



**POLITÉCNICA**

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

**ASIGNACIÓN DE CONTROLADORES DE TRÁFICO AÉREO  
BASADO EN RECOCIDO SIMULADO MULTICOMIENZO Y  
EXPRESIONES REGULARES**

**TESIS DOCTORAL**

**FAUSTINO TELLO CABALLO**

**Graduado en Informática**

**2019**



Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos  
Departamento de Inteligencia Artificial

**ASIGNACIÓN DE CONTROLADORES DE TRÁFICO AÉREO  
BASADO EN RECODIDO SIMULADO MULTICOMIENZO Y  
EXPRESIONES REGULARES**

**Autor:** Faustino Tello Caballo  
Graduado en Informática

**Directores:** D. Alfonso Mateos Caballero  
Licenciado en CC. Matemáticas y Doctor en Informática  
D. Antonio Jiménez Martín  
Licenciado en Informática y Doctor en Informática

**Diciembre, 2019**



# Agradecimientos

---

En primer lugar, agradecer a los profesores Alfonso Mateos Caballero y Antonio Jiménez Martín las enseñanzas recibidas y el tiempo y esfuerzo dedicado durante todo el camino que me ha llevado hasta aquí, ya que sin ellos no hubiera sido posible.

También quiero dar gracias al resto de personas con las que he trabajado a lo largo de este tiempo por toda su ayuda y apoyo. En especial a Jonatan Lara por su trabajo y esfuerzo el cual me ha ayudado e inspirado enormemente.

Agradecer al equipo de CRIDA con el que he tenido el placer de trabajar durante todo este tiempo por todo su esfuerzo y el tiempo dedicado para que todo resultara lo mejor posible.

A toda mi familia, y en particular a mis padres, por su ayuda y paciencia a lo largo de toda mi etapa académica. Por último, y no menos importante, a mi pareja por su apoyo y comprensión.

El desarrollo de esta Tesis Doctoral ha sido posible gracias a la financiación de la Universidad Politécnica de Madrid, la beca del programa propio UPM-Santander Universidades y a los proyectos MTM2017-86875-C3-3-R y MTM2014-56949-C3-2-R financiados por el Ministerio de Economía y Competitividad.



# Resumen

---

La presente Tesis Doctoral se ha centrado en la resolución mediante el uso de metaheurísticas de un problema de optimización complejo en el ámbito de la gestión del tráfico aéreo en aeropuertos, la asignación de controladores aéreos a puestos de control para asegurar una correcta gestión del tráfico aéreo. Dicho problema se trata de un problema de optimización complejo combinatorio del tipo de asignación en el que debemos tener en cuenta los turnos de trabajo de los controladores, sus condiciones laborales (muchas de ellas establecidas mediante Real Decreto) y sus habilitaciones para poder gestionar distintos tipos de sectores y la sectorización establecida, y en el que deben considerarse múltiples objetivos.

Se han considerado tres variantes del problema. En la primera, se desea minimizar el número de controladores necesarios para cubrir cierta sectorización. En la segunda, el número de controladores está fijado y se considera un enfoque multiobjetivo en el que se establecen ciertas metas sobre las condiciones de trabajo de los controladores (restricciones no estrictas), donde se tiene en cuenta que la carga de trabajo sea homogénea entre los controladores, que el número de cambios de posiciones en sala sea el menor posible y que la estructura de la solución sea lo más parecida a las plantillas que actualmente son utilizadas (confeccionadas de forma manual).

Estas dos variantes se corresponden con la resolución del problema en la fase pre-táctica, la cual se ejecuta entre uno y seis días previos al día de operación a resolver. En la tercera variante del problema se considera la fase táctica, en sala. En este caso, se supone que se está ejecutando un horario determinado y que surge un imprevisto, como la baja de un controlador o la llegada de un elevado flujo de vuelos desviado de otro aeropuerto (por ejemplo, por problemas climatológicos). En este caso, la solución

propuesta tendrá que recalcular la mejor solución posible (puede que en algunos casos no quede más remedio que violar algunas de las condiciones de trabajo de los controladores, quedando una solución infactible) en un tiempo muy reducido, estableciéndose un orden de prioridad en la violación de las restricciones.

Para resolver las tres variantes del problema se han utilizado una metodología consistente en varias fases, en la que se utiliza una heurística para obtener una solución inicial y el uso alternativo de dos metaheurísticas, el *recocido simulado* y la *búsqueda en entornos variables*, para la búsqueda de soluciones factibles y/o soluciones óptimas, en función de la variante del problema. En los tres casos, se ha analizado el comportamiento de las metaheurísticas utilizadas en base a la calidad de las soluciones obtenidas y el tiempo de ejecución asociado a cada uno de ellos.

Para el desarrollo de la presente Tesis Doctoral se ha colaborado a través de la firma de varios convenios de colaboración con CRIDA (Centro de Referencia de Investigación, Desarrollo e Innovación ATM), que ha aportado el conocimiento experto sobre el problema.

# Abstract

---

This PhD. thesis deals with the resolution of a complex optimization problem in the field of air traffic management in airports by means of metaheuristics, the assignment of air traffic controller operators (ATCo) to control positions to ensure a proper air traffic management. This is a complex combinatorial optimization problem of the scheduling type in which we must take into account the ATCos work shifts, their labour conditions (many of them established by a Spanish Royal Decree) and their qualification to manage different sector types, and the established sectorization, and in which we must consider multiple conflicting objectives.

Three variants of the problem have been considered. In the first, the aim consists of minimizing the number of ATCos needed to cover a given sectorization. In the second, the number of ATCos is fixed and a multi-objective approach is considered in which certain goals are established on the ATCo's labour conditions (non-mandatory restrictions), considering that the workload is homogeneous among the ATCos, that the number of position changes in the control center is as small as possible and that the structure of the solution is as similar as possible to the hand-made template-based solutions that are currently being used.

