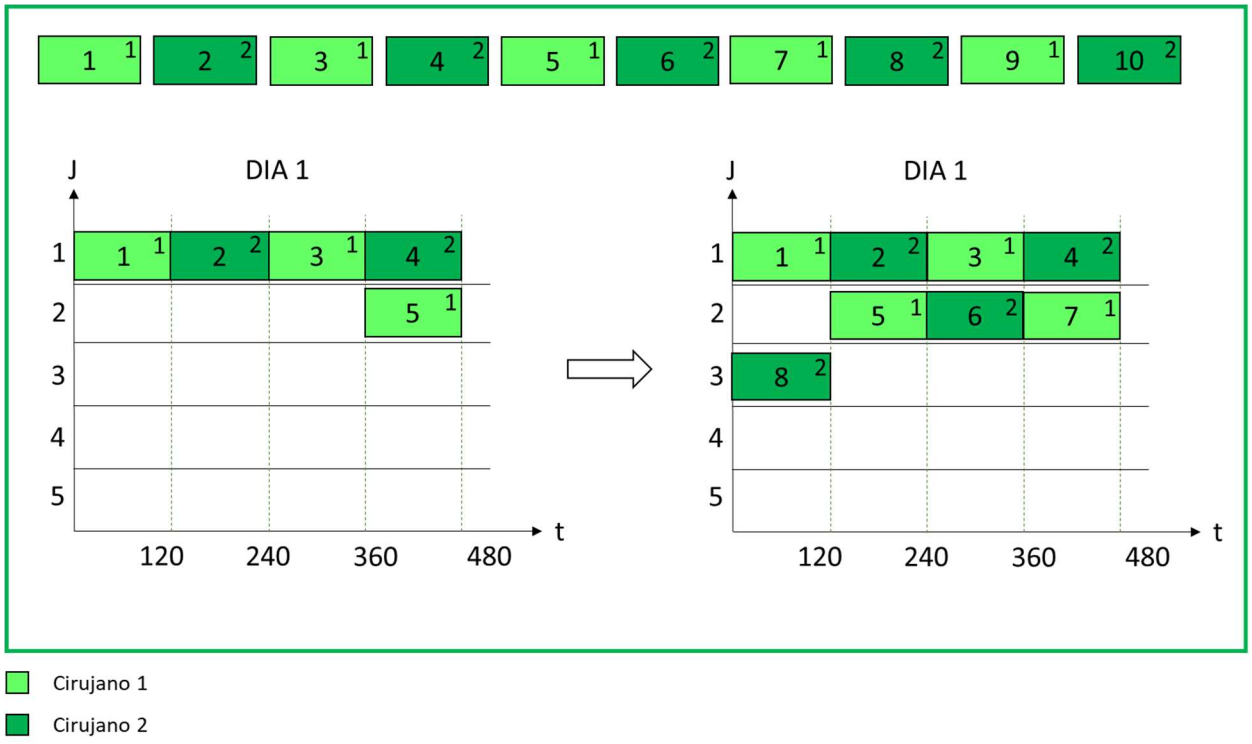


Ilustración 3–4. Caso favorable para la implementación de *intervals*

Ejemplo 3–3. *Caso desfavorable de la decodificación que implementa la variable relativa a los intervalos libres de los cirujanos.*

El entorno de la instancia generada se trata de una lista de pacientes ordenados en sentido ascendente de tamaño 10, para cuya programación se dispone de 2 quirófanos, 3 cirujanos y 1 días como horizonte de planificación. Los valores de las variables relativas a la capacidad (r_{sh} y a_{sh}) toman valor 480 min (8:30 - 14:00) y otros datos, como la admisibilidad entre cirugía-quirófano-día o los relativos a los tiempos más tempranos y tardíos son lo mínimo restrictivos posible. Por tanto, los datos a considerar son los siguientes.

$I = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$J = [1, 2]$

$H = [1]$

$S = [1, 2, 3]$

$t_i = [160, 160, 160, 320, 160, 64, 64, 64, 64, 64]$

$\gamma_i = [1, 1, 2, 2, 3, 3, 3, 3, 1, 1]$

De forma análoga se obtiene una solución distinta para ambas decodificaciones, mostrado en Ilustración 3–5.

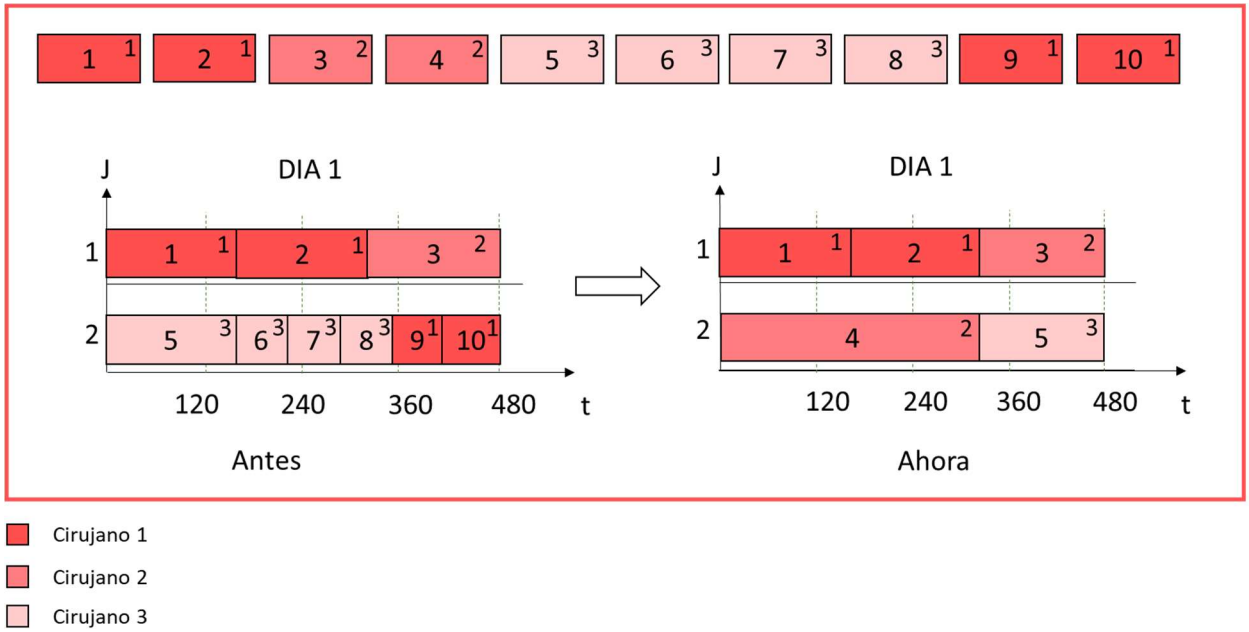


Ilustración 3–5. Caso desfavorable para la implementación de *intervals*

Relativo a las principales diferencias entre el modelo original, así como el nuevo que incorpora esta heurística interna, destacan que el uso de las variables $intervals_{sh}$ nos permiten prescindir en la decodificación de los condicionales relativos a las restricciones (2–8) y (2–9), sobre la no superposición de cirugías programadas a cirujanos, ya que los condicionales relativos a la misma del modelo original, son sustituidos por el condicional relativo a la nueva variable en el nuevo modelo, el cual verifica que alguno de los huecos del cirujano responsable de la cirugía analizada sea mayor que el tiempo de duración de la misma.

Del mismo modo, el valor que toma r_{ijh} en cada iteración, se ve modificado con respecto a r_{sj} , mostrado en (3–3). El cual se adapta a dicha implementación, mediante el máximo entre el máximo *Completion Time* para el día – quirófano programado y el instante inicial del intervalo seleccionado, evitando de forma similar, las superposiciones relativas a cirugías dentro de un mismo quirófano y las realizadas por cirujanos en un mismo día.

Finalmente, y según lo observado en los ejemplos mostrados sobre la nueva implementación, se puede llegar a la conclusión de que es favorable de usar con respecto a la original para casos muy concretos, sumado, a que también puede ser desfavorable para otras instancias, el elevado tamaño de las mismas en la realidad, así como en el banco de pruebas diseñado, y además, el significativo tiempo adicional con respecto a la versión original que requiere, fruto de la incorporación de niveles adicionales de anidación en la decodificación, para su correcto funcionamiento; deriva en la necesidad de comparar los resultados generadas por las mismas de forma más exhaustiva, para no hacer suposiciones precipitadas ni descartar antes de su comprobación una decodificación mejor, lo cual ha sido abordado en 7. Experimentación computacional

3.1.5 Variables relativas a las reglas de asignación

Relativo a las reglas de asignación, que, en resumen, dado que se trata en más detalle en 3.2. Decodificaciones, se refiere al tipo de algoritmo usado para la planificación de las cirugías que constituyen la lista de espera en

los distintos conjuntos día – quirófano, en algunas de ellas, ha sido necesario la implementación de una serie de variables para su correcto funcionamiento, más en concreto en las relativas a la *Best Fit* y *Level Fit*, las cuales se basan en asignar la cirugía al día – quirófano, que encaje y que más y menos tiempo disponible tenga, respectivamente. De este modo, se implementan una serie de variables en las decodificaciones relativas a las mimas, las cuales son.

- JH: Se trata de una lista de tuplas formada por tantas tuplas como combinaciones de j y h haya, $[(1,1),(2,1),\dots,(1,2),(2,2),\dots,(j,h)]$. Su función es la de almacenar dichas combinaciones, representativas de los conjuntos día – quirófano, de forma ordenada en base al criterio relativo al tiempo disponible en los mismos, y de forma ascendente y descendente, para la regla $Best\ Fit$ y $Level\ Fit$, respectivamente. Su implementación radica en simplificar la extracción de los índices j y h de forma iterativa y conjunta para los conjuntos día – quirófano ordenados.
- $available\_time$: Se trata de una matriz de tamaño $|J| * |H|$, donde se almacena el tiempo disponible de cada conjunto día – quirófano, al momento de realizar una programación, en la posición relativa al mismo. Este tiempo disponible se calcula según (3–6).

$$\forall j \in J, \forall h \in H: available\_time_{jh} = r_{jh} - \max_{s \in S} c_{sjh} \tag{3–6}$$

- $available\_time\_tuple$: Se trata de una conversión, o más correctamente un almacenamiento en formato de listas de tuplas, del tiempo disponible de cada día – quirófano ($available\_time_{jh}$), junto al índice del quirófano (j) y del día (h) en cuestión, tal y como se muestra en (3–7). Su implementación radica en realizar la ordenación en base al tiempo disponible de la misma, de forma conjunta con los índices, para la inmediata extracción de estos una vez realizada la ordenación.

$$available\_time\_tuple\ = (available\_time_{jh}, j, h) \tag{3–7}$$

3.1.6 Variables relativas a las funciones objetivo y la solución

Las últimas variables principales del modelo, son las relativas al almacenamiento de la solución, así como el cálculo de las funciones objetivo, las cuales son presentadas a continuación.

En primer lugar, la programación de las cirugías se almacena en forma de vector de permutación o secuencia, tal y como se muestra en (3–8), donde el contenido se refiere a las etiquetas de las distintas cirugías que han sido programadas con éxito, ordenadas en función de dicha programación. Dicha variable denotada como π.

$$\pi\ =\ [1,2,3,6,4,8,10,9,7,5\dots] \tag{3–8}$$

El cálculo del valor objetivo principal obtenido OF_1, relativo al máximo de cirugías programadas, se muestra en la fórmula (3–9), donde se observa que consiste en calcular la suma de los pesos de las cirugías programadas, siendo generado por tanto una vez se haya finalizado el cálculo de la misma.

$$OF_1 = sum(w_i)\ \forall i \in \pi \tag{3–9}$$

El objetivo secundario, relativo al mínimo número de movimientos realizados por los cirujanos es en parte calculado durante las distintas iteraciones realizadas en la programación de las cirugías. Siendo ésta la relativa a dos variables que posibilitan su cálculo, denominadas $movements_{sh}$ y $last\_OR_{sh}$ y que se tratan de matrices de tamaño $|S| * |H|$, a continuación se describen las características de estas variables, acompañado de una breve explicación del funcionamiento de su cálculo, para una mejor comprensión de las mismas.

- Número de movimientos de cada cirujano s en cada día h ($movements_{sh}$): Almacena el número total de movimientos, es decir cambios de quirófano, realizados por un cirujano en un mismo día. Por tanto, se le suma 1 unidad a la posición en cuestión, cuando dicho cambio se produzca y en el nivel más bajo de la decodificación, es decir, momento en el que se ha logrado programar la cirugía que

está siendo planificada. Se inicializa a 0 antes de la decodificación, es decir, ningún movimiento contabilizado antes del comienzo de la misma.

- Último quirófano de cirujano en un día ($last\_OR_{sh}$): Permite identificar el cambio de quirófano o movimiento de cada cirujano en cada día, almacenando en la posición en cuestión la etiqueta del quirófano donde ha sido programada la cirugía. De esta forma, se puede comprobar en una posterior iteración si para ese mismo cirujano y día, coincide el quirófano actual o no. De igual forma a la anterior es inicializada a 0 en cada una de sus posiciones, ya que se está considerando que la primera entrada a un quirófano por parte de un cirujano en el día en cuestión, también se cuenta como movimiento. Este planteamiento es compatible además con las etiquetas del conjunto de quirófanos (J), ya que el mínimo valor que poseen es 1.

El valor del objetivo secundario, referido a la suma de todos los valores contabilizados en la matriz $movements_{sh}$, puede ser calculado una vez se haya finalizado el cálculo de la misma, según muestra la fórmula (3–10).

$$OF_2 = \sum_{s=1}^{S} \sum_{h=1}^{H} movements_{sh} \tag{3–10}$$

Todas estas variables, exceptuando la relativa a los intervalos libres de cirujanos, se inicializan a 0 o como listas vacías en el modelo, además el lugar para ello es dentro de la propia función relativa a la decodificación y antes de comenzar a ejecutarla.

Relativo a Python, a diferencia de los datos, dichas variables no se inicializan como objeto de tipo *self*, pues en un principio no se da ningún escenario, donde se requiera dejar almacenado los valores que tomen durante el cálculo de la solución, además no es conveniente para el recorrido del espacio de soluciones, ya que se corre el riesgo de que quedaran almacenadas en memoria, acumulando valores y aportando resultados erróneos.

Finalmente, comentar que el tipo de solución planteada para almacenar las soluciones generadas y formalizada para todas las etapas que contiene el modelo, como decodificaciones, operadores, así como el algoritmo en su conjunto, es la de una tupla que contenga la secuencia de cirugías antes definida, así como ambos valores objetivos, tal y como se muestra en (3–11).

$$\phi = (\pi, OF_1, OF_2) \tag{3–11}$$

3.2 Decodificaciones

El esquema de codificación usado para la resolución del problema de programación anticipada se basa en una nueva implementación del mismo, propuesto en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015)

Este esquema se basa en codificar el plan quirúrgico en un vector de permutación p, como ya fue definido en el capítulo anterior, así como en un operador BP.

- Vector de permutación (p): representa un cierto orden de las cirugías de la lista de espera, el cual es determinado teniendo en cuenta la priorización de cirugías en base a determinados factores.
- Operador *Bin Packing* (BP): representa el tipo de algoritmo utilizado para asignar las cirugías a los conjuntos día – quirófano.

Dicho capítulo se centra en introducir el tipo de operador BP, utilizado en cada una de las decodificaciones implementadas, para posteriormente, en el 4 Heurísticas, abordar el ordenamiento relativo al vector de permutación (p), a través de una serie de heurísticas y reglas de despacho.

Respecto a la decodificación de la solución y explicado de forma general, se refiere, a la generación a partir de los datos inicializados, de una solución factible para el problema. Por ello es una etapa crucial, ya que su correcto planteamiento e implementación nos garantiza crear planes quirúrgicos viables y acordes al problema real, y, por ende, llevar a cabo una correcta asignación de las intervenciones quirúrgicas.

Las principales secciones de las que se compone el código relativo a las decodificaciones de forma general son:

0. Inicialización de variables: Definidas en el capítulo anterior
1. Recorrido de los conjuntos (I, H, J en el caso del problema abordado)
2. Condicionales: Basados en las restricciones del modelo y adaptadas a resolución mediante métodos aproximados.
3. Actualización de variables: Necesarias para la correcta programación de forma iterativa, así como registro de la misma, en lo que se refiere a la solución y a su valor objetivo.

A parte de dicha factibilidad, a la hora de generar el código se atienden otros aspectos como la eficiencia o la calidad computacional para su posterior implementación.

En el caso de dicho estudio se han planteado 3 distintas decodificaciones, las cuales se caracterizan por seguir un algoritmo de asignación concreto y por la incorporación o no en cada una de ellas de la variable anteriormente introducida, en el apartado 3.1.4. Variables específicas de decodificaciones, sobre el aprovechamiento de intervalos de tiempo libres, para asignación de cirugías y sus cirujanos responsables. Habiendo, por tanto, un total de 6 decodificaciones. Todo ello con el objetivo de tener un mayor número de modelos durante la etapa de pruebas que comparar. Dichos algoritmos han sido:

- *First Fit* (FF): La cirugía es programada en el primer día-quirófano donde encaje.
- *Best Fit* (BF): La cirugía es programada en el día-quirófano que tenga la menor cantidad de tiempo disponible y que encaje.
- *Level Fit* (LF): La cirugía es programada en el día-quirófano que tenga la mayor cantidad de tiempo disponible y que encaje.

A continuación, se procede a explicar más en detalle cada una de ellas, así como ventajas e inconvenientes que presentan a nivel de implementación, pues la calidad de las soluciones que generan, así como su eficiencia computacional, será analizada más en detalle en el capítulo destinado a ello.

3.2.1 Decodificación basada en la regla First Fit

Implementa el algoritmo de asignación First Fit, el cual consiste en programar la cirugía planeada para el primer día-quirófano en el que encaje. Debido al tipo de recorrido de los conjuntos, es la más sencilla a nivel de código. Por otro lado, es la de menor tiempo de cómputo, debido a que no requiere bucles anidados para su correcto funcionamiento, bastando únicamente con las genéricas, relativas a los conjuntos, así como la actualización de sus variables. Los pasos que se efectúan en la misma son:

1. Inicialización de variables comunes para todas las descodificaciones y propias de la misma. X_{ijh}, Z_{sjh}, c_{sjh}, r_{sjh}, d_{sjh}, $movements$, $last\_OR$, π, para el caso general. Así como la adición de $intervals$ y la sustitución de los *Completion Time* relativos al elemento *Cirujano*, por los relativos al elemento *Cirugía*, para la nueva implementación.
2. Recorrido de los conjuntos: Recorrido de las listas I, J, H, de forma anidada entre sí, de tal forma que se genere una combinación del tipo cirugía – quirófano – día, para ser evaluada posteriormente si su programación es factible. Permite realizar la planificación de forma iterativa y recorrer todas las posibles combinaciones, así como finalizar cuando han sido evaluadas todas o programadas en el mejor de los casos. De forma adicional se debe extraer la etiqueta de la cirugía de forma correcta, debido al caso en el que puede haber de entrada una π desordenada; así, como disponer el recorrido de las cirugías en el nivel superior, por cuestiones de eficiencia computacional, al ser esta generalmente la de mayor tamaño.
3. Condicionales: Se encuentran anidados entre sí y con respecto a los recorridos; con el fin de evaluar la combinación de cirugía – quirófano – día generado en el paso anterior, admitiendo su programación en caso de cumplimiento de todos ellos; o por el contrario siendo descartada, en caso de no cumplir alguno.
4. Actualización de variables: Se encuentran anidados respecto a los condicionales, es decir, se efectúa si se cumplen todos y cada uno ellos, y son necesarias para la correcta programación de forma iterativa, así como los cálculos para la solución generada. Las distintas actualizaciones de las variables antes descritas, realizadas en este nivel más interno, pueden ser desglosadas en las siguiente.
    a. Adición de la cirugía i programada a la secuencia de cirugías programadas π

b. Actualización de las variables de asignación X_{ijh} y Z_{ijh}, para las posiciones determinadas por la combinación cirugía – quirófano – día programada.

c. Determinación del intervalo u admisible y más temprano de la variable $intervals$ para el cirujano s y día h en cuestión (únicamente en la versión del modelo que lo implementan)

d. Cálculo de Release Date para la combinación en cuestión según (3–3) y actualización de la variable relativo al mismo para las posiciones determinadas por dicha combinación.

e. Cálculo de Due Date para la combinación en cuestión, mediante la suma del tiempo de duración de la cirugía programada t_i, al Release Date calculado en el paso anterior. Así como la actualización de su variable de forma análoga.

f. Cálculo de Completion Time de forma similar al Due Date del paso anterior.

g. Actualización de la variable $intervals$ según (3–5) (únicamente en la versión del modelo que lo implementa)

h. Actualización de las variables $movements$ y $last\_OR$, según lo detallado en 3.1.6. Variables relativas a las funciones objetivo y la solución.

5. Cálculo de los valores objetivos, OF_1 y OF_2 según (3–9) y (3–10), así como construcción de la solución ϕ de la forma mostrada en (3–11), una vez finalizada la fase iterativa anterior.

3.2.2 Decodificación basada en la regla Best Fit

Implementa el algoritmo de asignación Best Fit, el cual consiste en programar la cirugía planeada en el día-quirófano que tenga la menor cantidad de tiempo disponible y que encaje. Requiere por tanto una modificación en el recorrido de los conjuntos, así como la implementación de una serie de variables y cálculos para lograr el correcto funcionamiento de la misma, mostrados en 3.1.5. Variables relativas a las reglas de asignación los cuales se muestran en una serie de pasos a continuación. De forma resumida se basa en el ordenamiento de los índices de los quirófanos y días de forma conjunta, en función de cada uno de sus tiempos disponibles y de forma ascendente, es decir, a menor tiempo disponible antes en la lista.

Frente a la anterior decodificación presenta un código más complejo, así como el requerimiento de una serie de variables, ordenamientos, y cálculos que garantizan el correcto funcionamiento antes explicado, lo cual deriva en un mayor tiempo de cómputo.

1. Inicialización de variables: De forma similar a la decodificación relativa a la regla *First Fit*, con la adición de JH, dado que las relativas a los timepos disponibles se inicializan dentro de la propia decodificación, debido a que la ordenación se realiza de forma iterativa, es decir, tras una programación. Esto último debido, al hecho de que se ven modificados tras la misma.
2. Recorrido del conjunto cirugías (I): Similar a *First Fit*
3. Recorrido de la lista de tuplas (JH) y extracción de los índices del quirófano (j) y del día (h)
4. Condicionales: Similar a First Fit
5. Actualización de variables: Similar a *First Fit* en lo que respecta al marco general, con la implementación de las actualizaciones relativas a las variables para dicha regla de asignación.
    a. Adición de la cirugía programada a la secuencia solución: Similar a *First Fit*
    b. Actualización de variables de asignación (X_{ij} y Z_{ijh}): Similar a *First Fit*
    c. Determinación del intervalo libre relativo a la nueva decodificación planteada: Similar a *First Fit*
    d. Cálculo del *Release Date*, *Due Date* y *Completion Time*: Similar a *First Fit*
    e. Actualización de la variable $intervals$ relativo a la nueva decodificación planteada: Similar a *First Fit*
    f. Actualización de las variables $movements$ y $last\_OR$ relativas al cálculo de los valores objetivos: Similar a $First\ Fit$

g. Actualización de las variables relativas a los tiempos disponibles para el ordenamiento de los índices de los conjuntos día – quirófano: Cálculo de $available\_time_{jh}$, como se muestra en (3–7) y actualización de la matriz en cuestión, para la posterior creación de la variable $available\_time\_tuple$, con el formato mostrado en (3–7). Una vez calculada se procede a su ordenación en base al primer elemento de las tuplas, relativo al tiempo disponible y de forma ascendente, en base a la regla de asignación.

h. Actualización de la variable JH, relativa a las listas de tuplas, formada por todas las posibles combinaciones de los índices de quirófanos (j) con los de días (h), mediante la extracción del segundo y tercer elemento de $available\_time\_tuple$, es decir, j y h, y adición en JH según la ordenación realizada.