These two variants correspond to the resolution of the problem in the pre-tactical phase, which takes place between one and six days before the day of operation to be solved. In the third variant of the problem, the tactical phase is considered in the control center. In this case, it is assumed that a certain schedule is being executed and an unforeseen event arises, such as the loss of an ATCo or the arrival of a high flow of flights diverted from another airport (for example, due to weather problems). In this case, we will have to recalculate the proposed solution to the best possible solution (in some cases there may

be no choice but to violate some of the ATCo's labour conditions, leading to an infeasible solution) in a short time, establishing an order of priority in the constraint violations.

To solve the three variants of the problem, a methodology consisting of several phases has been proposed and developed. A heuristic is used to derive an initial solution and the alternative use of two metaheuristics (*simulated annealing* and *variable neighbourhood search*) for the search of feasible solutions and/or optimal solutions is considered, depending on the variant of the problem. In the three cases, the behaviour of the metaheuristics used has been analysed on the basis of the quality of the solutions reached and the execution time associated to each of them.

During the development of the PhD. thesis we have collaborated by means of several collaboration agreements with CRIDA (Reference Center for Research, Development and Innovation ATM, Spain), which has provided expert knowledge on the problem.

# Índice general

---

<b>1. Introducción</b>	<b>7</b>
<b>2. Objetivos</b>	<b>13</b>
<b>3. Descripción del problema</b>	<b>15</b>
3.1. Primera aproximación al problema . . . . .	22
3.1.1. Restricciones . . . . .	23
3.1.2. Modelización matemática . . . . .	24
3.2. Versión completa del problema . . . . .	32
3.2.1. Restricciones . . . . .	33
3.3. Problema táctico . . . . .	34
<b>4. Problemas de asignación y extensiones</b>	<b>37</b>
4.1. Problemas de asignación o <i>scheduling</i> . . . . .	39
4.2. Problemas de horarios o <i>timetabling</i> . . . . .	41
4.3. Problemas de <i>timetabling</i> en el ámbito del control del tráfico aéreo . . . . .	44
<b>5. Técnicas/Herramientas utilizadas</b>	<b>47</b>
5.1. Metaheurísticas trayectoriales . . . . .	49
5.2. Recocido Simulado (SA) . . . . .	50
5.2.1. Recocido Simulado Uniobjetivo . . . . .	52
Generación de soluciones iniciales . . . . .	53
Función objetivo . . . . .	53
Temperatura inicial . . . . .	54
Función de enfriamiento . . . . .	55

Número de iteraciones en el que se mantiene constante la temperatura ( $L$ ) . . . . .	58
Soluciones del entorno y movimientos . . . . .	59
Función de aceptación . . . . .	60
Condición de parada . . . . .	61
5.2.2. Recocido Simulado Multiobjetivo . . . . .	62
Función de aceptación . . . . .	63
5.3. Búsqueda en Entornos Variables (VNS) . . . . .	65
5.3.1. Búsqueda en Entornos Variables Básica . . . . .	65
5.3.2. Búsqueda en Entornos Variables Descendente . . . . .	66
5.3.3. Búsqueda en Entornos Variables Reducida . . . . .	67
5.3.4. Búsqueda en Entornos Variables con Descomposición . . . . .	68
5.3.5. Búsqueda en Entornos Variables Sesgada . . . . .	68
5.3.6. Búsqueda en Entornos Variables Paralela . . . . .	69
5.4. Búsqueda Tabú . . . . .	70
5.5. Expresiones regulares . . . . .	72
<b>6. Metodología de solución propuesta</b>	<b>75</b>
6.1. Representación de la solución . . . . .	75
6.2. Primera aproximación al problema. Fase pre-táctica . . . . .	77
6.2.1. Fase 1: Creación de soluciones iniciales factibles . . . . .	77
Introducción de plantillas . . . . .	79
Reparación de soluciones . . . . .	83
Comprobación de la factibilidad . . . . .	85
6.2.2. Fase 2: SA multicomienzo para obtención de soluciones óptimas . .	86
Función objetivo . . . . .	87
Temperatura inicial, función de enfriamiento y número de iteraciones hasta el descenso de la temperatura . . . . .	87
Definición del entorno de una solución . . . . .	88
Condición de parada . . . . .	91
Extensión de la Fase 2: Obtención de soluciones balanceadas . . . .	91

6.2.3. Ejemplo de aplicación . . . . .	91
6.3. Versión completa del problema . . . . .	94
6.3.1. Fase 1: Creación de soluciones iniciales . . . . .	96
Introducción de plantillas . . . . .	96
Reparación de soluciones . . . . .	97
Asignación de los recursos disponibles . . . . .	97
6.3.2. Fase 2: SA multicomienzo para obtención de soluciones factibles . .	100
Algoritmo multicomienzo . . . . .	100
Función objetivo . . . . .	100
Temperatura inicial, función de enfriamiento y número de iteracio-	
nes hasta el descenso de la temperatura . . . . .	104
Definición del entorno de una solución . . . . .	107
Función de aceptación . . . . .	112
Condición de parada . . . . .	113
6.3.3. Fase 3: SA multicomienzo para obtención de soluciones óptimas . .	114
Función objetivo . . . . .	114
Temperatura inicial, función de enfriamiento y número de iteracio-	
nes hasta el descenso de la temperatura . . . . .	121
Definición del entorno de una solución . . . . .	122
6.3.4. Algoritmo alternativo: VNS . . . . .	123
Definición del entorno de una solución . . . . .	124
Búsqueda local multicomienzo . . . . .	127
Condición de parada . . . . .	128
6.3.5. Mejoras computacionales . . . . .	128
6.3.6. Ejemplo de aplicación: Recocido simulado . . . . .	130
6.3.7. Comparativa de los resultados de los algoritmos VNS y SA . . . .	140
6.4. Problema táctico . . . . .	143
6.4.1. Fase 1: Modificación de la solución actual . . . . .	144
6.4.2. Fase 2: SA para obtención de soluciones factibles y óptimas . . . .	146
Función objetivo . . . . .	147

Temperatura inicial, función de enfriamiento y número de iteraciones hasta el descenso de la temperatura . . . . .	148
Definición del entorno de una solución . . . . .	152
6.4.3. Ejemplos de aplicación . . . . .	153
<b>7. Desarrollo de la herramienta software</b>	<b>161</b>
7.1. Análisis de requisitos . . . . .	162
7.1.1. Análisis de requisitos funcionales . . . . .	163
7.1.2. Análisis de requisitos no funcionales . . . . .	165
7.2. Diseño y arquitectura . . . . .	166
7.3. Implementación . . . . .	168
7.4. Pruebas de software . . . . .	169
7.5. Documentación y mantenimiento . . . . .	171
<b>8. Conclusiones y líneas futuras de investigación</b>	<b>175</b>
<b>A. Anexo 1: Expresiones Regulares</b>	<b>181</b>

# Índice de figuras

---

1.1. Evolución del tráfico aéreo . . . . .	7
3.1. FIR pertenecientes al territorio español . . . . .	17
3.2. TMA pertenecientes al territorio español . . . . .	18
3.3. CTA pertenecientes al territorio español . . . . .	18
3.4. Algunos sectores pertenecientes al FIR de Madrid . . . . .	19
3.5. Desdoblamiento del sector R1L . . . . .	20
3.6. Turnos de trabajo del centro de control de Barcelona . . . . .	21
3.7. Ejemplo de sectorización . . . . .	21
3.8. Ejemplo de sectorización . . . . .	31
4.1. Clasificación de los tipos de problemas de asignación . . . . .	38
5.1. Ejemplo de frente de Pareto . . . . .	62
6.1. Matriz de turnos . . . . .	77
6.2. Plantilla 3x1 de referencia para la creación de soluciones iniciales . . . . .	78
6.3. Plantilla de referencia para L=6 . . . . .	79
6.4. Ejemplo de sectorización . . . . .	80
6.5. Explicación del algoritmo de inicialización 1/4 . . . . .	81
6.6. Explicación del algoritmo de inicialización 2/4 . . . . .	81
6.7. Explicación del algoritmo de inicialización 3/4 . . . . .	82
6.8. Explicación del algoritmo de inicialización 4/4 . . . . .	82
6.9. Traspaso de fragmento de trabajo con unión anterior . . . . .	84
6.10. Traspaso de fragmento de trabajo con unión posterior . . . . .	84

6.11.	Adquisición de fragmento de trabajo con unión anterior . . . . .	85
6.12.	Adquisición de fragmento de trabajo con unión posterior . . . . .	85
6.13.	Solución resultante de la aplicación de la segunda fase . . . . .	88
6.14.	Ejemplo de evolución del porcentaje de aceptación . . . . .	89
6.15.	Sectorización del turno . . . . .	92
6.16.	Solución inicial . . . . .	93
6.17.	Solución óptima . . . . .	94
6.18.	Solución óptima balanceada . . . . .	95
6.19.	Solución inicial . . . . .	110
6.20.	Diagrama de flujo del algoritmo VNS . . . . .	124
6.21.	Ejemplo del <i>Entorno 1</i> . . . . .	125
6.22.	Ejemplo del <i>Entorno 2</i> . . . . .	125
6.23.	Ejemplo del <i>Entorno 3</i> . . . . .	126
6.24.	Ejemplo del <i>Entorno 4</i> . . . . .	126
6.25.	Comparativa de tiempos entre el uso de Regex y código . . . . .	129
6.26.	Comparativa de tiempos entre el uso de Regex y código . . . . .	130
6.27.	Sectorización del Caso 7 . . . . .	130
6.28.	Solución inicial del Caso 7 . . . . .	132
6.29.	Solución factible del Caso 7 . . . . .	133
6.30.	Solución óptima del Caso 7 . . . . .	134
6.31.	Sectorización del Caso 13 . . . . .	135
6.32.	Solución incial del Caso 13 . . . . .	136
6.33.	Solución factible del Caso 13 . . . . .	136
6.34.	Solución óptima del Caso 13 . . . . .	136
6.35.	Sectorización del Caso 23 . . . . .	136
6.36.	Solución inicial del Caso 23 . . . . .	137
6.37.	Solución factible del Caso 23 . . . . .	137
6.38.	Solución óptima del Caso 23 . . . . .	137
6.39.	Sectorización del Caso 29 . . . . .	138
6.40.	Solución inicial del Caso 29 . . . . .	138
6.41.	Solución factible del Caso 29 . . . . .	138

6.42. Solución óptima del Caso 29 . . . . .	138
6.43. Sectorización del Caso 65 . . . . .	139
6.44. Solución inicial del Caso 65 . . . . .	139
6.45. Solución factible del Caso 65 . . . . .	139
6.46. Solución óptima del Caso 65 . . . . .	140
6.47. Ejemplo de slot de cambio . . . . .	143
6.48. Sectorización del Caso 1 . . . . .	153
6.49. Solución inicial del Caso 1 . . . . .	154
6.50. Solución óptima del Caso 1 . . . . .	154
6.51. Sectorización del Caso 4 y 5 . . . . .	155
6.52. Solución inicial del Caso 4 . . . . .	156
6.53. Solución óptima del Caso 4 . . . . .	156
6.54. Solución inicial del Caso 5 . . . . .	157
6.55. Solución óptima del Caso 5 . . . . .	157
6.56. Sectorización del Caso 8 y 9 . . . . .	157
6.57. Solución inicial del Caso 8 . . . . .	158
6.58. Solución óptima del Caso 8 . . . . .	158
6.59. Solución inicial del Caso 9 . . . . .	158
6.60. Solución óptima del Caso 9 . . . . .	159
 7.1. Modelo especificativo del sistema . . . . .	167
7.2. Modelo de proceso de negocio . . . . .	172
7.3. Estructura del proyecto . . . . .	173
7.4. Ejemplo de la documentación generada sobre una clase . . . . .	173
7.5. Ejemplo de la documentación generada para un constructor . . . . .	174
7.6. Ejemplo de la documentación generada para un método . . . . .	174