6. Cálculo de valores objetivo y construcción de la solución: Similar a *First Fit*

3.2.3 Decodificación basada en la regla Level Fit

Análoga a la anterior y similar en todo lo que respecta al código, siendo la ordenación relativa a JH en función de los tiempos disponibles de los conjuntos día – quirófano y en este caso de forma descendente, es decir conjuntos día – quirófano con más tiempo disponible, primeros en la lista.

En Ilustración 3–6, se puede visualizar un diagrama de flujo con los pasos más generales que constituyen una decodificación, resumiendo lo recién explicado.

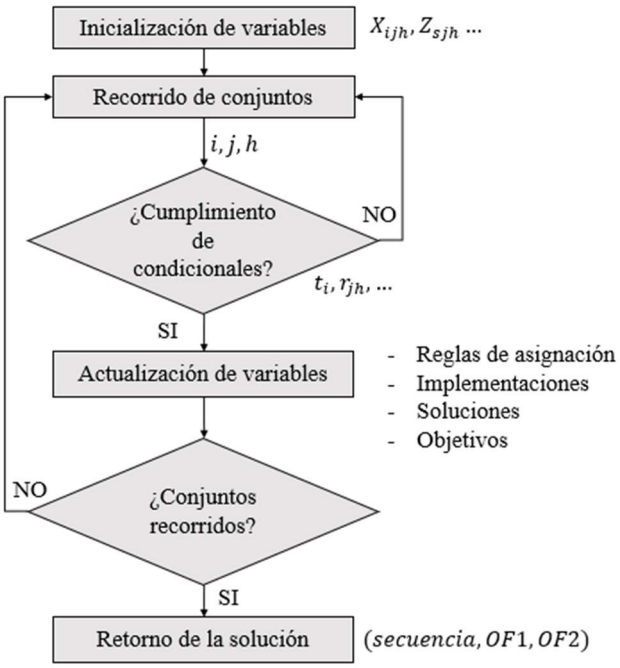


Ilustración 3–6. Diagrama de flujo representativo de una decodificación genérica

Para una mejor interpretación de las reglas de asignación, se presenta en **Ejemplo** 3–4. Comparación de **solución generada por decodificaciones basadas en** regla *First Fit, Best, Level Fit*, el programa resultante para la programación de la misma lista de espera mediante las distintas reglas de asignación presentadas.

Ejemplo 3–4. *Comparación de solución generada por decodificaciones basadas en regla First Fit, Best, Level Fit*

El entorno propuesto para dicho ejemplo se trata de una instancia de 10 cirugías, realizadas cada una de ellas por un cirujano distinto, con el fin de reflejar principalmente lo relativo a los algoritmos de asignación recién explicados, así como 5 quirófanos y un horizonte de planificación de 1 día, adicionalmente se establece la capacidad para todos los quirófanos y cirujanos en 390 minutos (8:30 – 14:00). Similar a ejemplos ya

mostrados, hay datos como las fechas más tempranas, admisibilidad entre cirugía-quirófano-día, etc. Los cuales se definen lo menos restrictivo posible, siendo los datos resultantes los siguientes.

$I = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$J = [1, 2, 3, 4, 5]$

$H = [1]$

$S = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$\gamma_i = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$t_i = [73, 107, 102, 96, 192, 126, 191, 133, 60, 146]$

$r_{jh} = 390 \quad \forall j \in J, \forall h \in H$

En Ilustración 3–7, se muestra la ordenación de la lista de pacientes a programar, donde cada cuadro representa cada cirugía, marcado tanto por la etiqueta de la misma en el centro, como por el cirujano responsable en su esquina superior derecha. Así como los diagramas, representativos de la programación generada, resultado de usar cada una de las decodificaciones planteadas basadas en los algoritmos de asignación, *First Fit*, *Best Fit* y *Level Fit*, respectivamente, implementados en dicho estudio.

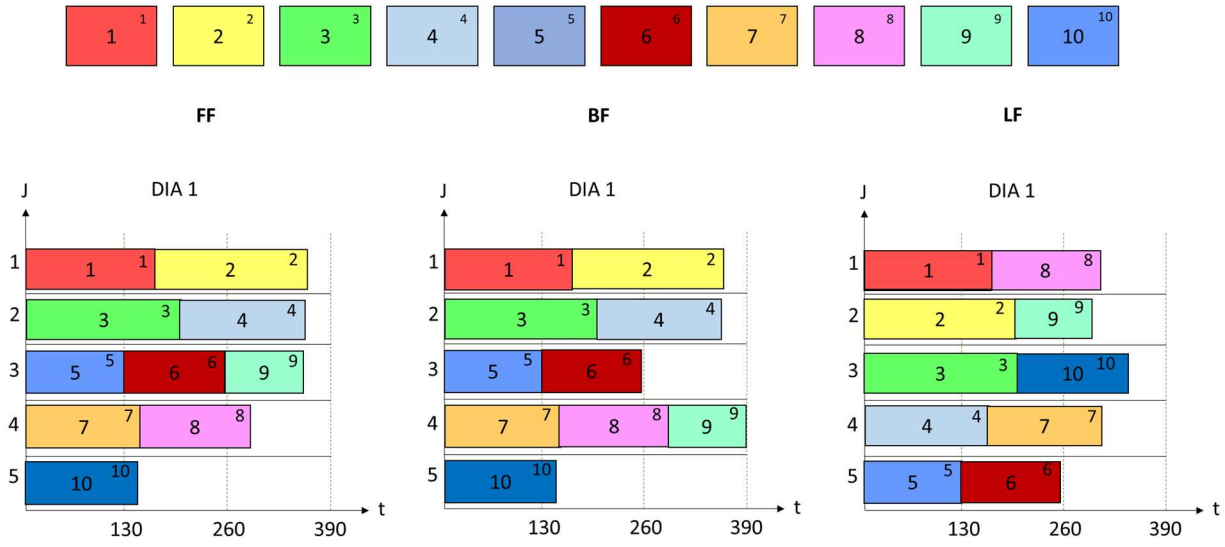


Ilustración 3–7. Comparaciones de programas en base a distintas reglas de asignación

3.3 Representación de la solución

Una vez explicadas las decodificaciones, usadas para generar una solución admisible para una lista de pacientes en forma de secuencia dada, se procede a explicar el tipo de representación genérica para las soluciones que generan.

Dicha solución, como ya ha sido introducido está formada por la secuencia de pacientes generada tras la programación, así como los valores para las dos funciones objetivo para la misma.

$$\phi = (\pi, OF_1, OF_2)$$

Sin embargo, al tratarse la secuencia de un vector de permutación cuya ordenación, es el resultante de la programación realizada en la decodificación, es recomendable implementar algún tipo de representación que refleje mejor las características del programa generado. Para dicho estudio han sido planteadas dos tipos de representaciones, las cuales se presentan a continuación.

3.3.1 Representación de la solución 1

La representación planteada se basa en la propuesta en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), la cual tiene las siguientes características

- Rectángulos: Representan cada una de las cirugías a programar, habiendo tantos como cirugías programadas en la solución. Está dispuesto de un color diferencial de la cirugía representada para poder diferenciarlas, así como de una serie de valores identificativos de la misma y de alguno de los valores que tiene asociados a los datos (cirujano responsable, duración, etc.). Por otro lado, la longitud de la base es proporcional a la duración de la misma.
- Ejes: Se trata de los ejes del diagrama, representativo de la solución. El eje vertical está formado por los días del horizonte de planificación, habiendo como resultado, tantos ejes horizontales como días. El eje horizontal, en cambio, representa el tiempo, es decir los distintos instantes de tiempo donde son iniciadas, realizadas y finalizadas las distintas cirugías en dicho día.
- Diagrama: El diagrama en su conjunto, formado por los elementos antes descritos, representa un quirófano, habiendo de esta forma tantos diagramas como quirófanos tenga el área.

Dicha programación permite una mejor visualización de la programación de las cirugías en el área para los distintos quirófanos que lo componen, otorgando al tomador de decisiones un elemento visual de las cirugías que serán realizadas en cada uno de ellos, para el horizonte de planificación definido. En Ilustración 3–8 se muestra una representación para un programa solución con dicha representación.

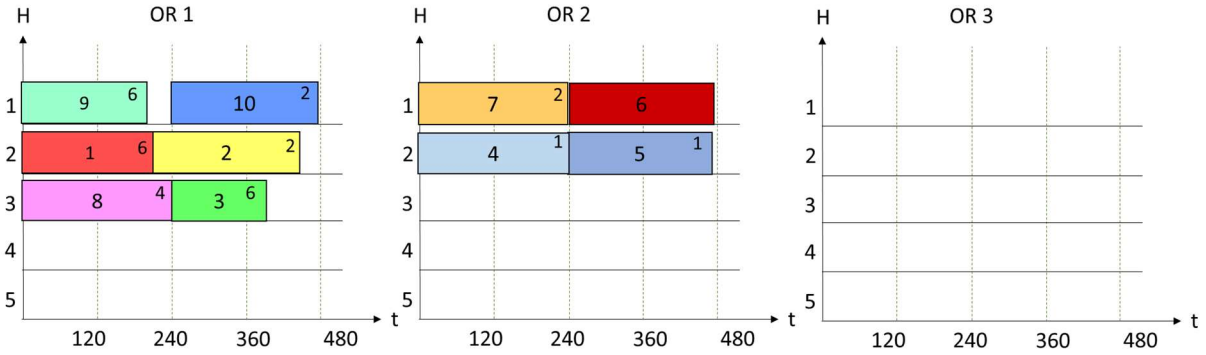


Ilustración 3–8. Representación con un enfoque de programación anticipada

3.3.2 Representación de la solución 2

La representación planteada en cuestión, la cual sigue la estructura de ejemplos ya mostrados con anterioridad, se trata de una serie de diagramas con las siguientes características.

- Rectángulos: Similar a la representación anteriormente explicada.
- Ejes: Se trata de los ejes del diagrama de Gantt, representativo de la solución. El eje vertical está formado por los quirófanos de la instancia generada, habiendo como resultado, tantos ejes horizontales como quirófanos en el área. El eje horizontal, en cambio, representa el tiempo, es decir los distintos instantes de tiempo donde son iniciadas, realizadas y finalizadas las distintas cirugías.
- Diagrama: El diagrama en su conjunto, formado por los elementos antes descritos, representa un día, habiendo de esta forma tantos diagramas como días tenga el horizonte de planificación definido

Dicha implementación, por tanto, otorga al tomador de decisiones y en definitiva al área donde fuera implementado, un elemento más visual de la programación obtenida para cada uno de los días del horizonte de planificación definido, permitiendo visualizar en cada diagrama las cirugías a realizar en el día cuestión, así como los instantes de realización de las mismas, y en definitiva la jornada resultante para el grupo de cirujanos y quirófanos del área. En Ilustración 3–9, se muestra un ejemplo similar al recién mostrado en base a esta nueva representación

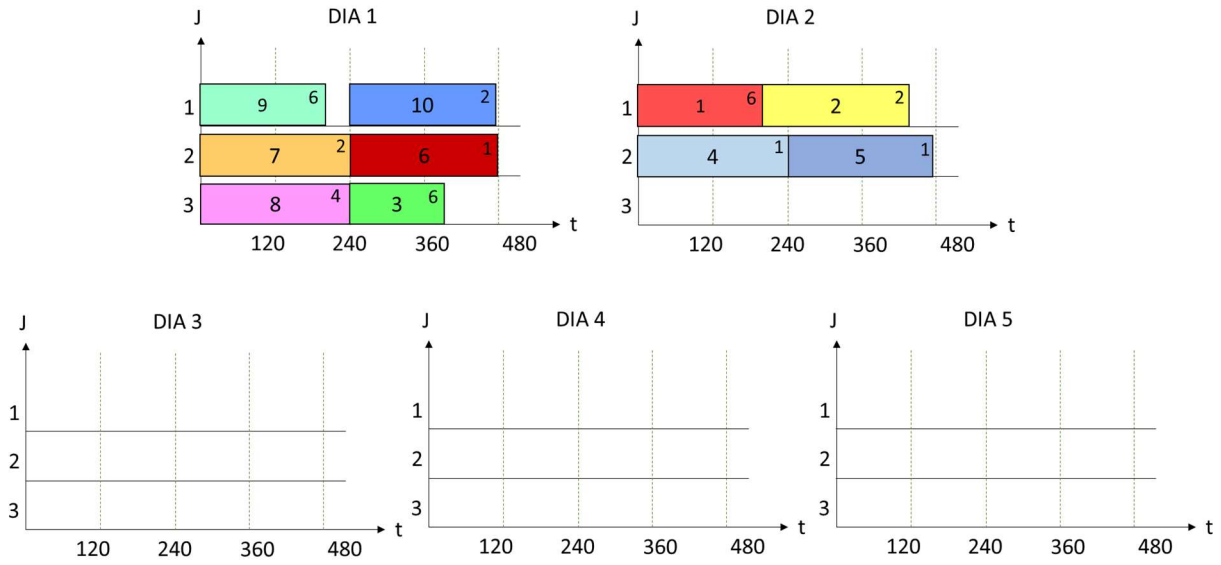


Ilustración 3–9. Representación con un enfoque de programación de asignación

Analizando ambas representaciones y enfocando su uso a un escenario real, se puede sacar la conclusión de que ambas son válidas y complementarias de cara a su implementación, aunque puede ser recomendable una más que otra en base al nivel operativo abordado o el método de generación de las mismas.

1ª opción: Es más aplicable a una toma de decisiones, en la planificación de cirugías constituyentes de listas de espera mayores y para un número reducido de quirófanos, por ejemplo, una determinada área del hospital con recursos más limitados, propios del paso de Programación Anticipada

2ª opción: Está más enfocada a una programación de un determinado día, o según los dos pasos de programación de cirugías en el ámbito sanitario, para la toma de decisiones en un nivel operativo, la secuenciación en el quirófano de cirugías más próximas o urgentes de ser realizadas, propios del paso sobre Programación de Asignación.

En definitiva, la elección de su uso, depende en gran medida de la manejabilidad de los mismos en base al tamaño de las instancias generadas, así como el resultado esperado. Reforzando el motivo de su implementación, el cual se trata de facilitar la interpretación de la programación realizada, al tomador de decisiones, y en definitiva a cualquier parte interesada.

4 HEURÍSTICAS

Una vez explicadas las decodificaciones usadas para la generación de soluciones admisibles, se puede llegar a la conclusión, a pesar del papel fundamental que presentan, que su uso aislado genera limitaciones en la resolución realizada, y, por tanto, en la calidad de la programación generada. Es aquí donde aparece un concepto clave en la resolución aproximada de problemas de optimización, el cual se trata de las denominadas Heurísticas.

En los problemas de optimización la heurística se refiere a aquellos procedimientos que se aplican en la resolución aproximada del mismo, caracterizados por estar diseñados e implementados mediante la intuición de quien lo desarrolla y aplica, gracias a un alto grado de comprensión de la estructura del mismo, sumado a experiencia previa en problemas similares.

Por ello, en dicho capítulo se introduce las heurísticas implementadas en los algoritmos, usados para la resolución del problema, el funcionamiento de las mismas y conceptos básicos en los que se basan.

4.1 Reglas de despacho

En primer lugar, se presenta las Reglas de Despacho que, sin considerarse una heurística como tal, al ser más básicas y no estar dotadas de tal grado de inteligencia; como los algoritmos considerados heurísticas, están muy relacionadas con las mismas, llegando a ser en la mayoría de casos, las bases de estas.

El fundamento de la regla de despacho se basa en la ordenación de un determinado conjunto en base a un determinado factor. Más en concreto y de la forma definida en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), su función se trata de una operación de ordenación del tipo (S_i, S_c), donde S_i es el indicador de clasificación, parámetro utilizado como referencia para su ordenación (duración de las cirugías, fecha de vencimiento, etc.), mientras que S_c, es el criterio de clasificación, el cual indica como se ordenan las cirugías según el anterior (ascendente, descendente, etc.). A continuación, se procede a introducir las combinaciones de los mismos que han sido implementadas en dicho estudio, así como el motivo de las mismas.

- Indicador de clasificación (S_i): Parámetro de ordenación, el cuál se trata de uno de los datos ya introducidos, usado como referencia para la ordenación de la lista de pacientes inicial. Los seleccionados han sido.
    - Tiempo de duración (t_i).
    - Fecha más tardía (d_i).
    - Cirujano responsable (γ_i).
    - Peso (w_i)
    - Ordenación aleatoria (ran).
- Criterio de ordenación (S_c): Parámetro de ordenación que acompaña al anterior y determina el tipo de ordenación. En dicho estudio se han considerado algunas de las propuestas en la literatura.
    - Ascendente (INC): ordena las cirugías según los valores crecientes del indicador S_i.
    - Descendente (DEC): ordena las cirugías según los valores decrecientes del indicador S_i.
    - Colina (HILL): Ordena las cirugías de tal forma que los valores altos del indicador S_i queden en el medio de la lista de espera y cifras bajas al principio y al final.
    - Valle (VALLEY): Ordena las cirugías de tal forma que los valores bajos del indicador S_i queden en el medio de la lista de espera y valores altos al principio al final.

En **Ejemplo 4–1.** *Ordenación de secuencias mediante la aplicación de reglas de despacho*, se puede visualizar más en detalle la lista de pacientes resultantes para un total de 4 reglas de despacho de la forma (S_i, S_c) aplicadas, para una mejor interpretación de las mismas.

Ejemplo 4–1. *Ordenación de secuencias mediante la aplicación de reglas de despacho*

Los datos a considerar, más en concreto aquellos seleccionados como indicador de clasificación S_i, son:

- $t_i = [102, 200, 98, 120, 157, 130, 128, 145, 198, 180]$
- $d_i = [5, 4, 6, 2, 2, 4, 5, 7, 3, 4]$
- $rd_i = [3, 1, 5, 1, 1, 2, 2, 5, 1, 3]$
- $\gamma_i = [1, 3, 2, 5, 2, 1, 5, 4, 2, 6]$

En Ilustración 4–1, se muestran las secuencias resultantes de aplicar los operadores (S_i, S_c), indicados a su izquierda, para la lista de espera mostrada en primer lugar. Cada rectángulo representativo de la secuencia lleva incorporado los valores de los datos empleados como indicador de clasificación.

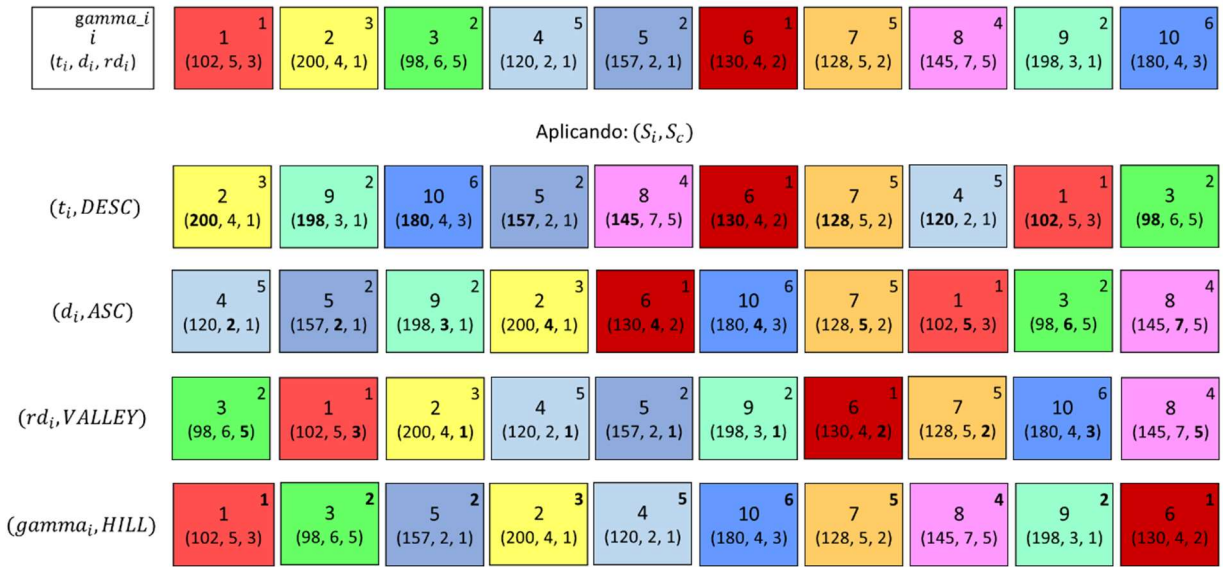


Ilustración 4–1. Secuencias resultantes de la aplicación de diversas reglas de despacho

Como se observa, las reglas de despacho se basan en general en la ordenación de la lista de pacientes en base a un dato propio de los mismos. Sin embargo, también es frecuente usar como indicador de clasificación, Si, combinaciones de los mismos. A continuación, en Tabla 4–1, se detalla una lista con las reglas de despacho incorporadas en los distintos modelos o versiones de los mismos. Así como otros datos de interés como el tipo de operador de ordenación (S_i, S_c) en el que se basan, y la denotación usada para referirse a ellos.

Tabla 4–1. Resumen de reglas de despacho incorporadas

Nombre	ID	Operación de ordenación	Indicador de clasificación	Criterio de clasificación
Earliest Due Date	EDD	(d_i, ASC)	Fecha de finalización de la cirugía (d_i)	Ascendente (ASC)
Largest Processing Time	LPT	(t_i, DESC)	Duración de la cirugía (t_i)	Descendente (ASC)
Shortest Processing Time	SPT	(t_i, ASC)	Duración de la cirugía (t_i)	Ascendente (ASC)
Valley Processing Time	VPT	(t_i, VALLEY)	Duración de la cirugía (t_i)	Valle (VALLEY)
Hill Processing Time	HPT	(t_i, HILL)	Duración de la cirugía (t_i)	Colina (HILL)
Biggest Weights Before	BWB	(w_i, ASC)	Peso de la cirugía (w_i)	Descendente (DESC)
Weighted Shortest Processing Time	WSPT	(w_i/t_i, ASC)	Peso de la cirugía (w_i) / tiempo de duración (t_i)	Ascendente (ASC)
Ordained Surgeons	OS	(γ_i, ASC)	Cirujano responsable de la cirugía (γ_i)	Ascendente (ASC)

Una vez presentados, se procede a explicar el motivo de su elección, así el lugar y la función para el algoritmo al completo, que estos presentan.

Relativo a la EDD y LPT, su uso se debe a la incorporación de los mismos en la heurística constructiva ELED, para la generación de soluciones iniciales y la formación del conjunto de población inicial en los algoritmos meméticos, explicado en más detalle en 5 Metaheurísticas, y según lo propuesto en (Huang, Pan, & Gao, 2023).