# Índice de tablas

---

5.1. Comparativa de las distintas funciones de enfriamiento . . . . .	59
6.1. Resultados del algoritmo de creación de soluciones . . . . .	86
6.2. Instancias más representativas del problema . . . . .	104
6.3. Resultados del ajuste de la temperatura inicial para la segunda fase . . . .	105
6.4. Resultados del ajuste del número de iteraciones y función de enfriamiento para la segunda fase . . . . .	105
6.5. Resultados del estudio de los distintos entornos para la segunda fase . . . .	112
6.6. Resultados del ajuste de la temperatura inicial para la tercera fase . . . .	121
6.7. Resultados del ajuste del número de iteraciones y función de enfriamiento para la tercera fase . . . . .	122
6.8. Resultados del estudio de los distintos entornos para la tercera fase . . . .	122
6.9. Resultados de la aplicación de la metodología de resolución sobre los Casos 7, 13, 23, 29 y 65. . . . .	134
6.10. Resultados explicativos de la aplicación de la metodología de resolución sobre los Casos 7, 13, 23, 29 y 65. . . . .	135
6.11. Comparativa de los resultados obtenidos aplicando los algoritmos VNS y SA. . . . .	141
6.12. Comparación de los resultados explicativos . . . . .	142
6.13. Instancias más representativas del problema. . . . .	149
6.14. Resultados del ajuste de la temperatura inicial para el problema táctico . .	149
6.15. Resultados del ajuste del número de iteraciones y función de enfriamiento para el problema táctico . . . . .	149
6.16. Resultados del estudio de los distintos entornos en el problema táctico . .	152

CAPÍTULO 0

---

# Capítulo 1

## INTRODUCCIÓN

---

La actividad principal de los controladores aéreos es gestionar el tráfico aéreo garantizando la seguridad del espacio aéreo. Para asegurarse que esta tarea se cumple, deben detectar y resolver posibles conflictos entre trayectorias de vuelo de las aeronaves.

Esta tarea cada vez es más compleja debido al drástico aumento del tráfico aéreo a lo largo de las últimas décadas, como podemos observar en la Figura 1.1.

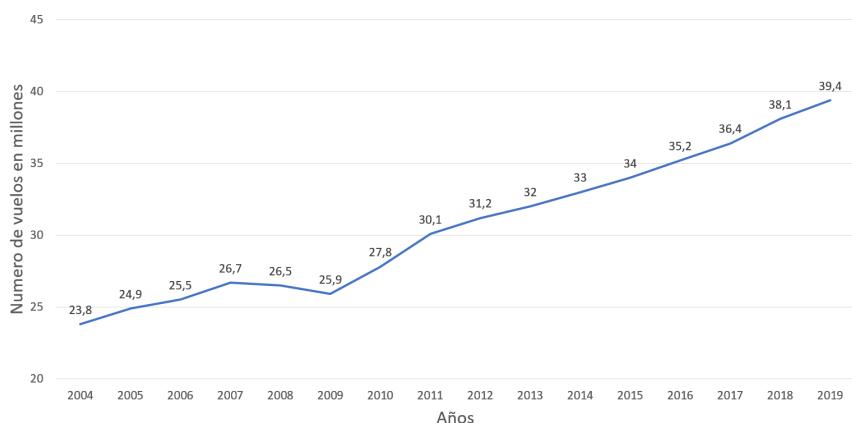


Figura 1.1: Evolución del tráfico aéreo

La gestión del tráfico aéreo (*Air Traffic Management*, ATM) es un aspecto fundamental. Según la Organización de Aviación Civil Internacional la ATM consta de las siguientes áreas: gestión del espacio aéreo, control del tráfico aéreo (*Air Traffic Control*, ATC), y gestión de afluencia de tránsito aéreo y capacidad.

Existen un gran número de problemas a resolver en las distintas áreas ATM. Debido al aumento del tráfico, cada vez es más habitual ver retrasos en los vuelos, aeronaves desviadas, o realizar más kilómetros y horas de vuelo que las necesarias porque las rutas se encuentran saturadas. Todos estos problemas deben resolverse sin olvidarnos de cumplir

con todas las medidas de seguridad. Tampoco podemos olvidarnos de las saturaciones que sufren los aeropuertos en determinadas fechas, debido a que el número de pistas de despegue y aterrizaje de los aeropuertos son limitadas y, en muchas ocasiones, generan retrasos.

Una de las áreas de mayor importancia para un correcto funcionamiento del espacio aéreo, es el área ATC. Los servicios que se incluyen dentro de este área son: el servicio de control de área o ruta, el servicio de control de aproximación y el servicio de control de aeródromo.

Todos estos servicios son realizados por controladores aéreos. En concreto, nos centramos en los servicios de control de ruta y de aproximación. Los servicios de control de aproximación asisten a las aeronaves que salen y llegan de los aeropuertos. En las salidas, transfieren la información al servicio de control de área para que se haga cargo de la asistencia a la aeronave, mientras que en las llegadas, lo transfieren al servicio de control de aeródromo.

En el caso de los servicios de control de ruta, gestionan los vuelos en las áreas de control terminal, en áreas de control o en aerovías.

Estos servicios deben ofrecerse durante las veinticuatro horas del día y, para hacer esto posible, los controladores se encuentran distribuidos en diferentes turnos de trabajo, debido a que el espacio aéreo debe estar controlado en todo momento.

El problema a resolver en esta Tesis Doctoral es determinar la asignación de los controladores aéreos a los distintos puestos de trabajo de cada turno, cumpliendo con todas las restricciones laborales de los controladores, muchas de las cuales son impuestas por ley en un Real Decreto.

Los controladores solo son capaces de gestionar una determinada cantidad de tráfico, por lo que el espacio aéreo se divide en sectores. Cada uno de estos sectores debe ser cubierto por dos controladores (controlador ejecutivo y controlador planificador). Al conjunto de sectores operados en cada período de tiempo a lo largo de un turno, se le denomina *sectorización*.

Dentro de este ámbito hemos resuelto tres tipos de problemas:

- El primero es la creación de un horario de trabajo para los controladores aéreos que cumpla con las restricciones laborales y cubra la sectorización completa a lo largo

---

del turno para la correcta gestión del espacio aéreo. La creación de este horario debe realizarse minimizando el número de controladores utilizados en el turno de trabajo y equilibrando la carga de trabajo entre los mismos. Este problema se encuentra descrito y resuelto en [99].

- El segundo problema consiste en la creación de un horario de trabajo cumpliendo las restricciones laborales de los controladores y cubriendo la sectorización del turno. La primera diferencia con el problema anterior reside en el conjunto de restricciones, el cual es más complejo y completo. En este problema, tenemos un número de controladores fijo, el cual no debemos minimizar. En cambio, tenemos un conjunto de objetivos deseables que deberemos optimizar, como son, el equilibrio de la carga de trabajo entre los controladores, tiempos óptimos de trabajo y descanso continuos, maximizar el paso por sectores que influyen para mantener las acreditaciones de trabajo, etc. Este problema se encuentra descrito y resuelto en [100, 98].

Cabe destacar que ambos problemas se resuelven durante la fase pre-táctica, la cual se realiza entre siete días y un día antes del día de operación, por lo que el tiempo de resolución no es un factor decisivo.

- En el tercer problema resuelto partimos con un horario de trabajo ya creado, y debemos solucionar un imprevisto que surja en el mismo turno en el que se esté aplicando el horario, durante la fase táctica. Por ello, se debe modificar parte del horario para poder adaptarlo y ofrecer una solución para el imprevisto, teniendo en cuenta el trabajo realizado anteriormente por los controladores.

Podemos contemplar imprevistos tales como la baja de un controlador, o la apertura y cierre de sectores por cambios en el tráfico aéreo estimado.

Uno de los principales inconvenientes es el escaso tiempo del que se dispone para resolver el problema, al cual añadimos la dificultad de tener en cuenta trabajo ya realizado por los controladores a la hora de cumplir las restricciones de trabajo.

En función de los imprevistos surgidos, se contempla la imposibilidad de cumplir con todas las restricciones laborales, por lo que se asignan pesos a las restricciones en función de su importancia. También se ha considerado un enfoque multiobjetivo

para reducir el impacto en la sala de control de los cambios realizados en el horario, intentando realizar los cambios de la manera más progresiva y pausada posible, evitando producir momentos de desconcierto o estrés.

Es importante destacar que en el desarrollo de la presente Tesis Doctoral se ha contado con la colaboración de expertos del Centro de Referencia de Investigación, Desarrollo e Innovación ATM A.I.E. (CRIDA), con el que se han firmado varios convenidos de colaboración que han dado lugar al desarrollo de varios proyectos de colaboración de CRIDA con la UPM, en los que ha participado tanto el autor de esta Tesis Doctoral, como sus tutores. Los expertos de CRIDA han proporcionado información sobre el dominio del problema, restricciones laborales de los controladores, objetivos a optimizar y sus preferencias sobre los mismos, e instancias a resolver sobre los tres tipos de problemas considerados.

Los proyectos desarrollados en colaboración y financiados por CRIDA son:

- “Investigación y desarrollo de una metaheurística de solución al problema ABACO”, Noviembre 2016 – Noviembre 2017. Proyecto relacionado con los dos primeros problemas considerados.
- “Evolución de la metaheurística de solución al problema ABACO y su aplicación al proyecto AIRPORTS”, Diciembre 2018 – Noviembre 2019. Proyecto relacionado con el tercero de los tres problemas resueltos.

Los tres tipos de problemas son problemas de optimización, los cuales pertenecen al conjunto de problemas de asignación [95, 96] u horarios (*timetabling*) [32, 12]. El tamaño y complejidad de este tipo de problemas combinatorios dificultan o, incluso, hacen imposible su resolución mediante métodos exactos. Por esto, para su resolución se han estudiado posibles alternativas y se han elegido algunas de las metaheurísticas más prometedoras en la resolución de este tipo de problemas.

Entre las metaheurísticas estudiadas y utilizadas para la resolución del problema se encuentran el *recocido simulado* [67, 16], la *búsqueda en entornos variables* [75] y la *búsqueda tabú* [42]. También se han aplicado heurísticas y algoritmos para agilizar algunos procesos y construir las soluciones iniciales de las metaheurísticas.

Una vez resueltos los problemas se ha tratado de analizar y mejorar las metodologías de resolución tanto en tiempo de ejecución, como en la calidad de los resultados ofrecidos.

---

Para finalizar el capítulo, procedemos a exponer la estructura de la presente Tesis Doctoral. En el Capítulo 2 se explican y enumeran los distintos objetivos a cumplir. En el Capítulo 3 se expone el problema a resolver en detalle, se explica cómo se realiza la representación de los datos, y también se describe una primera aproximación al problema realizada para facilitar el entendimiento del mismo y su resolución. Por último, se describe el problema completo junto con sus objetivos y restricciones.

En el Capítulo 4, se muestran los distintos tipos de problemas de *timetabling* que podemos encontrar y los distintos sistemas utilizados para la clasificación de estos. El capítulo finaliza con una pequeña sección sobre algunos de los problemas de asignación de trabajo a controladores aéreos y los distintos métodos utilizados para su resolución.