Las restantes, han sido incorporadas en una heurística, como método de comparación con el anterior, y dispuesto en el operador relativo a la generación de soluciones iniciales. Más en concreto, se basa en la heurística constructiva MT_{ALL} empleada en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), pero que a diferencia de éste el cual incorpora un conjunto de $(n^{S_i} * n^{S_i})$ secuencias; donde n es el número de indicadores y criterios respectivamente, en dicho caso han sido seleccionadas aquellas que se estima que pueden afectar en mayor medida a los valores de las funciones objetivo planteadas. El motivo de su elección se muestra a continuación.

- LPT, SPT, VPT y HPT: Incorporadas, debido a que; aunque en primera instancia se puede interpretar que cuanto más prioridad se le dé a cirugías a de menor duración, más cirugías serán programadas, el hecho es que factores como las limitaciones de los cirujanos, la no superposición de los mismos o la capacidad limitada de los recursos, pueden provocar que se generen programaciones no deseadas si se decide de entrada una regla de despacho concreta sin pruebas computacionales que lo refuten.
- BWB y WPST: Incorporada debido al efecto directo sobre la ponderación de la función objetivo principal del problema. WSPT, además de dar prioridad a los pesos, puede llegar a corregir limitaciones de la anterior, como podría ser el caso de ordenar las de mayor peso, pero que tuvieran tiempos de duración excesivos.
- OS: Incorporada atendiendo al objetivo secundario del problema sobre los movimientos de cirujanos.

4.2 Heurísticas constructivas

Como se ha observado, acerca de las reglas de despacho, se basan en un fundamento muy simple y su aplicación directa o única puede mejorar la solución en gran medida, sin embargo, pueden presentar limitaciones. Es por ello, que los algoritmos implementados, así como la resolución aproximada en general, necesita de la incorporación de un mayor grado de inteligencia para aumentar su eficacia.

Por ello es recomendable el uso de alguna heurística constructiva, como operador de partida, y que como su

propio nombre indica, tenga la función de generar soluciones de entrada al algoritmo de mejor calidad.

A continuación, se hace un resumen de las distintas heurísticas constructivas desarrolladas a lo largo del trabajo, así como una breve explicación de las mismas, dado que serán explicadas en más detalles en los capítulos relativos a su implementación.

4.2.1 Heurística constructiva basada en el uso de conjuntos

Se trata de la heurística ELED anteriormente mencionada, la cual se basa en el uso de una serie de conjuntos donde almacenar distintas partes de una secuencia dada, los cuales son ordenados bajo algún criterio de ordenación anteriormente explicado, más en concreto la regla EDD y la regla LPT, para posteriormente ir comparando uno a uno los trabajos de ambos conjuntos y formando la solución de partida.

Una vez generado se almacena y se vuelve a repetir el proceso hasta generar el número de soluciones deseado.

Dicha regla se basa en la implementada en (Huang, Pan, & Gao, 2023) y se trata en más en detalle en 5 Metaheurísticas.

4.2.2 Heurística constructiva basada en el uso de secuencias parciales

Se trata de una versión de la regla ST o método de *Single Tuple* y su versión ampliada de la MT o método de *Multiples Tuple* adaptado al problema en cuestión. Dichas heurísticas trabajan con una tupla de ordenación o con un conjunto de tuplas de ordenación, para generar una secuencia o $n^{Si} * n^{Sc}$ secuencias, respectivamente. La tupla de ordenación se trata del anterior operador de ordenación (Si, Sc) presentado en el capítulo relativo a las reglas de despacho.

En el caso de dicho trabajo se ha implementado dicha heurística MT de forma reducida, y denotada por tanto como MT_{brief}. Esto debido a que las tuplas de ordenación han sido seleccionadas de antemano bajo algún tipo de razonamiento, como se ha mostrado en Tabla 4–1.

De igual forma se basa en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015). De modo que, su funcionamiento consiste en generar en primera instancia una secuencia parcial con los pacientes cuya fecha de vencimiento (d_i) cae dentro del horizonte de planificación analizado, con el objetivo de no superar fechas límites; para posteriormente ordenar los trabajos restantes en base a los distintos operadores de ordenación (S_i, S_c) y añadirlo posterior a la secuencia parcial, formando así la secuencia completa.

4.2.3 Heurísticas internas de la decodificación

No son heurísticas constructivas propiamente dichas, pero se hace mención a ellas en dicho subcapítulo por sus características y funcionamientos similares. Tratándose de los distintos algoritmos usados en alguno de las decodificaciones ya mencionadas, necesarias para el cumplimiento de la función a la que se refieren, ya introducidos en 3 Decodificación de la solución . A modo de resumen, se tratan de:

- Decodificación basada en la regla de asignación Level Fit y Next Fit: El cual consiste en el almacenamiento de los tiempos disponibles de cada conjunto día – quirófano, en paralelo a la programación, es decir, al momento de asignar cirugías. Dichos tiempos disponibles se tratan de la diferencia entre la capacidad del mismo y las duraciones de sus cirugías programadas en iteraciones anteriores, es decir, sus Completion Time. Posteriormente se realiza una ordenación de la lista de tamaño JH, formada por las tuplas (j, h); es decir, todas las combinaciones posibles de trabajos y días, en base a dichos tiempos disponibles y de forma descendente y ascendente, en base a la regla seleccionada permitiendo la extracción de los índices deseados en la programación de la cirugía siguiente. Para la optimización del mismo se hace una trasposición de los tiempos disponibles con el objetivo de que priorice los quirófanos a los días, a la hora de ordenar la lista y se emplea una variable, del tipo de lista de tuplas, para simplificar pasos entre la ordenación y la extracción de los índices,
- Decodificación basada en la implementación de la variable relativa a los intervalos libres: La cual

consiste, como ya ha sido detallado, en el aprovechamiento de los intervalos de tiempo que quedan libres para un determinado cirujano. Para ello es necesario además de la variable anteriormente comentada, el uso de una lista que almacene las posiciones de los intervalos admisibles, para la cirugía que esté siendo planeada. Dicho intervalo se determina como admisible, si el tiempo de la cirugía es inferior al mismo. Por otro lado, se debe comprobar el máximo entre el intervalo y el *Completion Time* del conjunto día – quirófano donde esté siendo planeada. Para el funcionamiento correcto del mismo es necesario la ordenación de las listas de intervalos de forma ascendente en función de su tiempo,

5 METAHEURÍSTICAS

Definidas las heurísticas, se procede a explicar el concepto de metaheurística, la cual sigue los mismos fundamentos que el anterior, tratándose este de un algoritmo heurístico, el cual incorpora procedimientos y operaciones internas genéricas de la literatura y dicha rama de la optimización, esperando de él, una generación de soluciones de calidad eficaz, así como eficiencia en términos computacionales. "La metaheurística se puede concebir como estrategias generales para diseñar procedimientos heurísticos con alto rendimiento" (Melian, Moreno-Pérez, & Moreno-Vega, 2003)

Más en concreto se ha centrado el estudio en los algoritmos meméticos, los cuales destacan como una de las áreas de investigación de computación evolutiva más recientes y en crecimiento, tratándose de una combinación de búsqueda global basada en población y búsqueda heurística local basada en individuos (Chen, Ong, Lim, & Tan, 2011)

De este modo, los algoritmos meméticos incorporan conceptos de otras metaheurísticas, principalmente de los algoritmos evolutivos o genéticos, referidos a la búsqueda global basada en poblaciones, mediante operaciones sobre los distintos individuos que la forman, acompañado de procesos heurísticos de búsqueda local, los cuales incrementan el desempeño de los mismos.

Con el fin de comprender el algoritmo que ha sido tratado, cabe mencionar que los algoritmos genéticos, de los cuales en gran medida se basa, se inspiran en la evolución bióloga, siendo el funcionamiento de éste representativo de la misma. De este modo en dichos algoritmos se hace evolucionar una población de individuos, en este caso los vectores de permutación que definen las distintas secuencias, mediante operadores que incorporan aleatoriedad, como son mutaciones, cruce y selecciones, los cuales modifican los genes, es decir, los elementos de la secuencia, resultando en el mejor de los casos en la generación de individuos con mejores características que los anteriores.

El algoritmo memético en concreto sustituye el concepto de gen por meme, siendo éste su análogo en el contexto de la evolución cultural. Dichos memes son unidades de transmisión cultural: ideas, comportamientos, estilos o prácticas que se propagan de persona a persona y evolucionan con el tiempo. Así como los genes propagan rasgos biológicos, los memes propagan rasgos culturales. De este modo dicha evolución basada en memes opera mucho más rápido que la evolución genética, llevando al desarrollo y difusión rápidos de innovaciones y cambios culturales (Dawkins, 2006).

A continuación, se detalla una breve definición del contenido de los distintos subcapítulos que serán presentados.

El primer subcapítulo se centra en el algoritmo memético base, BMA, basado en el algoritmo memético propuesto en (Huang, Pan, & Gao, 2023), denominado de tal forma, ya que será el implementado en primera instancia en los distintos modelos propuestos. Cuya principal diferencia es la decodificación utilizada en los mismos, siendo común el resto del algoritmo, relativo al marco general de la metaheurística y los operadores que la constituyen.

El segundo subcapítulo introduce las versiones del mejor modelo del algoritmo memético base, versiones empleadas para su posterior etapa de calibración. Dichas versiones se caracterizan por el empleo de nuevos operadores genéticos o por modificaciones en la estructura general del BMA.

5.1 Algoritmo memético base

En primer lugar, cabe destacar que los distintos modelos implementados, así como sus posteriores versiones al ser calibrados, tienen de base el algoritmo memético propuesto por (Huang, Pan, & Gao, 2023), motivo por el que presentan ligeras modificaciones para poder ser adaptado al problema tratado sobre programación anticipada de listas de pacientes en conjuntos día – quirófano, así como el entorno del mismo. Más en concreto el anterior se centra en el problema de programación de flujo de permutación (*Flowshop*) de ensamblaje distribuido (DAPFSP), (Hatami, Ruiz, & Andrés-Romano, 2015), centrado en la optimización de

fabricaciones multiplanta formadas por líneas de ensamblaje, y la cadena de suministro múltiple que llevan asociada.

De este modo la estructura del algoritmo es similar, adaptado al problema y objetivos tratados, con el fin de mantener la estructura y funcionamiento del algoritmo memético eficaz (EMA) propuesto, y poder ser aplicado al problema de optimización quirúrgica que ha sido descrito.

A continuación, se procede a explicar su marco general, así como los distintos operadores que lo forman.

5.1.1 Marco general del BMA

Como ya ha sido comentado, el algoritmo memético implementado en primera instancia en los modelos base para la posterior etapa de pruebas, sirve para evaluar de entrada que decodificación genera mejores soluciones, con el fin de seleccionarlo y efectuar una etapa de calibración posteriormente. Por tanto, debido a esta estructura y bases del estudio realizado, será denotado, como BMA, *Base Memetic Algorithm*.

Dicho algoritmo se caracteriza por la búsqueda basada en población a través de una serie de operadores genéticos, y la incorporación de un método de búsqueda local, para mejorar el desempeño de la misma, de forma que la estructura del mismo incorpora una heurística constructiva para la generación de la población inicial, una serie de operadores genéticos sobre cruce, mutación, selección y actualización de la población, y dos tipos de búsquedas locales. A continuación, se presenta más detalladamente las características principales del algoritmo.

- Respecto a la generación de la población inicial, se implementa la heurística constructiva basada en la regla ELED, antes presentada, la cual combina las reglas sobre la fecha de vencimiento más temprana (EDD) y el tiempo de procesamiento más largo (LPT). De forma adicional se agrega un operador aleatorio para hacer que la población inicializada sea más diversa y reduzca las posibilidades de que los individuos caigan en óptimos locales.
- Por otro lado, presenta una iteración anidada dentro de una iteración superior con el fin de generar un conjunto de descendencia, resultante de aplicar los distintos operadores propios de la búsqueda global basada en población, junto a la búsqueda local basada en individuos, y ser usado como método de actualización de la población para mejorar la eficiencia del BMA en su totalidad.
- Respecto a los operadores de búsqueda local, han sido implementados dos, siendo uno la versión reducida del otro, los cuales se usan en distintas etapas del algoritmo y en base a una serie de condiciones, que atienden a los resultados de la programación que está siendo realizada.
- Respecto a los operadores de cruce y mutación, ambos incorporan parámetros de entrada los cuales son objeto de calibración, para adaptar el algoritmo al problema tratado.
- Finalmente, y relativo a la selección de individuos de la población, para la aplicación de los anteriores operadores, se realiza de forma aleatoria con el fin de aportar mayor diversidad a la búsqueda sobre el espacio de soluciones y evitar caer en óptimos locales, siguiendo el mismo fundamento que durante la generación de la misma.

Para una mejor comprensión de dichos operadores, así como su disposición en el algoritmo principal, se presenta a continuación la explicación de cada uno de ellos, así como una imagen ilustrativa de su pseudocódigo, generado a través de la herramienta LaTeX. Adicionalmente se hace mención a una serie de operaciones frecuentes en algunos de uno de ellos.

En primer lugar, y antes de proceder a su explicación, en la Tabla 5–1, se muestra un índice de los elementos y los parámetros que los componen, normalizados para todos los pseudocódigos generados, así como su notación empleada y formalizada en el resto del trabajo.

Tabla 5–1. Resumen de datos y variables en la codificación de los algoritmos aproximados

Conjuntos, datos y variables en la codificación de los algoritmos aproximados	
Índices y conjuntos	
$h \in H$	Conjunto de días dentro del horizonte de planificación (índice y tamaño)
$i \in I$	Conjunto de pacientes (cirugías) en la lista de espera (índice y tamaño)
$j \in J$	Conjunto de salas de operaciones del área (índice y tamaño)
$s \in S$	Conjunto de cirujanos del área (índice y tamaño)
Parámetros	
ω	Número de soluciones iniciales
ϑ	Población actual de individuos
φ	Conjunto de descendencia
ϕ	Una solución (individuo)
ϑ^b	Mejor solución de la población actual de individuos
ϑ^w	Peor solución de la población actual de individuos
Pa	Individuo que actúa como padre en la operación de cruce
Os	Individuo descendiente de la operación de cruce
Os'	Individuo resultante de la operación de mutación
$\pi^{\{regla\}}$	Secuencia resultado de aplicar la regla de despecho o heurística {regla} (ELED, ran, etc.)
$\pi^{\{sol\}}$	Secuencia correspondiente a una solución {sol} específica (Os, Pa, etc.)
π_i	Elemento de una secuencia en la posición i
δ	Un elemento de una secuencia (paciente)
$\delta^{\{sec\}}$	Elemento de una secuencia {sec} específica (π, $\pi^{\{sol\}}$, etc.)
ϑ^{b^*}	Posible nueva mejor solución de la población actual de individuos
ϕ^*	Posible nueva mejor solución de una etapa iterativa
Γ	Conjunto de varios elementos concretos (pacientes de la lista de espera)
$\Delta^{\{crit\}}$	Conjunto de varios elementos concretos ordenados en base al criterio {crit}
d	Due Date
p	Processing Time
$d_{\{sec\}_i}$	Due Date del elemento en la posición i en la secuencia {sec}
η	Número de la iteración en una etapa iterativa superior
n	Número de posibles soluciones distintas para operaciones basadas en vecindades
$flag$	Bandera de conteo
$improve$	Bandera de mejora

En Ilustración 5–1, se muestra el `pseudocódigo relativo al Algoritmo 1, relativo al marco general del algoritmo memético base.

Algorithm 1 BMA

Input: instance data, termination criterion, ω
Output: ϑ^b

1: $\vartheta :=$ a population with ω solutions generated by Initialization
2: perform Local-search-brief on the sequence of best solution ϑ^b in ϑ
3: **repeat**
4: $\varphi := \{\}$
5: **for** $\eta := 1$ **to** $\omega/2$ **do**
6: select Pa_1 and Pa_2 from ϑ randomly, Pa_1 has a higher value of the main objective function, lower value of the secondary objective function in case of a tie
7: $(Os_1, Os_2) :=$ Crossover(Pa_1, Pa_2)
8: $(Os_1{}', Os_2{}') :=$ Mutation(Os_1, Os_2)
9: perform Local-search-brief on the sequence of the better one between $Os_1{}'$ and $Os_2{}'$
10: $\vartheta^{b^*} :=$ compare $Os_1{}'$ and $Os_2{}'$ with ϑ^b
11: $\varphi := \varphi \cup \{Os_1{}', Os_2{}'\}$ ▷ append $Os_1{}'$ and $Os_2{}'$
12: **end for**
13: $(\vartheta, improve) :=$ Update-population$(\varphi, \vartheta, \vartheta^{b^*})$
14: **if** $improve =$ true **then**
15: perform Local-search-brief on the sequence of ϑ^b
16: **else**
17: perform Local-search on the sequence of ϑ^b
18: **end if**
19: **until** termination criterion met
20: **return** ϑ^b

Ilustración 5–1. Pseudocódigo del marco general del algoritmo memético

El algoritmo genera una población con ω soluciones utilizando un procedimiento de inicialización (*Initialization*) presentado. ϑ^b se establece como el mejor individuo en la población ϑ, y se realiza una búsqueda local reducida *(Local-search-brief)* en él. Posteriormente el BMA entra en la etapa iterativa superior, donde en cada iteración se deja inicialmente el conjunto de descendiente φ vacío y se ingresa a una iteración anidada interna.

En cada iteración interna, se selecciona dos individuos diferentes al azar de la población actual como padres. El de mayor valor de la función objetivo principal, o menor valor de la función secundaria en caso de empate, se establece como Pa_1 y el otro como Pa_2. Después de que los padres ingresan a la operación de cruce (*Crossover)*, se producen dos hijos Os_1 y Os_2. La operación de mutación (*Mutation)* en los dos hijos es para aumentar la variabilidad individual de ambos y generar $Os_1{}'$ y $Os_2{}'$. Posteriormente se aplica una búsqueda local reducida *(Local-Search)* al hijo con mayor valor de la función objetivo principal, o menor valor de la función objetivo secundaria en caso de empate. Si el mejor individuo entre $Os_1{}'$ y $Os_2{}'$ es mejor que ϑ^b, se reemplaza a ϑ^by se convierte en ϑ^{b^*}. Posteriormente se agregan al conjunto de descendientes de φ y una vez completadas las iteraciones internas, se actualiza la población actual (*Update-Population*), en base a los resultados obtenidos.

Finalmente, si la bandera de mejora (*improve*) se actualiza a verdadera (*true*), en la etapa de actualización de la población, se realiza una Búsqueda local reducida (*Local-Search-Brief*) sobre la mejor solución encontrada hasta el momento. De lo contrario se realiza una Búsqueda Local *(Local-Search)*.

Dichas iteraciones volverán a ser realizadas hasta que se alcance el criterio de terminación preestablecido. El cual se trata de un número de repeticiones o iteraciones dado como argumento de entrada, como establece el artículo en cuestión. De forma adicional, en dicho estudio, también ha sido implementado en el algoritmo completo, la condición de parada en base al tiempo de ejecución, debido a su gran relevancia en las etapas relativas a las pruebas computacionales y a la calibración del algoritmo, para ser realizadas de una manera justa.

Resaltar que la principal diferencia con respecto al algoritmo propuesto (Huang, Pan, & Gao, 2023), relativa al marco general, se encuentra en la disposición de las búsqueda locales (*Local-Search*), debido a que de forma original incorpora 4 distintas, dado que el problema que resuelve, trata sobre la secuenciación de productos a procesar en la línea y la secuenciación de los trabajos de cada uno de ellos a realizar, principal diferencia con respecto al de optimización quirúrgica, el cual solo trabaja con la secuencia de cirugías a programar. De este modo y en primera instancia, se usa durante el algoritmo la búsqueda local reducida (*Local-Search-Brief*), con el fin de no invertir mucho tiempo de ejecución en etapas tempranas, y poder realizar otras etapas de la búsqueda global, siendo usada la búsqueda local (*Local-Search*), únicamente en el caso que en la iteración interna no haya generado mejora.

5.1.2 Operador de inicialización del BMA

El procedimiento para la generación de la población inicial de soluciones tiene como fundamento la inicialización de una solución de mejor calidad, con el fin de reducir el tiempo de cómputo de la misma, así como la eficiencia computacional del algoritmo en su totalidad. Se basa en el uso de la heurística constructiva ELED, combinado con la generación de secuencias aleatorias con el fin de aportar una mayor diversidad al conjunto.

En Ilustración 5–2 se muestra el Algoritmo 2 sobre el procedimiento de Inicialización (*Initialization*).

Algorithm 2 Initialization

Input: instance data, π^{ELED}
Output: ϑ

1: $\vartheta := \{\}$
2: **for** $\eta := 1$ **to** $\omega/2$ **do**
3: $\quad \pi^{\mathrm{ELED}} :=$ sort patients according to the ELED rule
4: $\quad \Phi :=$ apply Decoding to π^{ELED}
5: $\quad \vartheta := \vartheta \cup \{\Phi\}$ ▷ append Φ to ϑ
6: **end for**
7: **for** $\eta := \omega/2 + 1$ **to** ω **do**
8: $\quad \pi^{\mathrm{rand}} :=$ sort patients randomly
9: $\quad \Phi :=$ apply Decoding to π^{rand}
10: $\quad \vartheta := \vartheta \cup \{\Phi\}$ ▷ append Φ to ϑ
11: **end for**
12: **return** ϑ

Ilustración 5–2. Pseudocódigo del procedimiento de inicialización

Como se muestra en el Algoritmo 2, se inicializa el conjunto vacío como indica la línea 1, para posteriormente almacenar $\omega/2$ soluciones que siguen la regla ELED como indica las líneas 2 – 6, y $\omega - \omega/2$ soluciones basadas en la ordenación aleatoria como indica las líneas 7 – 8.

La principal diferencia con respecto al original, es que éste usa el algoritmo PNEH para la ordenación de las secuencias de trabajos en base a un gradiente y la de los productos en base a la regla ELED, eliminado el primero de ellos debido a su ausencia, en dicho trabajo.

En Ilustración 5–3 se muestra el Algoritmo 3, sobre el procedimiento de la regla ELED.

Como ya ha sido comentado se trata de una heurística constructiva, la cual usa de forma combinada la regla EDD y LPT para la ordenación de secuencias, mediante el uso de una serie de conjuntos.

Algorithm 3 ELED

Input: instance data
Output: π^{ELED}

$\Gamma^d := \{\}, \Gamma^p := \{\}, \Delta^d := \{\}, \Delta^p := \{\}, \pi^{\text{ELED}} := \{\}$
$(\Gamma^d, \Gamma^p) := I$ patients are randomly divided into two groups
$\Delta^d :=$ sort Γ^d according to the EDD
$\Delta^p :=$ sort Γ^p according to the LPT
for $i := 0$ **to** $\min(\text{sizeof}(\Delta^d), \text{sizeof}(\Delta^p)) - 1$ **do**
    if $d_{\Delta_i^d} < d_{\Delta_i^p}$ **then**
        $\pi^{\text{ELED}} := \pi^{\text{ELED}} \cup \{\Delta_i^d\}$
        $\Delta^d := \Delta^d - \{\Delta_i^d\}$ ▷ remove Δ_i^d from Δ^d
    else
        $\pi^{\text{ELED}} := \pi^{\text{ELED}} \cup \{\Delta_i^p\}$
        $\Delta^p := \Delta^p - \{\Delta_i^p\}$ ▷ remove Δ_i^p from Δ^p
    end if
end for
if $\text{sizeof}(\Delta^d) > 0$ **then**
    $\pi^{\text{ELED}} := \pi^{\text{ELED}} \cup \{\Delta^d\}$
else
    $\pi^{\text{ELED}} := \pi^{\text{ELED}} \cup \{\Delta^p\}$
end if
return π^{ELED}

Ilustración 5–3. Pseudocódigo de la regla ELED

El procedimiento consiste en dividir de forma aleatoria la lista de pacientes en dos grupos, Γ^d y Γ^p. El grupo de pacientes Δ^d, se obtiene ordenando Γ^d, según la regla EDD y el grupo de pacientes Δ^p, se obtiene ordeando Γ^p, según la regla LPT. Una vez ordenados, se compara la fecha de vencimiento de la primera cirugía entre los dos grupos, y la de valor más bajo se mueve a la secuencia resultado Δ^{ELED}, lo cual es repetido hasta que uno de los dos grupos está vacío, momento en el que las cirugías restantes se añaden al final de la secuencia resultado. De este modo tenemos un procedimiento inteligente para la ordenación de la lista de espera de cirugías, que atiende a la fecha de vencimiento de las mismas, factor importante a considerar en la optimización quirúrgica, y que incorpora a su vez un factor aleatorio que aumenta la diversidad de las soluciones generadas en la inicialización del conjunto para la población inicial de individuos.

5.1.3 Operador de cruce del BMA

Dicha operación de cruce se usa para la generación de dos hijos Os_1 y Os_2, es decir, dos nuevos individuos, y está basado en el cruce de un punto (*One-Point-Crossover*), en el que se selecciona un punto de corte que sirve de referencia para el intercambio de las partes de la secuencia de dos padres Pa_1 y Pa_2, seleccionados de la población actual ϑ de forma aleatoria.

En Ilustración 5–4, se muestra el Algoritmo 4, sobre el procedimiento del operador de Cruce (*Crossover*) basado en un punto de corte.

Algorithm 4 Crossover

Input: instance data, Pa_1, Pa_2
Output: Os_1, Os_2

1: $\pi^{Os_1} := \{\}, \pi^{Os_2} := \{\}$
2: point := randint($\lfloor 0.25 \times \text{sizeof}(\pi^{Pa_1}) \rfloor, \lfloor 0.75 \times \text{sizeof}(\pi^{Pa_2}) \rfloor$)
3: **for** $i := 0$ **to** point **do**
4: $\quad \pi^{Os_1} := \pi^{Os_1} \cup \{\pi_i^{Pa_1}\}$
5: **end for**
6: **for** $i :=$ point $+ 1$ **to** sizeof$(\pi^{Pa_1}) - 1$ **do**
7: $\quad \pi^{Os_2} := \pi^{Os_2} \cup \{\pi_i^{Pa_1}\}$
8: **end for**
9: **for** $i := 0$ **to** sizeof$(\pi^{Pa_2}) - 1$ **do**
10: $\quad$ **if** $\pi_i^{Pa_2}$ **not in** π^{Os_1} **then**
11: $\quad\quad \pi^{Os_1} := \pi^{Os_1} \cup \{\pi_i^{Pa_2}\}$
12: $\quad$ **end if**
13: $\quad$ **if** $\pi_i^{Pa_2}$ **not in** π^{Os_2} **then**
14: $\quad\quad \pi^{Os_2} := \pi^{Os_2} \cup \{\pi_i^{Pa_2}\}$
15: $\quad$ **end if**
16: **end for**
17: Os_1 := Complete List and apply Decoding to π^{Os_1}
18: Os_2 := Complete List and apply Decoding to π^{Os_2}
19: **return** Os_1, Os_2

Ilustración 5–4. Pseudocódigo del operador de cruce basado en un punto de corte.

El procedimiento consiste en dividir la secuencia de pacientes Pa_1 en dos partes, en función de un punto generado de forma aleatoria como indica la línea 2, para posteriormente transferir la parte anterior al punto de Pa_1 a Os_1 y la posterior a Os_2 de forma intacta como indican las líneas 3 – 8. El orden de los pacientes que aún no han sido colocados en cada descendiente se determina por la secuencia de pacientes de Pa_2 como indica las líneas 9 – 15.

En Ilustración 5–5 se muestra un ejemplo en el que Pa_1 se divide aleatoriamente en 3, 4, 5 y 2, 1. La secuencia de las cirugías restantes en Os_1 es 1, 2 según Pa_2. De manera similar, la secuencia de los pacientes restantes en Os_2 es 5, 4, 3.

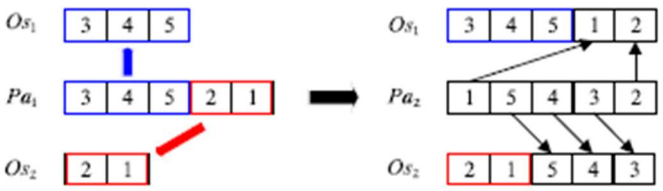


Ilustración 5–5. "Un ejemplo de operación de cruce para productos" (Huang, Pan, & Gao, 2023)

De este modo, el algoritmo realiza una operación de cruce de dos individuos aleatorios, con el fin de dar un salto en el espacio de soluciones y tratar de evitar el riesgo de caer en óptimos locales, mediante la generación de un punto de corte de aleatorio, todo ello para dar diversidad a las soluciones encontradas, durante la búsqueda en el espacio de soluciones. Por otro lado, da prioridad en el orden de la secuencia del conjunto de descendencia al orden que establece el Pa_1, el cual se trata como ya fue comentado del mejor de los dos padres, en base a los objetivos tratados.

5.1.4 Operador de mutación del BMA

Dicho operador de mutación trabaja sobre las dos soluciones generadas en la etapa de cruce Os_1 y Os_2 para generar Os_1'. y Os_2'. y se basa en la operación de inserción, siendo uno de los operadores más frecuentes en la mutación. (Huang, Pan, & Gao, 2023)

En Ilustración 5–6, se muestra el Algoritmo 4, sobre el procedimiento del operador de mutación (*Mutation*) basado en la operación de inserción.

Algorithm 5 Mutation

Input: instance data, Os_1, Os_2, α
Output: Os_1', Os_2'

1: $flag := 0$
2: $\varphi_1 := \{\}$
3: $\varphi_2 := \{\}$
4: **while** $flag < \alpha$ **do**
5: Randomly extract patient δ^{Os_1} from π^{Os_1} without repetition
6: Test δ^{Os_1} at each position of π^{Os_1}
7: $Os_1{}^*$ is the neighbourhood solution generated with the best objective value
8: $\varphi_1 := \varphi_1 \cup \{Os_1{}^*\}$
9: Randomly extract patient δ^{Os_2} from π^{Os_2} without repetition
10: Test δ^{Os_2} at each position of π^{Os_2}
11: $Os_2{}^*$ is the neighbourhood solution generated with the best objective value
12: $\varphi_2 := \varphi_2 \cup \{Os_2{}^*\}$
13: $flag := flag + 1$
14: **end while**
15: select Os_1' from φ_1, Os_1' has a higher value of the main objective function, lower of the secundary objective function in case of tie
16: select Os_2' from φ_2, Os_2' has a higher value of the main objective function, lower of the secundary objective function in case of tie
17: **return** Os_1', Os_2'

Ilustración 5–6. Pseudocódigo del operador de mutación basado en la operación de inserción

Se realiza la extracción aleatoria de uno de los pacientes en ambas soluciones de entrada, Os_1 y Os_2, y se vuelven a insertar en la posición con mejor valor de las funciones objetivos del problema, como indican las líneas 5-7 y 9-11, respectivamente, para posteriormente almacenar cada una de las soluciones generadas Os_1^* y Os_2^* en su correspondiente conjunto de soluciones mutadas, φ_1 y φ_2, línea 8 y 12, respectivamente. A su vez posee una iteración superior la cual permite generar $2 * \alpha$ soluciones mutadas, tratándose éste de un parámetro de entrada no determinado, el cual, si es demasiado grande, repercute en una inversión excesiva del tiempo de ejecución definido, y, por el contrario, si es demasiado pequeño, hace que el rendimiento de la mutación disminuya. Una vez cumple la condición de salida, es decir, α iteraciones realizadas, se selecciona la mejor solución de cada uno de los conjuntos, Os_1' y Os_2', como indican las líneas 15 y 16, las cuales serán devueltas para la siguiente etapa del BMA.

5.1.5 Operador de actualización de la población del BMA

Una vez realizadas las mutaciones y generado un conjunto de descendencia, es decir, conjunto de nuevos individuos generados a partir de operaciones de cruce y mutación de individuos seleccionados de forma aleatoria de la población original, se evalúa si actualizar dicha población por el nuevo conjunto generado.

En Ilustración 5–7, se muestra el Algoritmo 6, sobre el procedimiento del operador de actualización de la población (*Update-Population*).

Algorithm 6 Update-Population

Input: instance data, ϑ_b^*
Output: $\vartheta, improve$

1: $improve :=$ false
2: **if** the new best solution ϑ_b^* is better than the current best solution ϑ_b **then**
3: $\quad \vartheta := \varphi$
4: $\quad improve :=$ true
5: **else**
6: $\quad$ **if** $\forall \vartheta_i \in \vartheta, \vartheta_i \neq \vartheta_b^*$ **then**
7: $\quad\quad$ find the worst individual ϑ_w in ϑ
8: $\quad\quad \vartheta := \vartheta - \{\vartheta_w\}$
9: $\quad\quad \vartheta := \vartheta \cup \{\vartheta_b^*\}$
10: $\quad$ **end if**
11: **end if**
12: **return** $\vartheta, improve$

Ilustración 5–7. Pseudocódigo del operador de actualización de la población

El procedimiento cosiste en determinar en primer lugar si la mejor solución ϑ_b^* obtenida en en las iteraicones internas del BMA, es mejor que la mejor solución actual ϑ_b. Si de ello se trata, la población actual ϑ se reemplaza por el conjunto de descendencia φ y se actualiza la bandera de mejora (*improve*) a verdadero (*true*), para la posterior etapa del BMA. Por el contrario, si no es mejor solución, en base a ninguno de los dos objetivos tratados, y no hay individuo idéntico a ϑ_b^* en la población actual, el peor individuo de la misma ϑ_w es sustituido por la nueva mejor solución.

Dicha operación de actualización de la población no consume mucho tiempo de computación (Huang, Pan, & Gao, 2023). Además, la sustitución de los conjuntos, correspondiente a las líneas 2 – 4 se centra en la optimización de la calidad de la población, desde un punto de vista global. Realizando, al mismo tiempo, cambios sutiles en la misma para que la calidad de las soluciones que la forman vaya en aumento, correspondiente a las líneas 5 – 11, desde un punto de vista local.

5.1.6 Operador de búsqueda local del BMA

Como ya ha sido comentado, se han utilizado dos operadores en distintas etapas del BMA, siendo uno de ellos la versión reducida del otro. Dichos operadores se basan en la búsqueda basada en vecindades mediante la operación de inserción (*Insertion*), tratándose de una búsqueda fuerte, debido a su funcionamiento y propia de los algoritmos meméticos, con el fin de aumentar en gran medida el desempeño de la búsqueda global.

De este modo, el funcionamiento de ambos operadores se basa en realizar la operación de inserción, mediante la selección aleatoria de uno de los elementos del individuo que está siendo trabajado, hasta encontrar otro que mejore en términos de los valores objetivos tratados. Al momento de ser encontrado uno mejor, se vuelve a

iniciar el proceso de búsqueda local, esta vez con el nuevo individuo. En el caso de no encontrar mejora, termina la búsqueda local y se pasa a la siguiente etapa del algoritmo.

La principal diferencia entre el operador general y su versión reducida, es que la condición de parada para la búsqueda local general (*Local Search*) se da en el caso de que haya probado la inserción de todos los elementos de la secuencia y no haya encontrado mejora. En cambio, para su versión reducida se da simplemente con el hecho de que no haya encontrado mejora para el último elemento seleccionado e insertado en las distintas posiciones de la secuencia.

En Ilustración 5–8, se muestra el Algoritmo 7, sobre el procedimiento de la búsqueda local (*Local-Search*)

Algorithm 7 Local-search

Input: ϕ
Output: ϕ

1: $n := (\text{sizeof}(\pi^{\phi}) \times \text{sizeof}(\pi^{\phi}) - 2) + 1$
2: $\eta := 1$
3: **while** $\eta \leq n$ **do**
4: $i := 0$
5: $improve :=$ False
6: randomly extract patient δ^{ϕ} from π^{ϕ} without repetition
7: **while** $i \leq \text{sizeof}(\pi^{\phi}) - 1$ **and** $improve :=$ False **do**
8: test δ^{ϕ} at π_i^{ϕ} without repetition
9: ϕ^{*} is the new neighbourhood solution
10: **if** new solution ϕ^{*} is better than solution ϕ **then**
11: $\phi := \phi^{*}$
12: $\eta := 1$
13: $improve := True$
14: **end if**
15: $\eta := \eta + 1$
16: $i := i + 1$
17: **end while**
18: **end while**
19: **return** ϕ

Ilustración 5–8. Pseudocódigo del operador de búsqueda local

En Ilustración 5–9, se muestra el Algoritmo 8, sobre el procedimiento de la búsqueda local reducida (*Local-Search-Brief*).

Algorithm 8 Local-search-brief

Input: ϕ
Output: ϕ

1: $n := \text{sizeof}(\pi^{\phi})$
2: $\eta := 1$
3: **while** $\eta \leq n$ **do**
4: $\quad i := 0$
5: $\quad improve :=$ False
6: $\quad$ randomly extract patient δ^{ϕ} from π^{ϕ} without repetition
7: $\quad$ **while** $i \leq \text{sizeof}(\pi^{\phi}) - 1$ **and** $improve :=$ False **do**
8: $\quad\quad$ test δ^{ϕ} at π_i^{ϕ} without repetition
9: $\quad\quad \phi^{\star}$ is the new neighbourhood solution
10: $\quad\quad$ **if** new solution $\phi^{\star}$ is better than solution ϕ **then**
11: $\quad\quad\quad \phi := \phi^{\star}$
12: $\quad\quad\quad \eta := 1$
13: $\quad\quad\quad improve := True$
14: $\quad\quad$ **end if**
15: $\quad\quad \eta := \eta + 1$
16: $\quad\quad i := i + 1$
17: $\quad$ **end while**
18: **end while**
19: **return** ϕ

Ilustración 5–9. Pseudocódigo del operador de búsqueda local reducida

El procedimiento consiste en inicializar los parámetros para la condición de parada en base al número de inserciones a realizar, mostrado en las líneas 1 y 2. Posteriormente se pasa a la fase iterativa superior correspondiente al número de inserciones totales, donde se inicializa la bandera para la verificación del cambio de individuo por mejora, a False y se realiza la extracción del elemento de forma aleatoria, atendiendo a su no repetición de iteraciones anteriores, mostrado en las líneas 4 – 6. Una vez seleccionado el elemento a insertar, se procede a su inserción en la secuencia en las distintas posiciones sin repetición, es decir, descartando su posición original (para el caso de la *Local-Search-Brief*), y de forma adicional, la relativa a la primera posición fruto del movimiento originado en la misma debido a inserciones anteriores, y derivado de la extracción aleatoria (para el caso de la *Local-Search*), línea 7 – 8. Una vez generado el nuevo individuo fruto de la inserción del elemento en la correspondiente posición, es evaluado en términos de su función objetivo, obteniendo una solución completa, línea 9. Posteriormente se evalúa si hay mejora; y en caso de que así sea, se actualiza el individuo base de la operación de búsqueda, la bandera de mejora para salir del bucle interno, y los parámetros relativos a las condiciones de parada, dado que se empezará a evaluar un individuo nuevo, mostrado en las líneas 10 – 14. De no encontrar mejora, se aumenta el contador de inserciones realizadas en 1, para el cumplimiento de la condición de parada relativa al número de inserciones totales a realizar, líneas 15 – 16, y se evalúa la siguiente posición y su admisibilidad, hasta recorrer toda la secuencia y haber generado todos los individuos fruto de inserciones admisibles del elemento en cuestión, bucle interno de las líneas 7 – 17, y todos los individuos fruto de las inserciones admisibles de los distintos elementos de la secuencia base, bucle externo de las líneas 3- 18.

Finalmente, se debe añadir que debido a que el objetivo principal, se trata de maximizar el número de cirugías programadas ponderado, se realiza una pequeña operación durante la realización de la evaluación (tests) de los individuos generados en operaciones como la de búsqueda local, mutación, y en definitiva, inserciones donde

se realiza una ligera modificación de una iteración a otra, con el objetivo de aumentar la eficiencia de dichas operaciones, y darle más sentido a su uso.

5.1.7 Operación de completamiento de listas previa a la decodificación

Dicha operación se trata de completar la secuencia a ser evaluada, previo a su evaluación en la decodificación, con los pacientes de la lista de espera que no incorpora en ese momento, y al final de la misma, con el fin de respetar la secuencia que está siendo modificada, ya que la decodificación los planifica en orden de la secuencia dada.

Por otro lado, esta operación se realiza en el nivel más bajo de los operadores, con el fin de respetar la secuencia o individuo que se está trabajando en los mismos, es decir, para la inserción anteriormente descrita empleadas en las búsquedas locales o la mutación solo se extraen y se insertan, aquellos elementos que tiene de entrada y ninguno de los de la lista de espera que la completan posteriormente. Logrando coherencia en su funcionamiento y mejorando su eficiencia computacional.

De este modo, se mejora la eficacia del algoritmo al completo en base al objetivo tratado, ya que, en el mejor de los casos, logrará introducir en la programación realizada pacientes que por la disposición de la secuencia de entrada no lograba introducir.

En resumen, el operador de inserción cambia la disposición de las cirugías de la secuencia de entrada, el de completamiento de la lista, las añade al final para ser evaluadas posterior a las originales, y finalmente es evaluada la resultante en las decodificaciones, logrando mejorar tanto en términos del número de movimientos realizados por los cirujanos, como el total de cirugías programadas ponderadas.

A continuación, en el **Ejemplo 5–1**, se muestra la comparativa de las soluciones obtenidas, para una instancia dada, para el caso de no implementar dicha operación y si implementarla.

Ejemplo 5–1. *Comparativa para la implementación de la operación sobre completamiento de secuencias antes de la decodificación.*

La instancia generada se trata de una lista de espera formada por 10 pacientes, para cuya programación se dispone de 3 quirófanos y 1 día en el horizonte de planificación. Otros datos a considerar de la instancia son.

$I = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

$J = [1, 2, 3]$

$H = [1]$

$t_i = [200, 180, 150, 120, 130, 180, 120, 240, 200, 180]$

$w_i = [0.2, 0.1, 0.4, 0.5, 0.2, 0.3, 0.4, 0.5, 0.6, 0.4]$

$\gamma_i = [7, 8, 6, 5, 2, 1, 5, 4, 6, 2]$

A continuación, se muestra una representación del proceso de programación de pacientes de una lista de espera durante una operación de inserción, para el caso donde no se implementa la completamiento de listas, en la Ilustración 5–10, frente el caso donde si es implementada, Ilustración 5–11. En dichas ilustraciones se presenta la lista de espera inicial, así como la secuencia de entrada para la operación de inserción, π^{inic}; la intermedia para el paso de completar la lista, π_i', y la secuencia final resultado de aplicar la decodificación, π_i, relativas a

una de las iteraciones, i, de la inserción, así como los valores de la función objetivo para cada una de ellas, (OF_1, OF_2). Adicionalemente se muestra un diagrama representativo de la programación para una mejor interpretación.

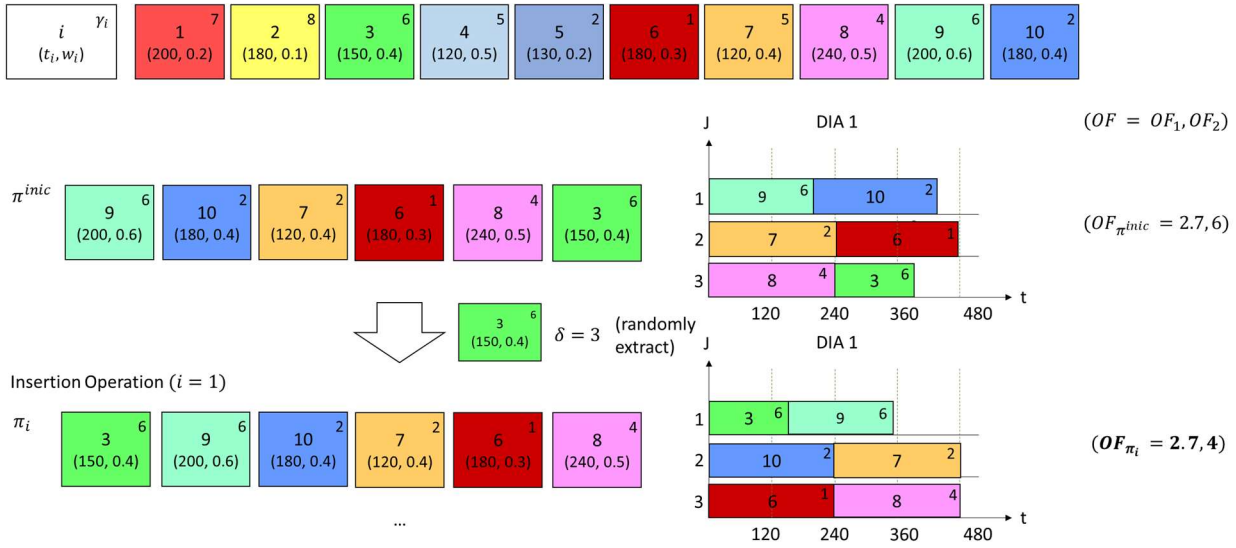


Ilustración 5–10. Ejemplo de solución generada para el caso de no completamiento de la secuencia

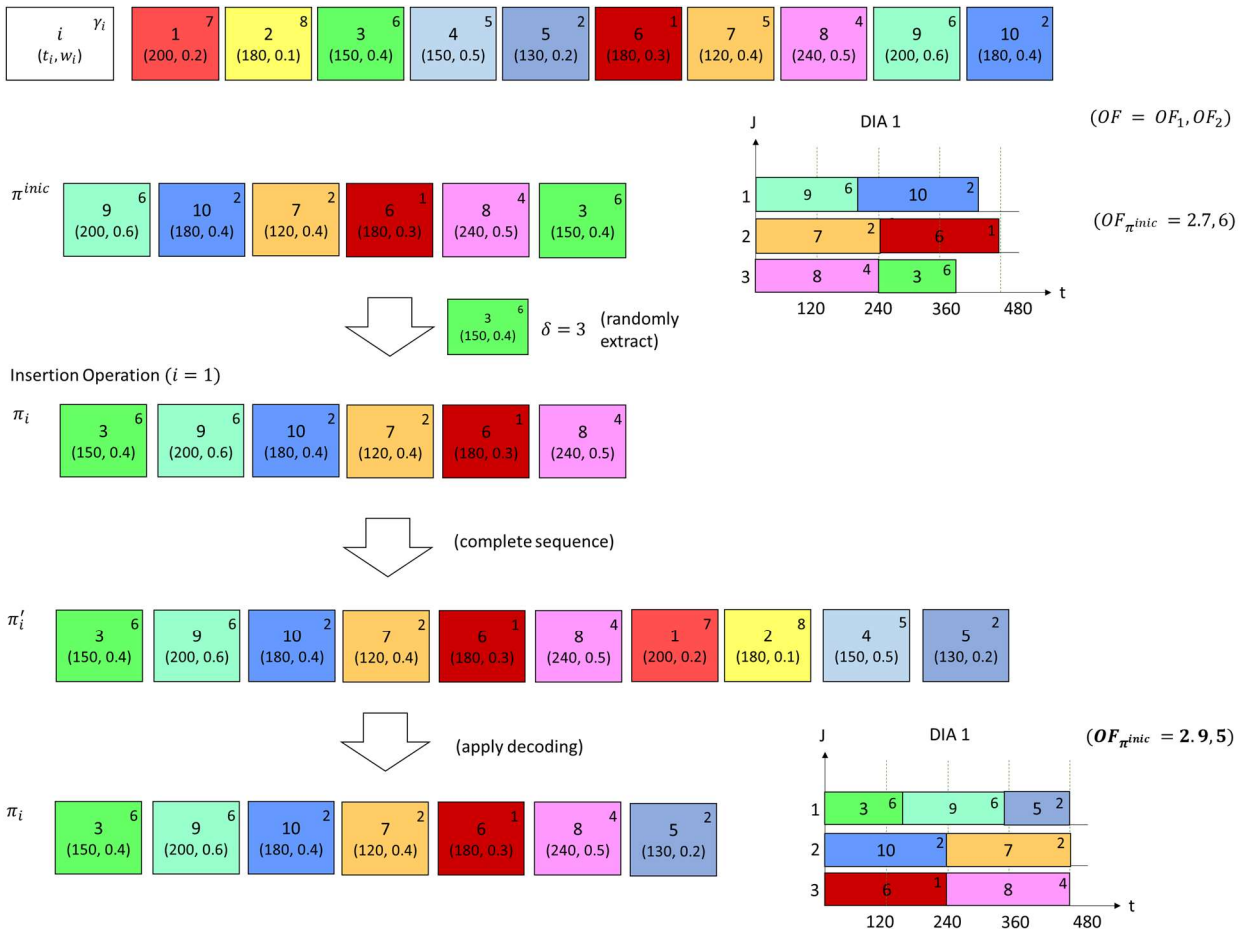


Ilustración 5–11. Ejemplo de solución generada para el caso de completamiento de la secuencia

5.1.8 Otras operaciones de frecuente uso

De forma adicional, se hace mención a una serie de operaciones, similares a la anterior operación sobre completamiento de la secuencia, antes descrita, las cuales, sin llegar a ser operadores genéticos como tal, son de uso frecuente en los distintos operadores que constituyen el algoritmo memético, así como en el marco general del mismo.