En el Capítulo 5, se exponen los métodos elegidos para la resolución del problema. El capítulo cuenta con una introducción a las metaheurísticas trayectoriales y, posteriormente, se definen metaheurísticas como el *recocido simulado* en su versión uniobjetivo y multiobjetivo, en el que profundizaremos en las variantes existentes y los distintos parámetros del algoritmo. También se expondrá la *búsqueda en entornos variables* junto con algunas de sus variantes. Por último, mostraremos el algoritmo de *búsqueda tabú* exponiendo alguna de las opciones de diseño. Una vez finalizada la descripción de las metaheurísticas, se explicará una de las herramientas de reconocimiento de patrones llamada expresiones regulares.

En el Capítulo 6, se muestran los resultados obtenidos con los distintos métodos de resolución aplicados para cada uno de los tres problemas considerados, y un análisis detallado de los mismos, incluyendo el motivo de su elección.

En el Capítulo 7, se explica brevemente los detalles técnicos y prácticos que han permitido la resolución del problema, ya sea código implementado, librerías utilizadas, etc. Además, se expone un ejemplo de aplicación de cada uno de los tres problemas considerados para facilitar la comprensión de la herramienta.

Por último, en el Capítulo 8, se realiza una breve exposición de las conclusiones derivadas del trabajo y un estudio de las posibles líneas futuras de investigación que pueden surgir a partir de este trabajo.



---

## Capítulo 2

# OBJETIVOS

---

El objetivo fundamental de esta Tesis Doctoral consiste en la resolución de tres problemas diferentes de asignación de los controladores aéreos a los distintos puestos de trabajo de cada turno, cumpliendo con todas las restricciones laborales de los controladores y con la sectorización establecida. Todo esto, añadiendo la optimización de un conjunto de objetivos, los cuales han sido establecidos por expertos en la materia, pertenecientes a CRIDA.

Para poder alcanzar una resolución positiva del problema y cumplir el objetivo principal, se han identificado los siguientes sub-objetivos:

1. Realizar un estudio detallado del problema y las restricciones que atañen al mismo.
2. Definir y describir aquellos elementos pertenecientes al dominio del problema.
3. Realizar una modelización matemática del problema para facilitar el entendimiento del mismo, así como definir la solución y las restricciones de manera precisa.
4. Investigar el estado del arte de los problemas de asignación y de horarios (*timetabling*), así como de los distintos métodos usados para su resolución.
5. Proponer posibles métodos de resolución aplicables al problema que permitan cumplir con los requisitos.
6. Estudiar el funcionamiento de las expresiones regulares y codificar todas las restricciones que lo permitan, como patrones para el uso de expresiones regulares. Además de realizar un estudio sobre sus ventajas e inconvenientes.

7. Realizar un estudio para definir los valores parametrizables del problema, cuáles son los datos de entrada y las posibles variaciones que puedan tener.
8. Diseñar una metodología de resolución aplicable a cualquier instancia del problema para cada uno de los tres tipos de problemas resueltos.
9. Aplicar los métodos de resolución propuestos a los distintos tipos de problemas que podemos encontrar dentro del ámbito de la asignación de trabajo a controladores aéreos.

Dado que la presente Tesis Doctoral supone el desarrollo e implementación de distintas herramientas informáticas para resolver varias instancias de problemas de asignación de controladores, se indican a continuación una serie de sub-objetivos asociados a las etapas típicas de la Ingeniería del Software para el desarrollo de proyectos informáticos:

10. Realizar un informe sobre el análisis de requisitos de una herramienta para su posterior diseño e implementación.
11. Diseño e implementación de la herramienta software aplicable para la resolución satisfactoria de cualquier instancia del problema.
12. Diseñar e implementar en un formato sencillo y comprensible una herramienta para la entrega de las soluciones obtenidas, junto con su evaluación y el resto de información relevante.
13. Diseño y ejecución de las pruebas para probar el correcto funcionamiento de la herramienta.

Finalmente, se incorporan los siguientes sub-objetivos asociados a los resultados alcanzados y las futuras líneas de investigación:

14. Análisis de los resultados y propuesta de posibles mejoras, con el objetivo de mejorar las soluciones obtenidas y su rapidez.
15. Exponer las conclusiones obtenidas y proponer futuras líneas de investigación.

---

## Capítulo 3

# Descripción del problema

---

La gestión del tráfico aéreo (*Air Traffic Management*, ATM), engloba un conjunto de servicios, y todos ellos relacionados con el tráfico aéreo. Dentro de estos servicios encontramos [33]:

1. Gestión del espacio aéreo. Entre las actividades relacionadas con este servicio podemos destacar las siguientes: diseñar planes de eficiencia, continuidad de servicio y optimización del espacio aéreo que definen medidas de mejora de su estructura, junto con la creación y optimización de rutas aéreas.
2. Servicio de tránsito aéreo (*Air Traffic Services*, ATS). Dentro de este área, podemos distinguir tres tipos de servicios: servicios de control de tráfico aéreo (*Air Traffic Control*, ATC), servicios de información de vuelo y servicios de alerta.
3. Gestión de afluencia y capacidad (*Air Traffic Flow and Capacity Management*, ATFCM). El principal objetivo de este área es contribuir a que el tráfico de las aeronaves sea lo más fluido posible y aprovechar al máximo la capacidad del espacio aéreo y de los aeropuertos. Desde este área también se coordinan con el *Network Manager* para minimizar el impacto de posibles cambios debido a condiciones meteorológicas adversas, aumento del tráfico inesperado o cierre de pistas.
4. Sistemas de gestión del tránsito aéreo (*Air Traffic Management systems*, ATM). Muy ligados a los servicios del área ATS encontramos los servicios ATM. Estos servicios son utilizados directamente por los controladores aéreos. La gama de servicios es extensa y, entre otros, encontramos servicios como facilitar la información de planes de vuelo, meteorológica, de radares para el seguimiento de aeronaves, etc.

Los problemas considerados en esta Tesis Doctoral se centrarán en los servicios de control de tráfico aéreo.