A modo de resumen se detalla una lista de las mismas junto a una breve descripción de su funcionamiento y uso en el algoritmo.

Tabla 5–2. Lista de operaciones frecuentes en el algoritmo aproximado

Operación de ordenación	Se trata de la operación (S_i, S_c), descrita en el capítulo 4. Heurísticas, cuya función es la de ordenar las secuencias en base a un determinado indicador y criterio de ordenación. De forma general es empleada en el operador ELED, heurística constructiva para la generación de soluciones iniciales.
Operación de mejor solución de un conjunto	Consiste en la extracción del mejor individuo entre varios pertenecientes a un mismo conjunto. Empleadas para la determinación de las mejores soluciones, por ejemplo, en
Operación de peor solución de un conjunto	Similar a la anterior, en este caso realizando la extracción del peor individuo del conjunto. Empleada, por ejemplo
Operación de comparación de dos soluciones	Permite determinar la mejor entre dos soluciones dadas. Empleada

Finalmente se mencionan, algunas características a considerar de la implementación en Python de dichos operadores, así como su modelo.

- En primer lugar, y relativo a la implementación de los distintos modelos del BMA, cada uno de ellos presenta una clase *(class)* propia, lo cual aporta, entre otras cuestiones, facilidad en el manejo de las distintas funciones que lo componen, seguridad en la presencia de todos los datos relativos a la instancia, así como mayor sencillez en las etapas relativas a la comparación de forma conjunta de los distintos modelos y versiones de los mismos.
- Por otro lado, cada uno de los pseudocódigos correspondientes a los operadores y al marco general del BMA, se definen como funciones (*functions*) dentro de cada uno de los modelos, para poder ser llamadas en cualquier parte del mismo. Por otro lado, los objetos que reciben correspondientes a la instancia, son de tipo *self*, simplificando así su llamada en cada uno de ellos.
- Respecto a las operaciones antes mencionadas, sobre ordenación, operación de mejor solución, etc. se definen como funciones en un archivo propio denominado "*utiles"*, debido a su amplio marco de uso, el cual se ha destinado a este tipo de funciones, así como otras adicionales, derivadas de nuevas versiones del modelo o relativas a la etapa de generación de datos.
- Finalmente comentar que los objetos más usados para el tratamiento de los cálculos generados durante las distintas fases del algoritmo son las listas y las tuplas, debido a su gran relevancia en los problemas de optimización de resolución aproximada mediante Python. Sin embargo, ha sido necesario implementar también diccionarios, por ejemplo, para el registro de posiciones de elementos durante las inserciones de los operadores de *Local Search*, impulsado por la extracción aleatoria del elemento a insertar, y con el objetivo de garantizar su funcionamiento eficiente, sin repeticiones en las secuencias generadas. Este tema es tratado en más detalle en el siguiente capítulo sobre distintas versiones del modelo para su calibración, donde entre otros operadores, se han implementado varios relativos a la búsqueda local y la búsqueda basada en vecindades, debido al papel que desempeña en metaheurísticas basadas en algoritmos meméticos.

5.2 Versiones del algoritmo memético base

Como ya ha sido comentado, el BMA lo componen 6 de los modelos principales de dicho trabajo, los cuales se diferencian únicamente en el tipo de decodificación usada, en base a las reglas de asignación (FF, BF, LF) y la implementación o no para cada una de ellas relativa a la variable $intervals$.

Esto ha sido impulsado por el hecho de que la decodificación es la operación que se encuentra en el nivel más interno de todo el algoritmo, teniendo presencia en prácticamente todos los operadores, y, por ende, dándose su uso con más frecuencia.

De este modo, los resultados de estos 6 modelos se comparan, lo cual es tratado en el capítulo 7 Experimentación computacional, de una manera justa, y determinando cuál de ellos es el mejor para los objetivos tratados y los indicadores planteados, el cual, será posteriormente objeto de calibración para distintas versiones del mismo mediante la implementación de nuevos operadores genéticos, heurísticas constructivas, y modificaciones en la estructura del mismo.

Por ello en dicho subcapítulo, se presentan de forma similar al anterior, los distintos operadores genéticos y el marco general del BMA, los cuales definen estas nuevas versiones.

5.2.1 Versiones relativas a la disposición de la búsqueda local

Como ya ha sido presentado, el uso de operaciones potentes sobre búsqueda local es de gran relevancia en los algoritmos meméticos, ya que mejoran en gran medida el desempeño de la búsqueda global.

Los operadores que definen la versiones relativas a la misma, se han centrado tanto en la posterior búsqueda de mejores resultados con respecto al BMA original, como la verificación de dicha afirmación.

Las versiones pueden ser clasificadas en tres grandes grupos:

- El primero relativo a la disposición de los operadores de búsqueda local original en el marco general, tratado en el subcapítulo 5.2.1 actual.
- El segundo relativo a la verificación del hecho, de que el uso de fuertes operadores de búsqueda local en algoritmos meméticos, para la mejora de su desempeño, mostrado en el subcapítulo 5.2.2
- El tercero relativo al tipo de operación empleada para la generación de vecinos en el original, mostrado en el subcapítulo 5.2.3

Dichas cuestiones serán abordadas en mayor detalle, en el capítulo 6 Banco de pruebas. Presentando en el actual las principales características de estas nuevas implementaciones, a nivel conceptual y de codificación.

En Ilustración 5–12 se muestra el algoritmo de la versión relativa al primer grupo, denominado BMA-LSL o *BMA-Local-Search-Layout*, como referencia en el cambio en la disposición de las mismas.

Algorithm 9 BMA-LSL

Input: instance data, termination criterion, ω
Output: ϑ^b

1: ϑ := a population with ω solutions generated by Initialization
2: perform Local-search on the sequence of best solution ϑ^b in ϑ
3: **repeat**
4: $\varphi := \{\}$
5: **for** $\eta := 1$ **to** $\omega/2$ **do**
6: select Pa_1 and Pa_2 from ϑ randomly, Pa_1 has a higher value of the main objective function, lower value of the secondary objective function in case of a tie
7: $(Os_1, Os_2) :=$ Crossover(Pa_1, Pa_2)
8: $(Os_1{}', Os_2{}') :=$ Mutation(Os_1, Os_2)
9: perform Local-search on the sequence of the better one between $Os_1{}'$ and $Os_2{}'$
10: $\vartheta^{b^\star} :=$ compare $Os_1{}'$ and $Os_2{}'$ with ϑ^b
11: $\varphi := \varphi \cup \{Os_1{}', Os_2{}'\}$ ▷ append $Os_1{}'$ and $Os_2{}'$
12: **end for**
13: $(\vartheta, improve) :=$ Update-population$(\varphi, \vartheta, \vartheta^{b^\star})$
14: **if** $improve =$ true **then**
15: perform Local-search-brief on the sequence of ϑ^b
16: **else**
17: perform Local-search on the sequence of ϑ^b
18: **end if**
19: **until** termination criterion met
20: **return** ϑ^b

Ilustración 5–12. Pseudocódigo sobre el cambio en la disposición de los operadores de búsqueda local

Relativo a su procedimiento, se basa en el empleo de un nuevo marco general, donde la operación de *Local-search-brief*, ha sido sustituida en todo el algoritmo del marco general, por la operación *Local-search*, líneas 2 y 9. Exceptuando el caso en el que la generación de un nuevo individuo a partir de los operadores internos, genere mejora con respecto a la mejor solución de la población actual, línea 15.

De este modo se espera emplear más tiempo en la misma, pero con el fin de mejorar la calidad de la solución, ya que el empleo excesivo del anterior, puede derivar en tiempo computacional desaprovechado, derivado de su finalización, en caso de no encontrar mejora, únicamente para un elemento extraído.

5.2.2 Versiones relativas a la verificación del desempeño de la búsqueda iterativa

Relativo al segundo grupo de versiones, sobre búsqueda local, se basa en el empleo de operaciones genéricas de búsqueda basadas en vecindades, más en concreto de las tres principales denominadas *Adjacent swap*, *Swap* e *Insertion*. Las cuales se diferencian entre sí, en el tipo de operación que realizan en las iteraciones internas, así como el tamaño de la vecindad explorada.

A diferencia del operador de búsqueda local presentado, estas operaciones no son iterativas, es decir, no realizan cambios de vecindad en caso de mejora. Reduciendo en gran medida la robustez del anterior. Adicionalmente, se añaden tanto sus versiones *First Improvement* como *Best Improvement*, las cuales diferencian a la operación en su condición de parada.

A continuación, se detallan cuestiones relativas a lo recién descrito, para una mejor comprensión de dichas operaciones

Relativo al tipo de operación que realizan, referido a la modificación iterativa sobre el individuo que está siendo evaluado, se presenta a continuación una explicación de las mismas en Tabla 5–3, así como un ejemplo para una mejor interpretación en Ilustración 5–13

Tabla 5–3. Funcionamiento de las operaciones basadas en vecindad

Adjacent Swap	Intercambio de todos los elementos con el adyacente a su posición.
Swap	Intercambio de pares de elementos iterativo sin repetición, es decir, entre pares que no hayan sido intercambiados con anterioridad
Insertion	Extracción de todos los elementos y posterior inserción en las distintas posiciones de la secuencia que no generen individuos repetidos

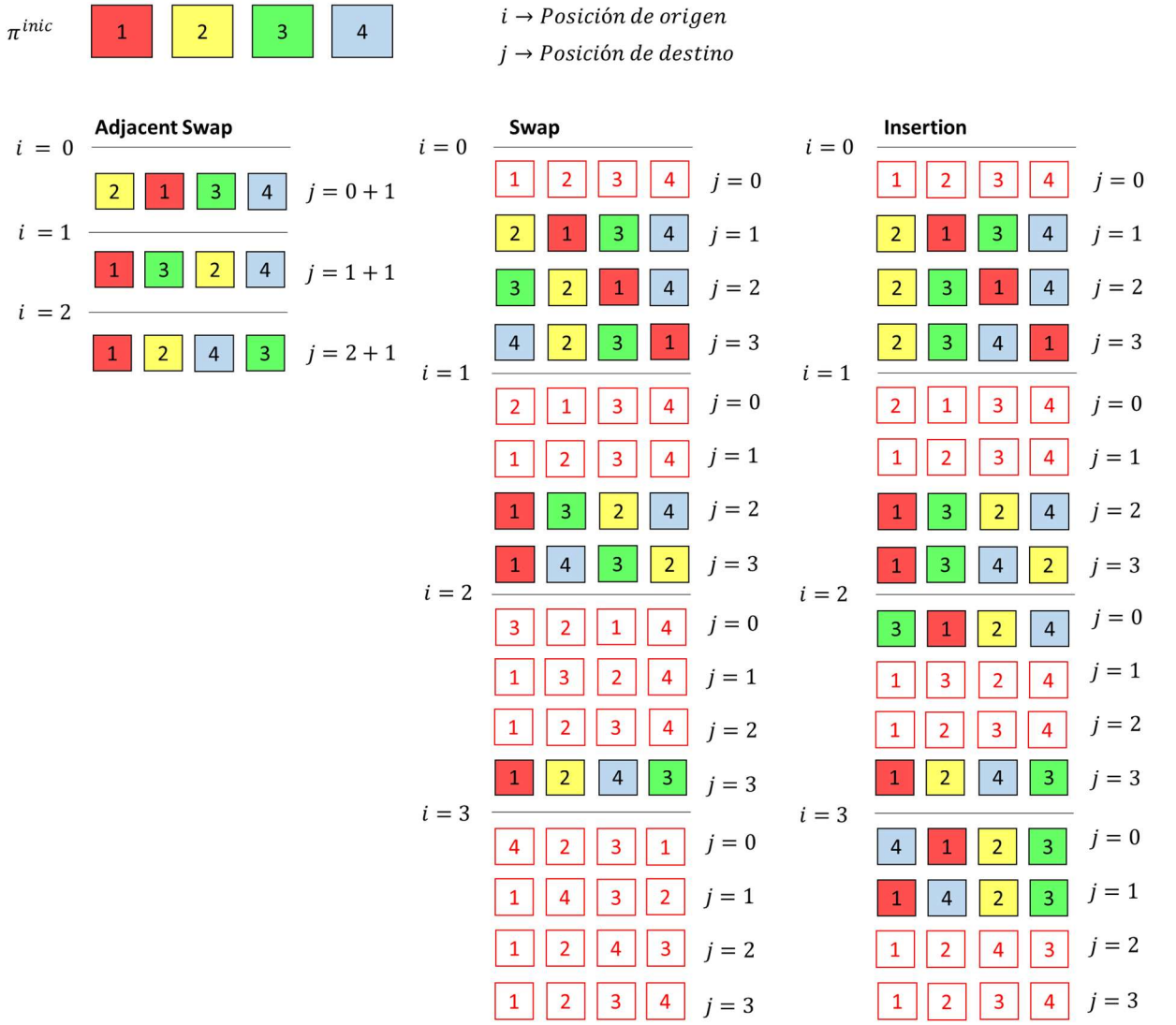


Ilustración 5–13. Ejemplo del funcionamiento de las operaciones basadas en vecindad

Como puede ser observado en la Ilustración 5–13, un factor importante a considerar, es conseguir evitar la generación de individuos repetidos en las distintas modificaciones realizadas, de forma iterativa. Los cuales se dan con bastante frecuencia en las operaciones de *Swap* e *Insertion*, marcadas en rojo. De este modo se logra mayor eficiencia computacional, dado que su evaluación sería tiempo de ejecución desaprovechado.

Por otro lado, se puede observar que la aparición de dichas soluciones repetidas sigue para ambas operaciones, siguen un cierto patrón. Dándose en el intercambio en la posición equivalente a la de procedencia y sus anteriores, y en la de inserción en la posición equivalente a la de procedencia y su inmediatamente anterior.

De este modo, el tipo de operación realizada por las tres operaciones anteriores, sobre la extracción ordenada de elementos de la secuencia empezando por los de la izquierda, simplifica en gran medida las codificaciones en comparación con las requeridas para el caso de sus versiones iterativas, derivado de la extracción aleatoria del elemento a intercambiar implementada.

Por otro lado, y relativo al tamaño de su vecindad, referido al número de posibles vecinos; es decir, diferentes

individuos generados a partir de uno dado, es distinto para cada una de ellas. Este hecho deriva del tipo de operación que realizan antes descrito. A continuación, se presenta el cálculo para el tamaño de sus vecindades, así como una breve explicación de las mismas.

Adjacent Swap: Su cálculo se presenta en la fórmula (5–1), donde se observa que el número de posibles soluciones es el tamaño de la misma – 1, es decir, número de pares adyacentes en la misma.

$$n = sizeof\left(\pi^{inic}\right) - 1 \tag{5–1}$$

Swap: Su cálculo se presenta en la fórmula (5–2), donde el factor multiplicativo se refiere a que, para cada uno de los elementos de la secuencia, tamaño de la misma, se tiene tantas opciones para su intercambio, como posiciones tenga la misma – 1, referida a su posición original. Por otro lado, el factor divisor se debe para evitar la duplicidad derivada del tipo de operación que realiza.

$$n = \frac{\left(sizeof\left(\pi^{inic}\right) * \left(sizeof\left(\pi^{inic}\right) - 1\right)\right)}{2} \tag{5–2}$$

Insertion: Su cálculo se presenta en la fórmula (5–3), donde el factor multiplicativo se refiere a que, para cada uno de los elementos de la secuencia; es decir, su tamaño, se tiene tantas opciones para su inserción, como posiciones tenga la misma – 2, referidas a su posición original, así como su inmediatamente anterior, ya usada en iteraciones anteriores. Por otro lado, el término + 1, es el ajuste, relativo a la inserción adicional del primer elemento a insertar, al momento de comenzar la operación.

$$n = \left(sizeof\left(\pi^{inic}\right) * \left(sizeof\left(\pi^{inic}\right) - 2\right)\right) + 1 \tag{5–3}$$

Por otro lado, y relativo a la condición de parada, referida al criterio implementado para la finalización de la ejecución de dichas operaciones, se emplean las reglas *First Improvement* y *Best Improvement*, las cuales se basan en terminar de realizar la búsqueda, en el caso de haber encontrado mejora en la misma, para el caso de *First Improvement*, o haber evaluado la vecindad al completo, es decir sus n secuencias, para el caso de la *Best Improvement*.

De este modo se pretende realizar una comparación entre las distintas operaciones, así como la verificación de que la búsqueda local general aporta mejores resultados.

5.2.3 Versiones relativas a la modificación de la búsqueda local

Finalmente, y relativo a la modificación de la búsqueda local, se ha realizado una versión análoga para el tipo de operación swap, recién presentado, el cual incorpora de forma análoga una versión normal y otra reducida. Así como la incorporación de aleatoriedad, mediante la extracción de elementos aleatorios de la secuencia.

Relativo al pseudocódigo es similar al mostrado en Ilustración 5–12, cambiando el número de vecinos posibles por el del nuevo tipo de vecindad, siendo similar el resto del proceso. A nivel de codificación tiene un ligero cambio con respecto al anterior en el registro de posiciones en su correspondiente diccionario, el cual se simplifica con respecto al de inserción, pues el tipo de operación, permite registrar las posiciones no repetidas del par de elementos intercambiados de forma simultánea.

De este modo se emplea una nueva operación de búsqueda basada en vecindades, con el fin de generar vecindades distintas a la anterior, y conservar ese alto grado de relevancia de una potente búsqueda local en el desempeño del algoritmo completo.

5.2.4 Versiones relativas al cruce

Con respecto al operador de cruce anteriormente descrito en Ilustración 5–4, se ha empleado uno nuevo, el cual da un mayor grado de variabilidad a la búsqueda global, mediante una generación del par de hijos con modificaciones más significativas con respecto a los padres.

Dicho operador, se basa en introducir dos puntos de corte. Adicionalmente presenta, un nuevo funcionamiento en el método de herencia, es decir, el traspaso de los elementos a los hijos. Por otro lado, tiene la misma disposición y metodología con respecto al marco general que el anterior, recibiendo dos individuos de la población actual, seleccionados de forma aleatoria que actúan como padres, y generando dos hijos, nuevos individuos para el conjunto de descendencia, y objetos de la posterior mutación.

5.2.5 Versiones relativas a la inicialización de la población

Dichas versiones se basan en la incorporación de las reglas de despacho y la heurística constructiva $\mathrm{MT}_{\mathrm{brief}}$, presentada en el capítulo 4 Heurísticas, con el fin de sustituir a la actual sobre la regla ELED, en busca de mejores soluciones iniciales, así como mayor variabilidad en la población inicial.

A diferencia de las versiones antes descritas las cuales se basan en una modificación sobre el modelo base, dichas versiones se basan únicamente en ejecutar ambas heurísticas constructivas, aplicarlas la decodificación y registrar la solución generada de entrada, similar al pseudocódigo mostrado en Ilustración 5–2, pero en dicho caso sin usar ordenación aleatoria, y con un tamaño de la población de 1 y n, para la regla ELED y $\mathrm{MT}_{\mathrm{brief}}$, respectivamente, donde n es el número de heurísticas ST que $\mathrm{MT}_{\mathrm{brief}}$ incorpora.

Todo ello, con el objetivo de ser lo más restrictivo posible en lo que respecta a la calidad de la solución, para determinar la mejor de las heurísticas, objeto de implementación en el operador de generación de la población inicial del algoritmo memético.

6 BANCO DE PRUEBAS

Una vez definidos los algoritmos implementados, y como paso previo a la etapa de experimentaciones computacionales sobre los modelos que constituyen. Debe diseñarse un banco de pruebas robusto y amplio, el cual establezca las bases de la posterior experimentación aportando mayor organización en la misma y validez en los resultados obtenidos. El cual debe atender a las siguientes características.

- Preparación de parámetros y datos, los cuales generen entornos representativos para las posteriores instancias generadas, así como variados entre sí.
- Definición de métricas y condiciones de parada, para una comparación justa de los distintos modelos, así como la valoración de los distintos recursos disponibles
- Implementación eficiente y automatizada para la ejecución de los algoritmos bajo las mimas condiciones y el registro de los resultados obtenidos para los mismos.
- Análisis de los resultados y su desempeño mediante métricas adecuadas para el problema en cuestión, así como representación y documentación de las mismas.

Por ello la disposición de dicho capítulo es la siguiente.

En el subcapítulo 6.1, se define los distintos parámetros para la generación de las distintas instancias y los datos que la forman.

En el subcapítulo 6.2, se hace mención del resto de consideraciones empleadas, relativas a metodologías y buenas prácticas empleadas para su correcta implementación.

6.1 Definición de parámetros y generación de datos

En dicho subcapítulo se introducen los distintos parámetros establecidos para la generación del banco de pruebas, empleados para la generación de las distintas instancias usadas en la posterior etapa de pruebas computaciones, los cuales se han basado en los propuestos en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), donde han sido determinados mediante un exhaustivo análisis de los distintos factores considerados en la literatura y las contribuciones al problema de planificación de quirófanos, así como adaptados al problema bajo consideración.

Dichos parámetros los podemos clasificar en factores y niveles para la construcción del banco de pruebas y parámetros y distribuciones estadísticas empleados para generar los datos de los elementos del problema.

6.1.1 Factores y niveles para la construcción del banco de pruebas

El primer paso relativo a la construcción del banco de pruebas, es la determinación de los distintos factores y niveles que definirán los distintos tipos de instancias generadas, es decir, entorno representativo del problema estudiado.

Estos factores se centran en definir datos como el número de días del horizonte de planificación para realizar la programación de las cirugías, número de salas de operaciones y cirujanos disponibles o el tamaño de la lista de espera con la que trabajar. En Tabla 6–1 se muestra un índice de los implementados, así como la notación empleada para referirse a ellos.

Tabla 6–1. Factores y niveles para la construcción del banco de pruebas

Factores y niveles para la construcción del banco de pruebas	
$\|H\|$	Número de días del horizonte de planificación
$\|J\|$	Número de quirófanos
$\|I\|$	Número de cirugías de la lista de espera
β	Factor de control para la generación del tamaño de la lista de espera (%)
mds	Número máximo de días por semana laboral en los que el cirujano está disponible
$\|S\|$	Número de cirujanos
α	Factor de control para la generación del tamaño de la lista de cirujanos (%)
c	Cantidad máxima del tiempo disponible para cualquier cirujano durante un día
l	Número de semanas laborales
u	Número entero no negativo de salas de operaciones distintas en las que el cirujano s puede realizar cirugías en el mismo día.

A continuación, se menciona la metodología empleada para su determinación, así como los valores extraídos de la literatura impuestos en los mismos y propuestos en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015)

- Número de días del horizonte de planificación ($|H|$): Ha sido definido en 5 días, valor fijo para el total de combinaciones posibles entre dichos parámetros, simulando una semana laboral para la programación de cirugías, acorde al nivel operativo tratado.
- Número de salas de operaciones $|J|$: Puede tomar tanto valor 3 como 9, simulando distintos tamaños de un área especializada para operaciones concretas, propias de los hospitales.
- Número de cirugías en la lista de espera $|I|$: Se determina mediante un tipo de metodología propuesta en algunos artículos, la cual cosiste en generar cirugías una por una hasta que la suma de sus duraciones esperadas exceda un β% el tiempo total disponible de las salas de operaciones para todos los días del horizonte de planificación. Por otro lado, β% se ha establecido en 100 % y 125%, siendo implementado el segundo para contemplar escenarios donde el tiempo total de las cirugías de la lista de espera excede la capacidad total de la sala de operaciones, los cuales son muy comunes debido a presupuestos restrictivos.
- Número máximo de días por semana laboral (mds): Toma valores 3 y 4, simulando distintas composiciones de la semana laboral de los cirujanos.
- Número de cirujanos ($|S|$): Se determina según la expresión (6–1) donde el número de semanas laborales del horizonte de planificación (l) es de valor 1. La cantidad máxima posible de tiempo disponible para cualquier cirujano durante un día en el horizonte de planificación (c) es establecido en 8 horas, valor coincidente con la establecida para la capacidad regular de los quirófanos ($r_j h$), conveniente al representar instantes temporales y realizar la programación de forma conjunta para ambos. Finalmente, el factor de control (α), no es detallado específicamente en la literatura, por lo que se ha establecido en 1.5 y 2.

$$|S| = \alpha * \frac{\sum_{j \epsilon J} \sum_{h \epsilon H} r_{jh}}{l * c * mds} \tag{6–1}$$

- Número de salas de operaciones distintas para realización de cirugías por parte de cada cirujano en un mismo día (u): Toma valor 1 o $|J|$

En Tabla 6–2, se muestra un resumen de los distintos valores posibles recién explicados, base para la construcción del banco de pruebas

Tabla 6–2. Valores para los factores constructores del banco de pruebas

Valores para los factores constructores del banco de pruebas	
$\|H\|$	5
$\|J\|$	$3, 9$
β%	$100\ \%, 125\ \%$
α	$1.5, 2$
mds	$3, 4$
u	$1, \|J\|$

De este modo se dispone de un total de 6 factores, para los cuales se dispone de dos posibles valores para cada uno de ellos, exceptuando el relativo a los días del horizonte de planificación, generando así un total de 32 combinaciones posibles entre ambos. Objeto para la generación de los anteriores parámetros y la generación del resto de datos de la instancia.

6.1.2 Parámetros y distribuciones estadísticas para la generación de los datos

Una vez definido los factores que definen los parámetros principales de la instancia, se deben generar los distintos datos de los elementos de la misma, de una forma adecuada. Para ello, es empleado una serie de parámetros, distribuciones y expresiones seleccionadas de las propuestas en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), las cuales, de forma similar a los anteriores, han sido definidas en base a distintas contribuciones a la literatura.

A continuación, se detalla el procedimiento para la generación de cada uno de los datos del problema.

- Duración de la cirugía (t_i) (en minutos): Sigue una distribución log normal (LN) de dos parámetros. Los cuales son.
    - La duración esperada (μ): Generado de forma aleatoria desde el intervalo $\{60, 120, 180, 240\}$.
    - La desviación estándar (σ): Determinada utilizando el coeficiente de variación, definido como la relación entre σ y μ.

  El coeficiente de variación (CV) es generado de forma aleatoria desde el intervalo $[0.1, 0.5]$.

  La implementación de su cálculo consiste en la generación de los valores de σ y μ según lo establecido, su posterior conversión a parámetros para la distribución log-normal, y la entrada de los mismos a una función de la libraría *numpy* de *Python* para la generación de las duraciones siguiendo dicha distribución. De este modo se implementa un método para la generación de la duración de cirugías de los pacientes de forma adecuada, mediante el uso de una distribución estadística que se adapta a dichos eventos, debido a su factor de no negatividad y posible variación significativa, así como con parámetros de entrada recogidos de la literatura, los cuales no solo tienen en cuenta el tiempo necesario para la realización de la cirugía, sino también su tiempo de preparación, de limpieza y de preparación para la próxima.

- Fecha de vencimiento de la cirugía (d_i) (en días): Se calcula usando la expresión (6–2), donde el tiempo máximo de días antes del tratamiento ($MTBT$), el cual en la realidad depende del Grupo Relacionado con la Urgencia del paciente, se genera de forma aleatoria a partir del conjunto $\{45{,}180, 360\}$; y el número de días en la lista de espera (dwl) se extrae de una distribución uniforme discreta a partir de $[1, MTBT - 1]$, con el fin de evitar fechas negativas

  $$d = MTBT - dwl \tag{6–2}$$

- Peso clínico de la cirugía (w_i): Se obtiene de la combinación lineal, mostrada en la expresión (6–3), entre los valores normalizados de la prioridad médica del paciente (mp^*) y el número de días en los que ha estado en la lista de espera (dwl^*). Donde mp segenera a partir de una distribución uniforme discreta $[1{,}5]$. Las normalziaciones de mp y dwl, se muestran en las expresiones (6–4) y (6–5),

respectivamente. Finalmente, a se establece en 0.5, dando así el mismo peso a ambos parámetros

$$w = a * mp^* + (1 - a) * dwl^* \quad (6–3)$$

$$mp^* = mp/5 \quad (6–4)$$

$$dwl^* = dwl/MTBT \quad (6–5)$$

- Cirujano responsable de la cirugía (γ_i): Se han realizado dos métodos para su generación, uno totalmente aleatorio, en el que se recorre la lista de las cirugías en espera y se le asigna un cirujano responsable de forma aleatoria. Y otro basado en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), el cual realiza también una asignación aleatoria pero más equilibrada que la del primero mediante el uso de un algoritmo, basado en un procedimiento iterativo, donde para cada iteración se asigna a las cirugías la secuencia de cirujanos ordenada aleatoriamente, asignando de este modo una cirugía a cada uno de ellos. Finaliza cuando todas las cirugías han sido asignadas.

  El motivo de ambas implementaciones, es para el segundo de ellos, el propio hecho de un reparto de la carga de trabajo más equilibrado, y evitando posibles situaciones de sobrecarga de cirugías asignadas a un cirujano, imposibles de realizar en el entorno generado. Y el segundo aportar más variabilidad a los datos de entrada con el fin de generar instancias más variadas.

- Elegibilidad de la sala de operaciones (δ_i): Se genera de forma aleatoria, mediante una función la cual da una probabilidad del 90 % a la admisibilidad, representando el resto, grupos de cirugías que deben ser realizadas en quirófanos específicos.

- Capacidad del cirujano responsable en cada día del horizonte de planificación (a_{sh}): Para la generación de la capacidad de los cirujanos se emplean dos tipos de algoritmos, debido a la gran repercusión que tiene sobre la calidad de las soluciones generadas, siendo pieza clave para la generación de un programa u otro.

  Ambas generaciones se basan en un algoritmo dividido en dos fases. En la primera de ellas se crea un horario semanal el cual, determina los cirujanos asignados a cada uno de los días del horizonte de planificación de forma aleatoria, respetando el máximo de días permitidos (mds) para todos ellos. Posteriormente, en la segunda se asigna el valor de la capacidad (a_{sh}) a aquellas combinaciones de cirujanos y días generadas en el horario semanal.

  A continuación, se detallan el funcionamiento del algoritmo empleado en cada uno de ellos, así como las principales diferencias entre sí.

  El primer tipo de generación realiza una asignación aleatoria de los cirujanos sobre los distintos días del horizonte, respetando, a su vez el valor máximo de días por semana laboral a los que poder asignarles cirugías (mds). Por otro lado, hace uso de una función la cual da una probabilidad a dicha asignación con el fin de minimizar el número de días sin asignación.

  El segundo de ellos está basado en la propuesta del artículo (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015). Siendo su principal diferencia con el anterior, la garantía de generar un horario semanal con ausencia de días sin asignación. Sustituye el uso de la función de probabilidad del primero, por un método en el cual se asigna de forma aleatoria tantos cirujanos al día como quirófanos haya.

  El motivo de ambas implementaciones se debe, en el caso del segundo, al propio hecho de que no genera días sin asignación, así como la creación de un horario semanal más equilibrado. En cambio, el primero de ellos, se centra en asignar a un mismo día un mayor número de cirujanos, con el fin de permitir al algoritmo evaluar un mayor número de posibles soluciones, en lo que respecta a las cirugías a realizar, debido a que estas últimas tienen un cirujano responsable asignado de entrada.

- Capacidad regular del quirófano en cada día del horizonte de planificación (r_{jh}): Se establece en 480 minutos.

6.1.3 Generación de varios bancos de pruebas

Como ya ha sido mostrado hay una serie de datos, para los que se ha empleado más de un método en su generación, más en concreto la lista de cirujanos responsables asignados a cirugías (γ_i) y la capacidad de cada uno de esos cirujanos en los distintos conjuntos día – quirófano (a_{sh}), fruto de la creación de un horario semanal.

Su implementación ha consistido, en primer lugar, en emplear la generación de γ_i totalmente aleatoria, así como la generación de la capacidad a_{sh}; relativa a la asignación de un mayor número de cirujanos que quirófanos disponibles en cada día del horario semanal, en las primeras etapas de las pruebas computacionales; relativas a la mejora del algoritmo memético desarrollado, mediante la comparación de modelos base, caracterizados por decodificaciones concretas, así como la posterior calibración del mejor de ellos. Todo ello debido a la mayor variabilidad de las soluciones obtenidas para instancias del mismo tipo con respecto a las segundas generaciones, así como posibles factores de mejora en base a los objetivos tratados, como es el hecho de aportar un mayor número de cirugías posibles para ser evaluadas.

De este modo, las segundas generaciones, sobre la generación de γ_i equilibrada, así como la generación de la capacidad a_{sh}; relativa a un mismo número de cirujanos que de quirófanos disponibles, más en concreto las propuestas en el artículo (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015), se han analizado como posible objeto de mejora para los resultados obtenidos por el algoritmo memético ya calibrado, debido a cuestiones como no dejar ningún día sin asignación y realizar un reparto equilibrado de las cirugías realizadas por los cirujanos; lo cual puede afectar en gran medida al objetivo principal del problema sobre el máximo número de cirugías realizadas, así como la asignación de un número de cirujanos igual al número de quirófanos en cada día del horizonte, lo cual puede afectar en gran medida al objetivo secundario sobre el mínimo de movimientos realizados por los cirujanos.

6.2 Metodologías y buenas prácticas en el diseño del banco de pruebas

Como ya ha sido presentado al comienzo del capítulo y adicional a la correcta generación de los datos y parámetros para la construcción de instancias representativas de diferentes entornos, también es de gran importancia una serie de consideraciones relativas a la efectividad y eficiencia de las posteriores pruebas realizadas.

Por ello en dicho subcapítulo se enumeran las de mayor relevancia, explicando sus características, así como el motivo de su implementación.

6.2.1 Instancias

Relativo a la generación de instancias, se han generado un total de 320, más concretamente 10 para cada una de las 32 combinaciones posibles de los factores antes detallados, con el fin de tener un amplio abanico de distintos datos generados, debido a que una misma combinación de parámetros puede generar dos instancias muy distintas, fruto de la incorporación de aleatoriedad en la generación de los datos.

Por otro lado, dichas instancias se generan desde Python y son exportadas a distintos archivos de texto, los cuales terminan en el número de la instancia generada en el bucle *for*. Esto no solo facilita la generación de las mismas, sino que atiende a otras cuestiones relativas a pruebas computacionales de algoritmos aproximados, como su almacenamiento seguro, su llamada para evaluar dos o más modelos con los mismos datos de entrada.

Relativo a la lectura de las mismas, se ha formalizado el tipo de generación, para que Python pueda interpretar posteriormente de forma correcta si el dato leído se trata de un vector, una matriz o una matriz tridimensional, esto debido al hecho de que en el archivo de texto son objetos de tipo string, y en Python deben ser transformadas a listas de enteros, flotantes, listas de listas, etc, y su correcta interpretación en los algitmos. Para ello se formalizan distintos separadores para dicho reconocimiento, siendo ‘,’ el separador relativo a los elementos del vector, ‘;’, el relativo a los vectores que forman las matrices de dos dimensiones, y ‘|’, el relativo a las matrices de dos dimensiones que forman las de tres. Por otro lado, cada uno de ellos se cierra con corchetes, para que lea de forma correcta cada una de las líneas, se acompañan de su notación, para el

reconocimiento del elemento que se extrae, y el consecuente uso del '=' como símbolo separador entre la notación y la parte numérica. De este modo se formaliza un procedimiento para el correcto trabajo con objetos de tipo "*string"* y la consecuente escritura y lectura de los mismos.

6.2.2 Condición de parada

Por otro lado, y referido a la realización de pruebas de una manera justa, se establece una condición de parada en base al tiempo, la cual depende del tamaño de la instancia, tratando de este modo, el hecho de que haya algoritmos que por defecto tarden más que otros en ser completados, consecuencia de la incorporación de mayores estructuras anidadas y causa posible de ventaja frente a los que tardan menos.

De este modo y debido a la presencia de distintas combinaciones las cuales pueden ser muy distantes en términos de tamaño entre sí, se ha impuesto en el algoritmo completo un límite de tiempo de computación el cual es establecido de forma interna y en base a la expresión (6–6). Donde los tamaños de los conjuntos, se multiplican entre sí y por, η, factor de tiempo el cual se establece en 0.0125 y 0.025.

$$tiempo\ de\ parada = \eta * |I| * |J| * |H| \tag{6–6}$$

De este modo puede ser determinado de forma interna durante la lectura de las distintas instancias y establecido como atributo de las clases de los modelos para su interpretación. Por otro lado, requiere una serie de modificaciones en los algoritmos, fruto del uso de la librería time, más concretamente su función time, la cual registra los segundos transcurridos desde un año en específico.

Dichas modificaciones se basan en bucles *while* que comprueben si se ha alcanzado el tiempo de ejecución, los cuales y debido al funcionamiento del mismo, deben implementarse de forma adicional en el resto de bucles anidados en la misma. En caso contrario, y debido al funcionamiento de los mismos en Python, no se evaluarían hasta completar dichos procesos iterativos de más bajo nivel al completo, pudiendo sobrepasar el tiempo establecido. Adicionalmente los objetos devueltos por las funciones deben introducirse como objetos de tipo self, para que, en caso de darse la condición de parada durante la ejecución de alguno de ellos, la función siempre tenga algo que devolver, ya que en caso de no implementarse puede no llegar a introducir nada en la variable que retorna, dando por consiguiente un error. Finalmente se crean banderas en cada una de las funciones, que determinan si el código interno que llevan asociado a finalizado de forma natural, con el objetivo que acabe la misma, y evitando su permanente ejecución hasta llegar al límite de tiempo establecido.

6.2.3 Eficiencia computacional

Debido a la implementación de la condición de parada en base el tamaño de la instancia, y la consecuente presencia de tiempos que pueden ir desde pocos segundos a algunos minutos, y al hecho de haber establecido un alto número de instancias, se han realizado dos tipos de métodos de ejecución de las pruebas computacionales, referidos a los recursos empleados en los mismos.

El primero de ellos, se ha basado, en la realización de las pruebas referidas a la comparación de los modelos base y al nivel de factor de tiempo superior de 0.025, mediante el uso de 20 ordenadores de forma simultánea, dispuestos en el centro de cálculo de la escuela. Para ello se reparten las 320 instancias en base a su tamaño, con el objetivo de emplear el mismo tiempo de forma aproximada en cada uno de los ordenadores. De este modo se logra reducir en 20 el tiempo total requerido de no realizar dicha metodología, el cual fue calculado previamente en aproximadamente 4,5 h y 9 h, para el factor de tiempo inferior y superior, respectivamente.

Para la posterior etapa de calibración y debido a su enfoque iterativo, de actualización del modelo a calibrar en el momento de encontrar mejora, se han realizado con un ordenador las siguientes características, donde para cada una de las versiones se ha necesitado emplear 4,5 h, debido a la elección del factor de tiempo inferior, ante los resultados obtenidos de la primera, los cuales reflejaban que no hay mucha variabilidad en la calidad de las soluciones obtenidas entre ambos.

6.2.4 Análisis de resultados

Para la evaluación de la efectividad de cada uno de los algoritmos y sus versiones, se ha empleado una métrica de la desviación porcentual relativa (RPD), la cual compara para ambos objetivos y para cada uno de los algoritmos la desviación de la solución dada por cualquiera de ellos, con respecto a la mejor solución, en cada una de las instancias, permitiendo obtener posteriormente la desviación porcentual relativa media del total de las instancias en cada uno de ellos ($\%RPD$).

Los cálculos de RPD para ambos objetivos, se muestran en las (6–7) y (6–8), donde OX_{best} se refiere al mejor valor aportado en la instancia en cuestión entre cada uno de los algoritmos tratados y OX, el del algoritmo para el que se está calculando la desviación. Por otro lado, se observa en ambas fórmulas que la diferencia es adaptada en función de si se está maximizando o minimizando, caso del objetivo primario y secundario, respectivamente.

El cálculo de %RPD, se trata del promedio de las RPD obtenidas para las instancias en cuestión, donde n es el número de instancias o RPD calculados.

$$RPD_1 = \frac{O1_{best} - O1}{O1_{best}} \tag{6–7}$$

$$RPD_2 = \frac{O2 - O2_{best}}{O2_{best}} \tag{6–8}$$

Para una mayor eficiencia de las pruebas realizadas, se introducen los valores en diccionarios, los cuales permiten exportar de forma clasificada e iterativa los resultados obtenidos en cada una de las instancias evaluadas, a tablas de Excel, mediante el uso de la librería pandas.

Este formato además facilita el posterior análisis de los resultados, dado que se trata de un número elevado, a través de métodos y operaciones que incorpora Excel como fórmulas, tablas y gráficas dinámicas, etc.

7 EXPERIMENTACIÓN COMPUTACIONAL

Posterior al diseño y construcción del banco de prueba, se procede a hacer uso del mismo para la ejecución de los algoritmos ideados, con el fin de obtener múltiples resultados para su posterior análisis y sacar conclusiones de los mismos.

Más en concreto, en dicho trabajo se ha realizado un proceso sistemático, el cual ha consistido en la comparación de una serie de algoritmos base, los cuales presentan diferencias significativas en términos de optimización, para posteriormente seleccionar el mejor de ellos, objeto de calibración mediante la modificación iterativa del mismo e implementación de nuevas operaciones. Con el objetivo de mejorar el algoritmo en cuestión, lo máximo posible, en términos de las soluciones generadas y las dos funciones objetivo planteadas, en base a la métrica RPD, introducida en la sección anterior.

Todo ello con el fin de crear un algoritmo eficaz para el problema considerado y ser comparado con otros populares de los problemas de optimización mediante resolución aproximada y contribuciones a la literatura del problema en específico.

Por ello el capítulo se estructura de la siguiente forma.

En el subcapítulo 7.1, se presenta los resultados generados y las conclusiones obtenidas en la comparación de seis modelos base, para 320 instancias, determinando el mejor de todos ellos, objeto de mejora.

En el subcapítulo 7.2, se presenta el proceso iterativo de mejora, así como los resultados que se han ido obteniendo para las 320 instancias anteriores

En el subcapítulo 7.3, finalmente se compara el algoritmo final, en base a los dos métodos para el diseño del banco de pruebas sobre dos de los datos más relevantes en la programación realizada, horario semanal para determinación de las capacidades de los cirujanos y

7.1 Comparación de los modelos base del algoritmo memético

En primer lugar, se realiza la comparación de los modelos base del algoritmo, los cuales varían entre sí en el tipo de decodificación que incorporan. Hay 6 tipos de decodificaciones distintos, los cuales se caracterizan por tener asociada una regla de asignación distinta de las cirugías en los conjuntos días – quirófano, y la implementación o no de la variable relativa al aprovechamiento de huecos admisibles para la realización de cirugías por parte de los cirujanos originados durante la programación.

Por lo tanto, hay 6 modelos a comparar en base a dos tipos de objetivos, máximo de cirugías programadas ponderado (O1) y mínimo número de movimientos realizados por los cirujanos (O2) o cambios de quirófanos, para un total de 320 instancias, en base a la media de la desviación porcentual relativa.

Los tamaños medios de las mismas se muestran en Tabla 7–1, clasificados en base al número de quirófanos J y el factor de control para la generación de las listas de espera, (β), siendo estos los factores que más repercuten en el tamaño de la instancia, y por ende en el tiempo empleado en la misma.

Tabla 7–1. Tamaños medios de las listas de espera para las instancias generadas

3|100	3|125	9|100	9|125
48,825	61,375	143,925	181,0375

En primer lugar, y antes de proceder al análisis de los resultados obtenidos en la etapa de pruebas computacionales, se presenta una tabla resumen con la notación usada para referirse a los distintos modelos a largo del capítulo, así como una breve descripción de las mismas.