El servicio ATFCM se desarrolla en cuatro fases [80]:

1. La *fase estratégica*, debe realizarse siete días antes del día de operación. Durante esta fase, el *Network Manager Operations Centre* (NMOC) ayuda a los proveedores de servicios de navegación aérea a estimar la capacidad que se necesitará en cada uno de los centros de control para así elaborar un horario de trabajo en función de las previsiones realizadas.
2. La *fase pre-táctica* tiene entre uno y seis días de plazo previo al día de operación. El NMOC coordina la definición del plan del día e informa a los controladores aéreos y a los operadores de aeronaves sobre las medidas ATFCM que se establecerán en el espacio aéreo europeo en el día de operación.
3. La *fase táctica* tiene lugar el mismo día de operación. El NMOC revisa y actualiza el plan del día en base a la situación actual y la continua capacidad de optimización acorde al tráfico aéreo en tiempo real. Cuando una aeronave se ve afectada por una regulación, el centro ofrece soluciones alternativas para minimizar los posibles retrasos.
4. Por último, la *fase post-táctica* tiene lugar después del día de operación. Se lleva a cabo un informe para analizar, investigar e informar sobre el proceso y las actividades realizadas en todos los ámbitos y por todas las entidades relevantes al servicio ATFCM. El objetivo de esta fase es implementar mejores metodologías y mejorar los procesos y actividades operacionales.

Durante la fase pre-táctica se elabora un plan de día ATFCM o lo más detallado posible. Es en esta fase cuando el equipo de gestión de afluencia distribuye información al *Network Manager* (NM) para la elaboración del plan pre-táctico.

La seguridad de los aeropuertos no depende únicamente de los controladores. Cuando se pronostica que la demanda de tráfico de un espacio aéreo, o de un aeródromo, supera la capacidad de los controladores disponibles, se aplican un conjunto de medidas por el NM

---

en colaboración con el equipo de gestión de afluencia para regular el volumen de tráfico y garantizar que la seguridad no se vea comprometida en ningún momento.

Entre la información que recibe, se encuentra la activación y configuración de sectores, motorización de valores, volumen del tráfico aéreo, detalles de eventos o información que pueda afectar a la capacidad de los centros de control o aeródromos, etc.

Por otra parte, existen los servicios de tránsito aéreo, dentro de los cuales se incluyen los servicios de control de tráfico aéreo.

El mayor proveedor de servicios ATC en España es ENAIRE, la cual dispone de cinco centros de control (ACC): Barcelona, Madrid, Sevilla, Palma de Mallorca y Gran Canaria. El servicio es prestado directamente por los controladores y su principal tarea es gestionar el tráfico aéreo garantizando la seguridad del espacio aéreo en todo momento. Para ello, deben dirigir el tráfico, autorizar las peticiones de los pilotos, aplicar la separación necesaria entre aeronaves, etc.

Para hacer esto posible, la Organización de Aviación Civil Internacional segmenta el mundo en nueve regiones de información de vuelo (*Flight Information Region, FIR*). Cada una de estas, es subdividida en otras tantas FIR.

Cada país es responsable del servicio de las regiones comprendidas dentro de su área de responsabilidad. Las FIR de España son FIR Madrid, FIR Barcelona y FIR Canarias. En la Figura 3.1 podemos ver las FIR del territorio español.

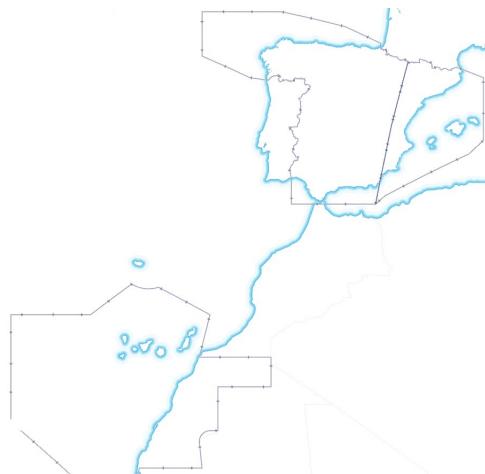


Figura 3.1: FIR pertenecientes al territorio español

En función de la altitud, el FIR tiene dos divisiones; la región inferior o FIR y la región superior o UIR. El FIR se extiende desde el suelo hasta los 24.500 pies y el UIR desde los

24.500 hasta los 46.000 pies.

Dentro de los FIR, encontramos las *áreas de control terminal* o (*Terminal Manoeuvring Area*, TMA), las cuales son puntos del espacio aéreo donde confluyen aerovías próximas a uno o más aeropuertos y se enlaza la fase de vuelo de aproximación y ruta. En España contamos con doce TMA, las cuales se muestran en la Figura 3.2.

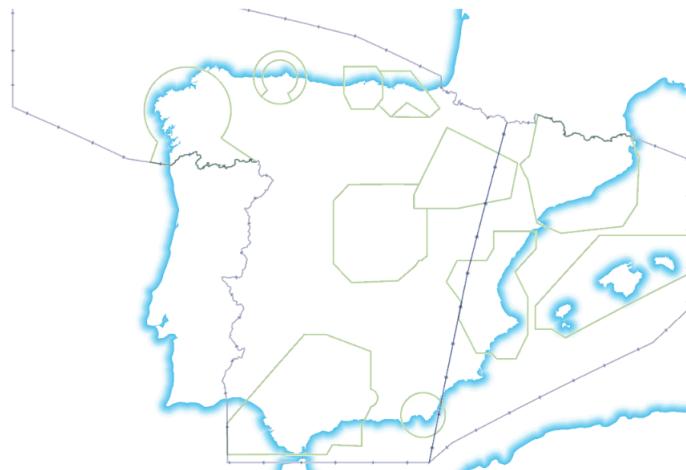


Figura 3.2: TMA pertenecientes al territorio español

En el espacio aéreo también podemos encontrar las *Áreas de Control* o (*Control Area*, CTA). Estas son zonas del espacio aéreo que engloban las proximidades de la mayoría de aeropuertos. En España disponemos de siete CTA, las cuales pueden verse en la Figura 3.3.