Tabla 7–2. Notación de los modelos para pruebas computacionales

Notación de los modelos implementados	
Modelos	
FF	Modelo base con regla de asignación First Fit en la decodificación
BF	Modelo base con regla de asignación Best Fit en la decodificación
LF	Modelo base con regla de asignación Level Fit en la decodificación
FF_nw	Modelo con nueva implementación con regla de asignación First Fit en la decodificación
BF_nw	Modelo con nueva implementación con regla de asignación Best Fit en la decodificación
LF_nw	Modelo con nueva implementación con regla de asignación Level Fit en la decodificación
LF_ls	Modelo de calibración que prioriza la búsqueda local en la disposición general
LF_ls_b	Modelo de calibración que prioriza la búsqueda local reducida en la disposición general
LF_adj	Modelo de calibración que incorpora búsqueda de vecindad basada en la operación Adjacent Swap
LF_swap	Modelo de calibración que incorpora búsqueda de vecindad basada en la operación Swap
LF_ins	Modelo de calibración que incorpora búsqueda de vecindad basada en la operación Insertion
LF_ls_ins	Modelo de calibración que incorpora búsqueda local iterativa y con extracción aleatoria, y usa la operación Insertion
LF_ls_swap	Modelo de calibración que incorpora búsqueda local iterativa y con extracción aleatoria, y usa la operación Swap
LF_ox1	Modelo de calibración que incorpora operador de cruce basado en un punto
LF_ox2	Modelo de calibración que incorpora operador de cruce basado en dos puntos
Otras cuestiones	
$O1\_\{model\}$	Valor objetivo promedio de la función principal para el modelo {model}
$O2\_\{model\}$	Valor objetivo promedio de la función secundaria para el modelo {model}
$\%RPD\_O1\_\{model\}$	Valor de la desviación porcentual relativa media en base al objetivo principal del modelo {model}
$\%RPD\_O2\_\{model\}$	Valor de la desviación porcentual relativa media en base al objetivo secundario del modelo {model}

7.1.1 Comparación relativa a las reglas de asignación

En primer lugar, se muestran los resultados obtenidos para el algoritmo sin la implementación de la variable huecos, para cada una de las reglas de asignación implementadas. Dichos resultados se tratan de los valores medios para ambos objetivos y la media de sus desviaciones porcentuales relativas, siendo %RPD, la métrica empleada para la selección de los mismos. Por otro lado, están clasificados en base a dos de los factores cuyos valores repercuten en mayor medida en la diferencia de los resultados obtenidos entre un tipo de combinación y otra, los cuales son el máximo de quirófanos donde un cirujano puede realizar cirugías en un mismo día (u), y el número de quirófanos de la instancia (J).

De igual modo se han generado resultados tanto para el caso del factor de tiempo tanto inferior como superior, de valor 0.0125 y 0.025, respectivamente.

En primer lugar, se procede a mostrar los resultados obtenidos para el factor de tiempo 0,0125 en Tabla 7–3 y Tabla 7–4.

Tabla 7–3. Resultados O1 para modelos con decodificaciones base (0,00125)

u	J	O1_FF	O1_BF	O1_LF	% RPD_O1_FF	% RPD_O1_BF	% RPD_O1_LF
|J|	3	22,64493	22,26963	23,00231	3,374%	4,794%	2,120%
	9	53,13597	50,09747	66,90368	20,218%	24,566%	0,000%
	Media	37,89045	36,18355	44,95299	11,796%	14,680%	1,060%
1	3	13,25477	13,22625	12,95519	2,283%	2,510%	4,472%
	9	38,22479	38,30759	38,32618	1,710%	1,505%	1,480%
	Media	25,73978	25,76692	25,64068	1,996%	2,007%	2,976%
TOTAL		31,81512	30,97523	35,29684	6,896%	8,344%	2,018%

Tabla 7–4. Resultados O2 para modelos con decodificaciones base (0,00125)

u	J	O2_FF	O2_BF	O2_LF	% RPD_O2_FF	% RPD_O2_BF	% RPD_O2_LF
|J|	3	23,76250	23,72500	26,87500	6,868%	6,414%	22,037%
	9	80,82500	82,20000	92,22500	1,531%	3,363%	16,337%
	Media	52,29375	52,96250	59,55000	4,199%	4,889%	19,187%
1	3	19,57500	19,52500	19,48750	0,466%	0,261%	0,102%
	9	60,58750	60,60000	60,46250	0,204%	0,229%	0,025%
	Media	40,08125	40,06250	39,97500	0,335%	0,245%	0,064%
TOTAL		46,18750	46,51250	49,76250	2,267%	2,567%	9,625%

A continuación, se enumeran una serie de conclusiones sobre los resultados obtenidos en esta comparación.

- Relativo al mejor modelo y por ende la mejor regla de asignación en base a % RPD, la decodificación LF genera los mejores resultados de media, relativos al objetivo principal tratado para todas las combinaciones analizadas
- Relativo a los objetivos, el modelo que incorpora la regla Level Fit es el mejor para el objetivo principal, siendo a su vez el peor para el objetivo secundario, motivo del propio funcionamiento de las reglas. La regla Level Fit asigna en el día – quirófano admisible que más tiempo disponible tenga, en cambio la First Fit y la Best Fit, en el primero que encaje y que menos tiempo disponible tenga, respectivamente. Debido a esto y sumado a que la regla Level Fit programa de media más cirugías que las otras dos, los movimientos que realiza son superiores.
- Otras conclusiones, son el hecho de que ambos objetivos están en conflicto, desventaja para el mínimo número de movimientos al dar prioridad al máximo número de cirugías, así como, la gran relevancia del tamaño del conjunto de quirófanos sobre el máximo de cirugías programadas y por ende los movimientos realizados.

Adicional a las tablas recién presentadas, se muestra una gráfica de los valores promedios para el objetivo principal sobre el máximo de cirugías ponderado obtenido, donde se puede visualizar en mayor medida la causa de que el modelo relativo a la regla Level Fit sea el que mejores resultados aporta.

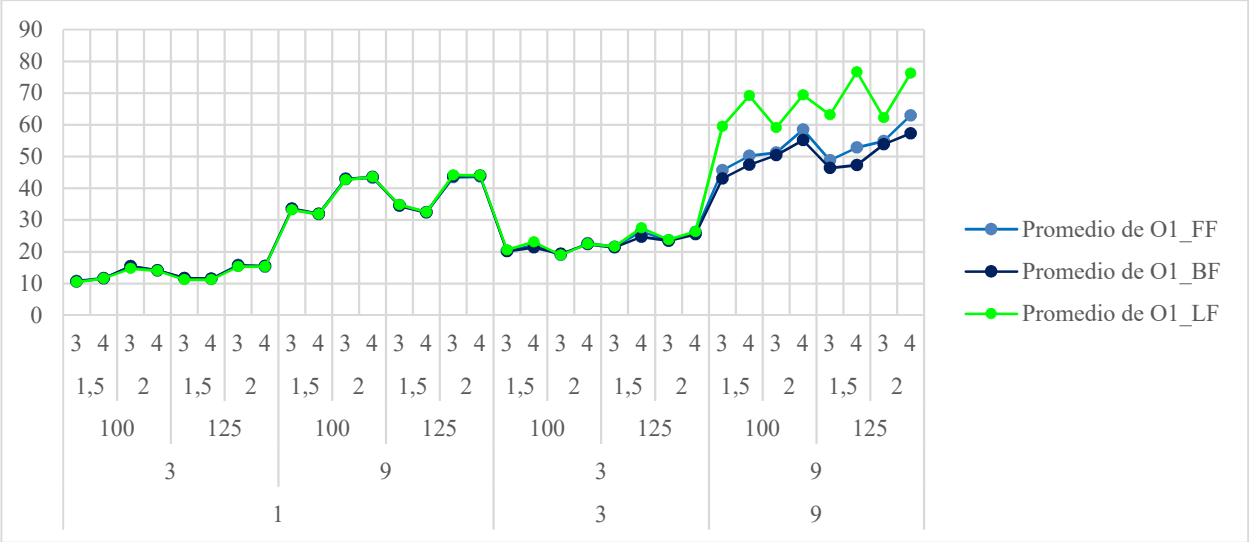


Ilustración 7–1. Gráfica de promedios de O1 para modelos con decodificaciones base

Analizando los resultados, se saca la conclusión de que la combinación relativa, al mayor número de quirófanos, para la no limitación del número de quirófanos en los que el cirujano puede realizar cirugías en un mismo día es la causa de que la regla Level Fit sea la mejor. Siendo para el caso de quirófanos limitados, ($u = 1$), prácticamente idénticas.

A continuación, en Tabla 7–5 y

Tabla 7–6 se muestra los resultados obtenidos para el factor de tiempo superior ($\eta = 0.025$), realizado para la comparación de la calidad de los anteriores modelos en base a su tiempo de ejecución.

Tabla 7–5. Resultados O1 para modelos con decodificaciones base (0,025)

u	J	O1_FF	O1_BF	O1_LF	% RPD_O1_FF	% RPD_O1_BF	% RPD_O1_LF
|J|	3	22,98882	22,55767	24,11021	5,093%	6,682%	0,573%
	9	55,67771	51,41388	67,31175	16,833%	23,070%	0,000%
	Media	39,33326	36,98578	45,71098	10,963%	14,876%	0,287%
1	3	13,40897	13,41927	13,43394	2,553%	2,502%	2,414%
	9	38,45332	38,59205	38,38356	1,861%	1,451%	1,930%
	Media	25,93115	26,00566	25,90875	2,207%	1,976%	2,172%
TOTAL		32,63220	31,49572	35,80986	6,585%	8,426%	1,229%

Tabla 7–6 Resultados O2 para modelos con decodificaciones base (0,025)

u	J	O2_FF	O2_BF	O2_LF	% RPD_O2_FF	% RPD_O2_BF	% RPD_O2_LF
|J|	3	22,81250	24,10000	26,18750	3,605%	10,229%	20,025%
	9	78,97500	81,77500	91,62500	2,545%	6,728%	20,029%
	Media	50,89375	52,93750	58,90625	3,075%	8,478%	20,027%
1	3	19,57500	19,57500	19,56250	0,050%	0,050%	0,000%
	9	60,58750	60,60000	60,36250	0,334%	0,352%	0,000%
	Media	40,08125	40,08750	39,96250	0,192%	0,201%	0,000%
TOTAL		45,48750	46,51250	49,43438	1,634%	4,340%	10,014%

Como se observa se mantiene, el tipo de mejor y peor regla de asignación para la decodificación del modelo que en el caso del tiempo inferior, mejorando todas las soluciones promedio tanto del objetivo primario como del secundario para todas ellas, debido al mayor tiempo impuesto para la ejecución de las mismas.

7.1.2 Comparación relativa a la variable para aprovechamiento de intervalos libres

Posteriormente y de forma análoga a los anteriores modelos analizados en el subcapítulo anterior, se realiza una comparación del mejor modelo base relativo a las reglas de asignación, con la implementación de la decodificación que programa cirugías en intervalos vacíos generados.

En primer lugar, se muestran los resultados obtenidos para el factor de tiempo inferior ($\eta = 0.00125$), en Tabla 7–7 y

Tabla 7–8

Tabla 7–7. Resultados O1 para modelos con decodificaciones con nueva implementación (0,00125)

u	J	O1_FF_nw	O1_BF_nw	O1_LF_nw	% RPD O1_FF_nw	% RPD O1_BF_nw	% RPD O1_LF_nw
|J|	3	22,16505	22,15005	21,88974	2,867%	2,987%	4,022%
	9	63,33061	64,18354	63,47247	3,210%	1,874%	2,774%
	Media	42,74783	43,16679	42,68110	3,039%	2,431%	3,398%
1	3	12,73455	12,80370	12,83877	3,704%	3,079%	2,869%
	9	38,28352	38,21269	38,14854	1,729%	1,872%	2,016%
	Media	25,50903	25,50819	25,49366	2,717%	2,475%	2,442%
TOTAL		34,12843	34,33749	34,08738	2,878%	2,453%	2,920%

Tabla 7–8. Resultados O2 para modelos con decodificaciones con nueva implementación (0,00125)

u	J	O2_FF_nw	O2_BF_nw	O2_LF_nw	% RPD O2_FF_nw	% RPD O2_BF_nw	% RPD O2_LF_nw
|J|	3	27,37500	27,65000	28,61250	6,202%	7,517%	11,379%
	9	95,73750	97,31250	98,80000	1,945%	3,677%	5,413%
	Media	61,55625	62,48125	63,70625	4,073%	5,597%	8,396%
1	3	19,60000	19,63750	19,53750	0,262%	0,432%	0,000%
	9	60,87500	60,87500	60,61250	0,378%	0,378%	0,000%
	Media	40,23750	40,25625	40,07500	0,320%	0,405%	0,000%
TOTAL		50,89688	51,36875	51,89063	2,197%	3,001%	4,198%

A continuación, se enumeran una serie de conclusiones, sobre los resultados obtenidos.

- En primer lugar, se observa, una mejora significativa de los modelos que incorporan las reglas de asignación First Fit y Best Fit, este debido al aprovechamiento de los intervalos generados, llegando a superar al Level Fit relativo a la nueva implementación.
- Por otro lado, empeora los valores promedio del objetivo secundario con respecto a los modelos originales, debido al propio contexto del algoritmo de asignación el cual al asignar en intervalos libres de otros quirófanos también suma movimientos a la función objetivo. Esto no sucede para el caso en el que el número máximo de quirófanos disponibles para un cirujano en el mismo día es uno, pues la asignación adicional no se produce, manteniendo prácticamente el mismo valor.

Adicional a las tablas, en la [] se muestra una gráfica de los resultados para el objetivo principal donde se refleja la similitud de los valores obtenidos para los tres modelos base.

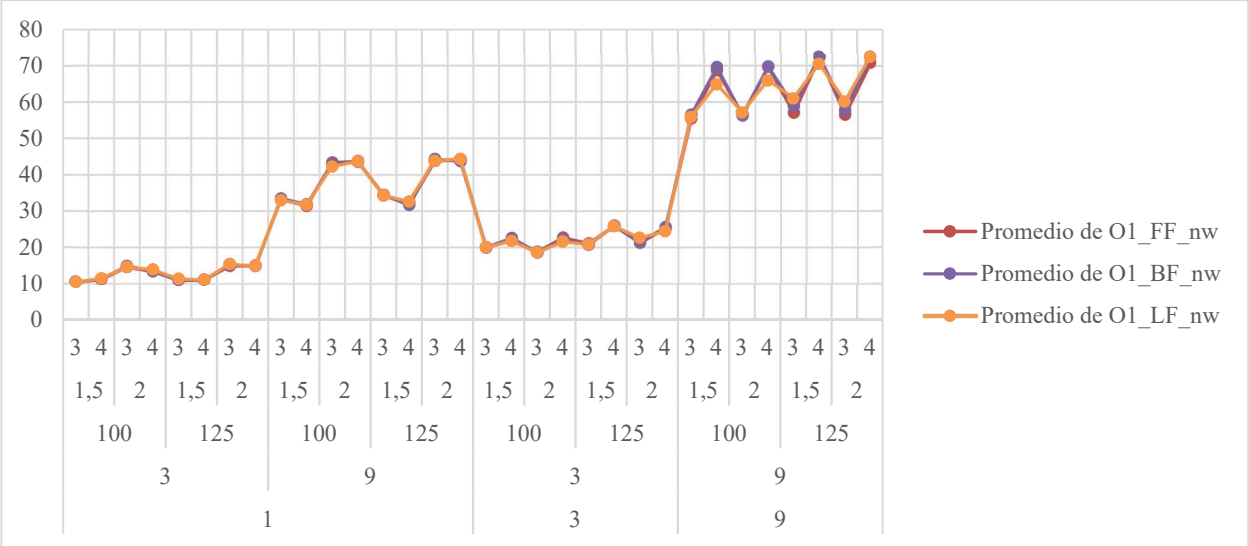


Ilustración 7–2. Gráfica de promedios de O2 para modelos con implementación de *intervals*

De forma análoga a la comparación de los primeros 3 modelos se presenta los resultados obtenidos en Tabla 7–9 y

Tabla 7–10, para los modelos relativas a la nueva implementación para el caso de un tiempo de ejecución en base al factor de tiempo superior ($\eta = 0.025$).

Tabla 7–9. Resultados O1 para modelos con implementación de *intervals* (0,025)

u	J	O1_FF_nw	O1_BF_nw	O1_LF_nw	% RPD O1_FF_nw	% RPD O1_BF_nw	% RPD O1_LF_nw
|J|	3	23,54856	23,68844	23,05489	2,015%	1,419%	4,057%
	9	64,35554	64,65691	62,91425	2,428%	1,898%	4,331%
	Media	43,95205	44,17267	42,98457	2,221%	1,659%	4,194%
1	3	13,14837	13,20241	13,16649	2,825%	2,484%	2,753%
	9	38,24009	38,31724	38,19882	1,783%	1,647%	1,793%
	Media	25,69420	25,75983	25,68266	2,304%	2,066%	2,273%
TOTAL		34,82312	34,966250	34,33362	2,263%	1,862%	3,233%

Tabla 7–10. Resultados O2 para modelos con implementación de *intervals* (0,025)

u	J	O2_FF_nw	O2_BF_nw	O2_LF_nw	% RPD O2_FF_nw	% RPD O2_BF_nw	% RPD O2_LF_nw
|J|	3	25,16250	26,98750	27,63750	4,985%	12,861%	15,892%
	9	96,06250	98,51250	96,73750	2,533%	5,307%	3,490%
	Media	60,61250	62,75000	62,18750	3,759%	9,084%	9,691%
1	3	19,65000	19,66250	19,61250	0,151%	0,201%	0,000%
	9	60,86250	60,87500	60,57500	0,416%	0,436%	0,000%
	Media	40,25625	40,26875	40,09375	0,283%	0,318%	0,000%
TOTAL		50,43438	51,50938	51,14063	2,021%	4,701%	4,846%

De igual forma a la decodificación base, se producen mejoras en todos los resultados para cada una de los modelos.

Con el fin de determinar cuál, de los 6 modelos es mejor, se realiza una comparación conjunta similar a las anteriores para cada uno de ellos. En Tabla 7–11 se muestra el valor medio de la desviación porcentual relativa

(% RPD), obtenido.

Tabla 7–11. Resultados comparación del mejor modelo en base a sus decodificaciones

u	J	% RPD O1_FF	% RPD O1_BF	% RPD O1_LF	% RPD O1_FF_nw	% RPD O1_BF_nw	% RPD O1_LF_nw
|J|	3	3,699%	5,104%	2,447%	5,760%	5,860%	6,923%
	9	20,543%	24,874%	0,406%	5,794%	4,484%	5,403%
	Media	12,121%	14,989%	1,427%	5,777%	5,172%	6,163%
1	3	2,829%	3,053%	5,014%	6,670%	6,072%	5,874%
	9	2,726%	2,520%	2,494%	2,681%	2,824%	2,972%
	Media	2,777%	2,786%	3,754%	4,675%	4,448%	4,423%
TOTAL		7,449%	8,887%	2,590%	5,226%	4,810%	5,293%

En base a los resultados para la métrica analizada, se ha determinado que el modelo, relativo a la regla *Level Fit*, original, sin la implementación del aprovechamiento de intervalos, es el que mejores resultados genera para ambos objetivos tratados. Esto se debe a que la decodificación que implementa la nueva requiere un mayor tiempo de ejecución debido a bucles anidados para lograr su funcionamiento deseado. Por este motivo, y debido a que LF, de forma indirecta y debido a su funcionamiento, ya trata la posible limitación a resolver, es ideal para el objetivo principal estudiado.

7.2 Determinación del mejor modelo base del algoritmo memético

Una vez determinado el mejor de los modelos base en función de la decodificación que incorpora, función de más bajo nivel del algoritmo completo. Se procede a realizar una evaluación computacional para determinar el mejor algoritmo memético mediante un análisis sistemático a través de modificaciones iterativas sobre los operadores que incorpora, así como parámetros de entrada.

En Tabla 7–12 se presenta un resumen de los distintos pasos o iteraciones realizadas, así como una breve descripción de las modificaciones realizadas

Tabla 7–12. Serie de iteraciones realizadas en la etapa de calibración

1ª iteración	Calibración de los operadores de búsqueda local	Modificación en la disposición de los operadores de búsqueda local originales en base a su versión normal y su versión reducida
2ª iteración		Comparación con operaciones basadas en vecindades no iterativas.
3ª iteración		Modificación de la operación usada en las búsquedas locales originales sobre inserción a intercambio de pares.
4ª iteración	Calibración del operador de cruce	Modificación del operador de cruce basado en un punto de corte a uno nuevo basado en dos puntos de corte
5ª iteración	Calibración del operador de generación de la población inicial	Comparación de la heurística constructiva original ELED, con la versión de MT, para la generación de un conjunto de soluciones inicial.
6ª iteración	Comparación para diversos bancos de prueba	Comparación de resultados mediante implementación de distintas políticas de gestión

Para la calibración de las iteraciones 1 – 5, relativa a operadores y marco general del modelo se ha empleado las 320 instancias usadas en el subcapítulo anterior, así como el factor de tiempo inferior ($\eta = 0.0125$) para la condición de parada de los algoritmos debido a las conclusiones sacadas sobre el mismo.

Para la calibración final, relativa a la generación del banco de pruebas se han empleado 320 nuevas instancias, las cuales incorporan los datos generados por los algoritmos propuestos como posible escenario de mejora.

A continuación, se procede a mostrar los resultados obtenidos y sus conclusiones, en el orden de las iteraciones realizadas en dicha etapa de evaluación computacional

7.2.1 Evaluación computacional relativa a la disposición de las búsquedas locales

A modo de resumen, de lo detallado en el capítulo 5 sobre los operadores de búsqueda local, originales del modelo base, se han implementado en primera instancia dos distintos de forma conjunta en el marco general del BMA, los cuales se basan en la búsqueda basada en vecindades de forma iterativa, mediante la operación de inserción. Agregando, adicionalmente, aleatoriedad a dicho operador en la extracción del elemento a insertar.

La principal diferencia entre ambos es la condición de parada. Más en concreto el operador normal, termina cuando se han evaluado todas las posibles soluciones sin repetición de la operación de inserción para la última secuencia sin mejora, y su versión reducida para la inserción sin mejora del último elemento extraído en cada una de las posiciones de la secuencia.

De este modo, la versión normal, recorre en mayor medida el espacio de soluciones, empleando en contraposición, más tiempo durante su ejecución que la versión reducida.

Por ello la primera calibración de todas, se enfoca en la disposición de dichos operadores en el marco general del BMA, más en concreto, intercambiando las búsquedas locales reducidas por la normales en una serie de puntos del mismo, según lo detallado en 5.2.1

De este modo en las figuras Tabla 7–13 y

Tabla 7–14 se presentan los resultados obtenidos de la comparación de la versión original (LF_ls_b) con la nueva (LF_ls)

Tabla 7–13. Resultados O1 para calibración sobre la disposición de las búsquedas locales

u	J	O1_LF_ls	O1_LF_ls_b	% RPD O1_LF_ls	% RPD O1_LF_ls_b
|J|	3	23,47873	23,00231	0,517%	2,650%
	9	67,46649	66,90368	0,844%	1,618%
	Media	45,47261	44,95299	0,680%	2,134%
1	3	12,98703	12,95519	1,948%	2,053%
	9	38,19868	38,32618	1,355%	1,005%
	Media	25,59285	25,64068	1,652%	1,529%
TOTAL		35,53273	35,29684	1,166%	1,832%

Tabla 7–14. Resultados O2 para calibración sobre la disposición de las búsquedas locales

u	J	O2_LF_ls	O2_LF_ls_b	% RPD_O2_LF_ls	% RPD_O2_LF_ls_b
|J|	3	17,42500	17,67500	2,247%	3,734%
	9	57,07500	57,12500	1,270%	1,342%
	Media	37,25000	37,40000	1,758%	2,538%
1	3	19,61250	19,57500	0,207%	0,052%
	9	60,66250	60,67500	0,083%	0,089%
	Media	40,13750	40,12500	0,145%	0,070%
TOTAL		38,69375	38,76250	0,952%	1,304%

De este modo, se observa que la modificación en la disposición de las búsquedas locales, mejora los resultados obtenidos para los valores promedios de ambos objetivos, en base al % RPD. En contraposición, la versión que

predomina el uso de la búsqueda local reducida es ligeramente superior que la nueva para las instancias con un mayor número de quirófanos, más en concreto 9. Esto sucede debido a que el tiempo que se emplea a la búsqueda local es excesivo en este caso, pues incrementa exponencialmente en base al aumento de la secuencia, reflejando que la búsqueda del equilibrio entre el tiempo de ejecución empleado para los distintos operadores y la calidad de las soluciones obtenidas, es una tarea compleja de desarrollar en la resolución aproximada.

7.2.2 Evaluación computacional relativa al desempeño de la búsqueda iterativa

Una de las principales características favorecedoras para la calidad de las soluciones obtenidas, es el empleo de búsquedas locales iterativas, las cuales, a diferencia de sus versiones más genéricas, no limitan la búsqueda a una sola vecindad, si no que la actualizan, en función de si encuentran mejora. De este modo dicha iteración se centra en verificar la premisa sobre las mejoras fruto de la implementación de búsquedas locales iterativas, y realizar una comparación de los resultados obtenidos por las operaciones base de la búsqueda basada en vecindades, para identificar posibles escenarios de mejora

En primer lugar, se compara la versión de la *Local Search* basada en inserción (LF_ls), con su versión genérica (LF_ins), cuyos resultados se observan en la Tabla 7–15 y

Tabla 7–16.

Tabla 7–15. Resultados O1 para verificación del desempeño de la búsqueda local iterativa

u	J	O1_LF_ls	O1_LF_ins	% RPD_O1_LF_ls	% RPD_O1_LF_ins
|J|	3	23,47873	23,16439	0,766%	2,136%
	9	67,46649	67,05451	0,743%	1,289%
	Media	45,47261	45,10945	0,755%	1,712%
1	3	12,98703	12,94543	1,694%	2,041%
	9	38,19868	38,32028	1,293%	0,969%
	Media	25,59285	25,63286	1,494%	1,505%
TOTAL		35,53273	35,37115	1,124%	1,609%

Tabla 7–16. Resultados O2 para verificación del desempeño de la búsqueda local iterativa

u	J	O2_LF_ls	O2_LF_ls_b	% RPD O2_LF_ls	% RPD O2_LF_ls_b
|J|	3	26,45000	26,87500	3,824%	5,993%
	9	91,60000	92,22500	1,566%	2,161%
	Media	59,02500	59,55000	2,695%	4,077%
1	3	19,53750	19,48750	0,265%	0,054%
	9	60,48750	60,46250	0,150%	0,107%
	Media	40,01250	39,97500	0,208%	0,081%
TOTAL		49,51875	49,76250	1,451%	2,079%

Según los datos obtenidos, se determina que el empleo de un método iterativo, que incluya aleatoriedad, es la mejor opción para el problema de planificación de quirófano, destacando en los resultados obtenidos para el caso de no limitación en el número de quirófanos donde realizar cirugías por parte de los cirujanos. Por otro lado, no presentan una variación significativa con respecto al problema de quirófanos limitados, debido al contexto del mismo.

Una vez realizada la aplicación, se realiza una comparación para las distintas operaciones genéricas antes detalladas (*Adjacent swap, Swap, Insertion*), debido a que en primera instancia se ha determinado usar *Insertion* sin un respaldo experimental para el problema concreto a resolver y sus objetivos a mejorar.

A modo de resumen, las modificaciones se basan en generar una nueva versión del modelo, para cada una de

las operaciones, sobre inserción (*Insertion*), intercambio (*Swap*) e intercambio adyacente (*Ajacent Swap*), mediante la sustitución de las búsquedas locales (*Local Search*) por su versión con condición de parada basada en la regla *Best Improvement* y su versión reducida (*Local-Search-Brief*) por la basada en la regla *First Improvement*.

De este modo en Tabla 7–17 y

Tabla 7–18, se muestran los resultados obtenidos para cada una de estas operaciones.

Tabla 7–17. Resultados O1 para los modelos con operaciones sobre vecindades

u	J	O1_LF_adj	O1_LF_swap	O1_LF_ins	% RPD O1_LF_adj	%RPD O1_LF_swap	% RPD O1_LF_ins
	3	22,98757	23,50491	23,16439	3,081%	0,920%	2,341%
|J|	9	66,42599	67,87295	67,05451	2,793%	0,668%	1,858%
	Media	44,70678	45,68893	45,10945	2,937%	0,794%	2,100%
	3	13,01601	12,93738	12,94543	2,219%	2,879%	2,813%
1	9	38,42868	38,29852	38,32028	1,408%	1,720%	1,657%
	Media	25,72234	25,61795	25,63286	1,813%	2,299%	2,235%
TOTAL		35,21456	35,65344	35,37115	2,375%	1,547%	2,167%

Tabla 7–18. Resultados O2 para los modelos con operaciones sobre vecindades

u	J	O2_LF_adj	O2_LF_swap	O2_LF_ins	% RPD O2_LF_adj	%RPD O2_LF_swap	% RPD O2_LF_ins
	3	27,75000	27,92500	27,68750	7,351%	7,943%	7,144%
|J|	9	91,35000	93,35000	92,42500	1,978%	4,291%	3,204%
	Media	59,55000	60,63750	60,05625	4,664%	6,117%	5,174%
	3	19,53750	19,53750	19,50000	0,269%	0,269%	0,098%
1	9	60,36250	60,51250	60,41250	0,125%	0,360%	0,221%
	Media	39,95000	40,02500	39,95625	0,197%	0,315%	0,160%
TOTAL		49,75000	50,33125	50,00625	2,430%	3,216%	2,667%

Analizando las tablas se sacan las siguientes conclusiones.

- En primer lugar y relativo al % RPD del objetivo principal, la operación *Swap* es con diferencia la mejor de todas para el problema de planificación de quirófanos con maximización del número de cirugías programadas ponderado.
- Por otro lado, es seguida por las operaciones *Insertion* y *Adjacen*t. Este hecho refleja que la vecindad relativa a la operación swap, es la mejor para el tipo de problema tratado, ya que la Insertion es limitada por la gran vecindad que posee, empleando un sobretiempo de ejecución innecesario y desfavorecedor. La *Adjacent* a su vez es favorecida por su tipo de vecindad, siendo una versión muy reducida de la swap, pero de forma análoga a la anterior, es limitada por su tamaño, el cual, en este caso, es insuficiente para generar soluciones de mejor calidad como la swap. Por este motivo la vecindad *Swap* es perfecta para los objetivos tratados en el problema, logrando el equilibrio entre el tiempo y el tamaño empleado.

De esta forma, se llega a la conclusión de que la operación swap es una operación a implementar, con potencial de mejora para el algoritmo.

7.2.3 Evaluación computacional relativa a la modificación de las búsquedas locales

A continuación, se presenta la siguiente y última evaluación relativa a las búsquedas locales, la cual deriva de

los resultados obtenidos de las iteraciones anteriores sobre el factor iterativo y los distintos tipos de vecindades.

Por ende, se realiza una modificación en el algoritmo, mediante la implementación de un operador que combina los mejores aspectos analizados con anterioridad, el cual se trata de una búsqueda local iterativa y con incorporación de aleatoriedad, al igual que la actual usada hasta el momento, y en este caso, basada en la operación swap. De igual modo, se mantiene ambas versiones, normal (*Local_Search_Swap*) y reducida (*Local-Search-Swap-brief*), relativas a la condición de parada, así como su disposición anterior en el marco general.

De este modo, se presenta en la tabla Tabla 7–19 y

Tabla 7–20, la comparación de los resultados obtenidos para el algoritmo con la *Local Search* original, basada en Insertion, y la nueva implementada basada en swap.

Tabla 7–19. Resultados O1 para los modelos con búsqueda local iterativa y aleatoria

u	J	O1_LF_ls_ins	O1_LF_ls_swap	% RPD_O1_LF_ls_ins	% RPD_O1_LF_ls_swap
|J|	3	23,47873	23,96923	2,429%	0,424%
	9	67,46649	68,68710	2,235%	0,408%
	Media	45,47261	46,32817	2,332%	0,416%
1	3	12,98703	13,01368	2,296%	2,213%
	9	38,19868	38,50083	1,515%	0,705%
	Media	25,59285	25,75725	1,905%	1,459%
TOTAL		35,53273	36,04271	2,119%	0,937%

Tabla 7–20. Resultados O2 para los modelos sobre búsqueda local iterativa y aleatoria

u	J	O2_LF_ls_ins	O2_LF_ls_swap	% RPD_O2_LF_ls_ins	% RPD_O2_LF_ls_swap
|J|	3	26,45000	25,61250	7,527%	4,233%
	9	91,60000	93,41250	1,201%	3,204%
	Media	59,02500	59,51250	4,364%	3,718%
1	3	19,53750	19,53750	0,113%	0,096%
	9	60,48750	60,48750	0,060%	0,055%
	Media	40,01250	40,01250	0,086%	0,075%
TOTAL		49,51875	49,76250	2,225%	1,897%

Como se observa, el cambio de la operación *Insertion* por la operación swap en los operadores de búsqueda local, genera mejoras en las soluciones obtenidas en base a los objetivos tratados.

Por otro lado, se observa que, con respecto al objetivo secundario, toman valores muy próximos, en especial para las combinaciones correspondientes al problema con limitación de quirófanos donde poder realizar cirugías por parte de los cirujanos en el mismo día, ($u = 1$), donde de media son iguales. Este caso también sirve de ejemplo, para la mejor comprensión de la métrica relativa al RPD, la cual, a diferencia del promedio, el cual compensa las variaciones del valor objetivo entre ambos modelos para las distintas instancias, generando un promedio similar, en el caso del RPD no sucede, debido al factor divisor que incorpora, introducido en (6–8).

7.2.4 Evaluación computacional relativa al operador de cruce

Una vez evaluada la búsqueda local, operadores basados en ligeras modificaciones sobre una secuencia dada, para la obtención de la mejor en esa región del espacio, denominada óptimo local. Se procede a implementar un nuevo operador de cruce, el cual aporta más variabilidad. El objetivo es analizar cuestiones relativas a

realizar cambios más significativos en las soluciones para explorar más zonas del espacio, si el algoritmo es propenso a la convergencia prematura, y en definitiva si una mayor diversidad de las soluciones genera mejoras en las soluciones obtenidas.

Para ello se ha sustituido el operador de cruce basado en un punto, por uno basado en dos, e implementado en el mejor modelo obtenido en las iteraciones predecesoras a la actual.

En Tabla 7–21 y

Tabla 7–22 se muestra los resultados de la comparativa para ambas versiones.

Tabla 7–21. Resultados O1 para calibración relativa al cruce

u	J	O1_LF_ox1	O1_LF_ox2	% RPD O1_LF_ox1	% RPD O1_LF_ox2
|J|	3	23,96923	23,86303	1,252%	1,628%
	9	68,68710	68,60811	0,829%	0,968%
	Media	46,32817	46,23557	1,041%	1,298%
1	3	13,01368	13,00562	1,840%	1,662%
	9	38,50083	38,61146	1,270%	1,004%
	Media	25,75725	25,80854	1,555%	1,333%
TOTAL		36,04271	36,02206	1,298%	1,316%

Tabla 7–22. Resultados O2 para calibración relativa al cruce

u	J	O2_LF_ox1	O2_LF_ox2	% RPD_O1_LF_ox1	% RPD_O1_LF_ox2
|J|	3	25,61250	24,61250	8,902%	3,765%
	9	93,41250	93,13750	2,379%	2,081%
	Media	59,51250	58,87500	5,640%	2,923%
1	3	19,53750	19,52500	0,098%	0,054%
	9	60,48750	60,51250	0,053%	0,094%
	Media	40,01250	40,01875	0,076%	0,074%
TOTAL		49,76250	49,44688	2,858%	1,499%

En base a los resultados obtenidos, se puede llegar a la conclusión de que introducir un grado más de diversidad a las soluciones generadas del algoritmo, no afecta en gran medida al objetivo principal. Lo cual puede ser debido, al hecho de que una ordenación u otra de la secuencia, genera el mismo valor objetivo principal, viéndose este únicamente incrementado en el caso de que se logre programar alguna cirugía más. Por el contrario, se observa una mejora significativa del valor del objetivo secundario, el cual en caso de ser evaluado si depende en mayor medida del orden de la secuencia, dado que será la base del orden de programación de las cirugías que la forman, y, por ende, de la disposición de las mismas en el programa, afectando a los movimientos que realizan los cirujanos.

7.3 Comparación del modelo base calibrado para distintos bancos de pruebas

Una vez determinado el mejor modelo de los implementados para el problema estudiado, relativo a una decodificación con regla de asignación Level Fit, y búsquedas locales basadas en la regla Swap, siendo las características más significativas y relevantes sobre sus resultados, se procede a realizar una comparación de los resultados obtenidos para dos políticas de gestión distintas, las cuales se caracterizan por presentar dos métodos distintos para el establecimiento de los valores de dos de los datos más relevantes en el problema tratado.

De este modo, la principal diferencia de ambas es:

El tipo de generación de la variable a_{sh}, más en concreto el horario semanal de asignación de cirujanos a días

El tipo de generación de la variable γ_i, sobre la asignación de cirujanos a cirugías de la lista de espera

La primera política de gestión, establece el banco de pruebas usado hasta el momento, la cual se basa en generar un γ_i, puramente aleatorio y un horario semanal, el cual puede asignar a un mismo día, más cirujanos que los del número de quirófanos en el área. Su implementación radica en aprovechar una mayor variabilidad de las duraciones de las cirugías a programar, por el hecho de haber más cirujanos asignados al día. Sin embargo, presenta la desventaja de no asegurar asignaciones en todos los días, pues utiliza para minimizar este riesgo una función basada en la generación de una probabilidad de asignación al día.

La segunda política de gestión, establece el nuevo banco de pruebas, cuyo diseño es el propuesto en (Molina-Pariente, Hans, Framinan, & Gomez-Cia, 2015). En comparación al primero, es más realista, ya que asigna un número menor de cirujanos a cada día, así como un reparto más equilibrado en las cirugías asignadas a cirujanos. Sin embargo, reduce el espacio de soluciones del anterior en gran medida, debido a que requiere para dicha garantía, del uso de un algoritmo el cual asigna un número de cirujanos al día, de valor coincidente al número de quirófanos.

Para su evaluación, se crea un nuevo conjunto de 320 instancias, con el fin de tener dos pares de conjuntos de valores para los datos presentados, generados cada uno según las políticas de gestión definidas, y que el resto de datos, como las duraciones de las cirugías o el peso de las mismas sea coincidente para ambas. Por otro lado, el algoritmo empleado es el mejor encontrado hasta el momento y para el análisis de los resultados generados se calcula los valores medios para ambas combinaciones y en base a los factores más diferenciales empleados hasta el momento.

A continuación, se muestra los resultados de los valores medios de ambas políticas, para la interpretación de una serie de conclusiones.

Tabla 7–23. Promedios para el primer banco de pruebas implementado

u	J	**O1_política_1**	**O1_política_2**
|J|	3	23,42811	22,29680
	9	64,52236	61,20993
	Media	43,97523	41,75337
1	3	10,19495	13,01825
	9	28,58535	38,33672
	Media	19,39015	25,67748
TOTAL		31,68269	33,71542

Tabla 7–24. Promedios para el segundo banco de pruebas implementado.

u	J	**O2_política_1**	**O2_política_2**
|J|	3	25,61250	23,27500
	9	93,41250	95,26250
	Media	59,51250	59,26875
1	3	19,53750	14,96250
	9	60,48750	44,90000
	Media	40,01250	29,93125
TOTAL		49,76250	44,60000

Como puede ser observado de media, el banco de pruebas basado en el reparto equilibrado de cirugías a cirujanos y la creación del horario semanal supera en ambos objetivos al original. Hecho no tan relevante debido al uso de datos de entrada distintos, pero aceptable, debido el gran número de las mismas que se ha usado

Por otro lado, se observa que el banco de pruebas original ofrece mejores resultados de media para el objetivo principal y la combinación de casos sin limitación en el número máximo de quirófanos, $u = |J|$, lo cual puede ser debido al mayor número de posibles cirugías que puede asignar, dado que hay más cirujanos asignados al día.

De forma análoga, para el caso de limitación del número máximo de quirófanos, $u = 1$, ofrece mejores resultados, debido al motivo contrario. Además, puede ser observado que la programación que está realizando en dicho caso es muy favorable para el objetivo secundario, acerca del mínimo número de movimientos realizados por los cirujanos, el cual está consiguiendo un reparto equilibrado de sus jornadas, debido a la limitación impuesta en el número de quirófanos, así como el propio funcionamiento del algoritmo. Lo cual puede ser observado en que O2 toma valor 15 y 45, para 3 y 9 quirófanos aproximadamente, en la mayoría de los casos, lo cual quiere decir que los cirujanos entran a un quirófano, movimiento que registra y ya no se produce cambios de quirófanos para los mismos.

8 CONCLUSIONES

Adicional a las conclusiones realizadas en la etapa de pruebas computacionales, se pueden extraer otras, del trabajo realizado en su totalidad.

En primer lugar, y relativo a las fases iniciales sobre la elaboración del modelo ILP, y su adaptación a los algoritmos aproximados, se puede sacar como conclusión, que desarrollar ambos tipos, ayuda a una mejor compresión de los datos y variables implementados, así como la extracción de nuevas posibles decodificaciones como la relativa al aprovechamiento de intervalos libres.

Por otro lado, y más enfocado a la implementación ordenada de Reglas de Despacho, Heurísticas y por último Metaheurísticas; refleja en gran medida el aumento del grado de inteligencia de los algoritmos empleados, así como las soluciones que aportan.

Más en concreto y relativo a los algoritmos meméticos implementados, se ha verificado las afirmaciones de los mismos, sobre la repercusión del empleo de buenas búsquedas locales, apoyo, y causa, de gran mejora de la búsqueda global. Hecho que ha sido refutado con los datos obtenidos para las versiones implementadas en la evaluación computacional, basadas en modificaciones de las mismas, las cuales han mejorado de forma más significativa el algoritmo, el cual, de entrada, ya ha sido una metaheurística eficaz, debido a su reciente incorporación a la literatura.

A su vez, la implementación de un método sistemático de determinación de los mejores modelos y su posterior mejora, sumado a la implementación de varias políticas de gestión, no solo sirve para la obtención de soluciones de mayor calidad en base a los resultados obtenidos, sino también, a la mejor comprensión del funcionamiento de los mismos.

En definitiva, se ha realizado un estudio basado en la implementación y mejora de algoritmos meméticos para dos objetivos en conflicto, similar a los escenarios que se dan en la realidad, el cual además ha permitido la introducción a la optimización quirúrgica mediante un estudio profundo de la misma en las etapas iniciales, mediante la extracción y búsqueda de artículos profesionales y enfocados al problema tratado, así como un área de aplicación de métodos de optimización común de los niveles operativos en todo tipo de sectores, y la implementación de varias herramientas y su desarrollo durante todo el trabajo, como son la programación en lenguaje Python, documentación mediante Pseudocódigos, formulación matemática, etc.

En resumen, ha sido un proceso de investigación, implementación, desarrollo y aplicación de un problema complejo, el cual ha llevado asociado, el requerimiento de aprendizaje de otras herramientas, conceptos y metodologías muy ligadas al tipo de problema resuelto.

REFERENCIAS

Blake, J., & Donald, J. (2002). Mount Sinai Hospital Uses Integer Programming to Allocate Operating Room Time. *Interfaces, 32*, 63-73. Obtenido de https://www.researchgate.net/publication/228762348_Mount_Sinai_Hospital_Uses_Integer_Programming_to_Allocate_Operating_Room_Time

Cardoen, B., Demeulemeester, E., & Beliën, J. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research, 201*(3), 921-932. Obtenido de https://www.sciencedirect.com/science/article/pii/S0377221709002616

Chen, X., Ong, Y.-S., Lim, M.-H., & Tan, K. (2011). A Multi-Facet Survey on Memetic Computation. *IEEE Transactions on Evolutionary Computation, 15*(5), 591-607. Obtenido de https://ieeexplore.ieee.org/document/6036173

Dawkins, R. (2006). *The selfish gene* (30 ed.). New York, USA: Oxford University Press. Obtenido de https://alraziuni.edu.ye/uploads/pdf/The-Selfish-Gene-R.-Dawkins-1976-WW-.pdf

Fei, H., Meskens, N., & Chu, C. (2010). A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering, 58*(2), 221-230. Obtenido de https://www.sciencedirect.com/science/article/abs/pii/S0360835209000801

Framinan, J., Leisten, R., & Ruiz, R. (2013). *Manufacturing Scheduling Systems. An Integrated View on Models, Methods and Tools.* Springer. Obtenido de https://link.springer.com/book/10.1007/978-1-4471-6272-8

Hans, E., van Houdenhoven, M., & Hulshof, P. (2011). A Framework for Healthcare Planning and Control. En *Handbook of Healthcare System Scheduling* (Vol. 168, págs. 303-320). Obtenido de https://link.springer.com/chapter/10.1007/978-1-4614-1734-7_12

Hatami, S., Ruiz, R., & Andrés-Romano, C. (2015). Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *International Journal of Production Economics, 169*, 76-88. Obtenido de https://www.sciencedirect.com/science/article/pii/S0925527315002765

Huang, Y.-Y., Pan, Q.-K., & Gao, L. (2023). An effective memetic algorithm for the distributed flowshop scheduling problem with an assemble machine. *International Journal of Production Research, 61*(6), 1755-1770. doi:10.1080/00207543.2022.2047238

Ivo, A., & Vissers, J. (2002). Patient mix optimisation in hospital admission planning: A case study. *International Journal of Operations and Production Management, 22*(4), 445-461. Obtenido de https://www.scopus.com/record/display.uri?eid=2-s2.0-0036017717&origin=inward

Magerlein, J., & Martin, J. (1978). Surgical demand scheduling: a review. *Health Services Research, 13*(4), 418-433. Obtenido de https://www.scopus.com/record/display.uri?eid=2-s2.0-0018233732&origin=inward

Melian, B., Moreno-Pérez, J., & Moreno-Vega, J. (2003). Metaheuristics: A global view. *Inteligencia Artificial,revista Iberoamericana De Inteligencia Artificial - AEPIA, 7*. Obtenido de https://www.researchgate.net/publication/242934849_Metaheuristics_A_global_view

Molina-Pariente, J., Hans, E., Framinan, J., & Gomez-Cia, T. (2015). New heuristics for planning operating rooms. *Computers & Industrial Engineering, 90*, 429-443. Obtenido de https://www.sciencedirect.com/science/article/pii/S0360835215003952?via%3Dihub

Pinedo, M. (2005). *Planning and Scheduling in Manufacturing and Services.* Springer. Obtenido de https://link.springer.com/book/10.1007/978-1-4419-0910-7

Roland, B., Di Martinelly, C., Riane, F., & Pochet, Y. (2010). Scheduling an operating theatre under human resource constraints. *Computers & Industrial Engineering, 58*, 212-220. Obtenido de

https://www.sciencedirect.com/science/article/abs/pii/S0360835209000084