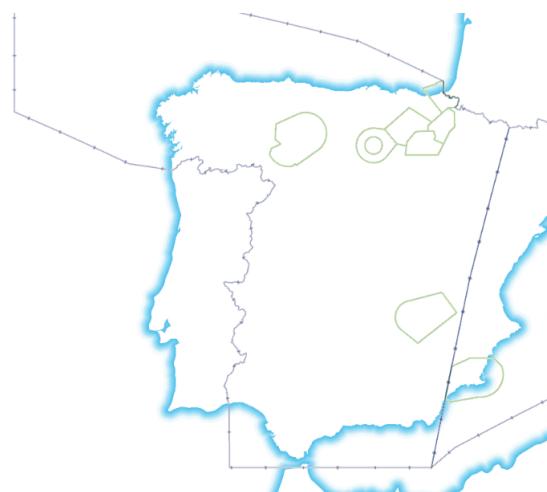


Figura 3.3: CTA pertenecientes al territorio español

A su vez, cada uno de los FIR tienen divisiones para hacer posible el control del espacio

---

aéreo y cada una de las divisiones se llama *núcleo*. Estos núcleos se subdividen en espacios más pequeños, y estos espacios obtenidos son denominados *sectores*. Los sectores son las divisiones lógicas que se realizan sobre el espacio aéreo para facilitar el control del mismo. Los sectores nombrados anteriormente, están formados por agrupaciones de uno o más *volúmenes*, los cuales son la unidad elemental del espacio aéreo. En cualquier instante de tiempo, los sectores abiertos deben agrupar todos los volúmenes del espacio aéreo.

En la Figura 3.4 tenemos un ejemplo de algunos de los sectores pertenecientes al FIR de Madrid.

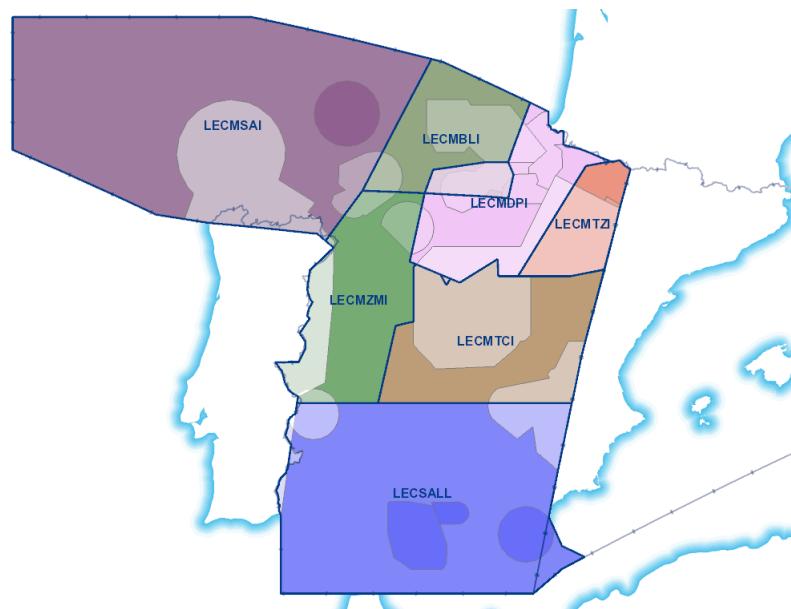


Figura 3.4: Algunos sectores pertenecientes al FIR de Madrid

Durante un turno de trabajo se pueden abrir o cerrar sectores, lo que se conoce como *desdoblamiento*. Por ejemplo, el sector R1L del FIR de Madrid (LECM) se puede desdoblar en SAI y BDP, como vemos en la Figura 3.5.

Por otro lado, los núcleos anteriormente citados son conjuntos de sectores. Pueden existir sectores que pertenezcan simultáneamente a dos núcleos, mientras que los controladores se habilitan para operar en un único núcleo.

Dos sectores se denominan *afines* si poseen algún volumen en común. Como es lógico, no puede darse el caso de que existan dos sectores abiertos simultáneamente con volúmenes comunes ya que, cuando los sectores se encuentran operativos, no pueden tener intersecciones.

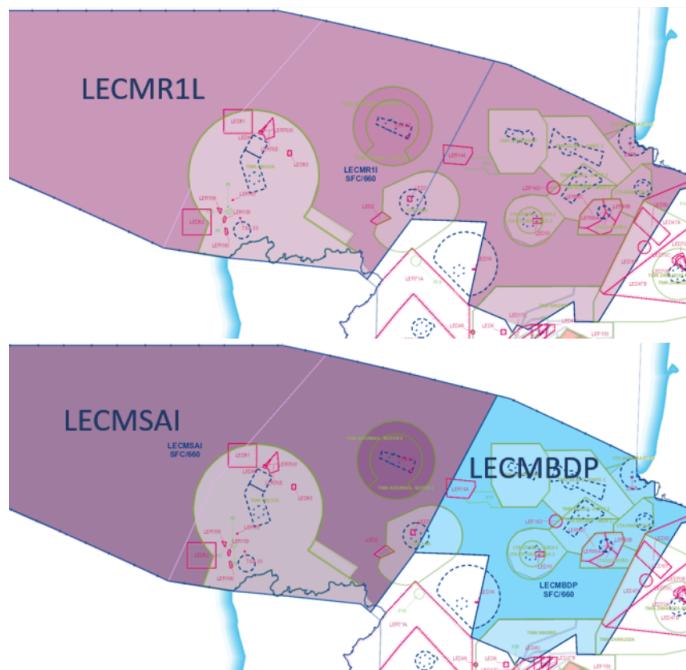


Figura 3.5: Desdoblamiento del sector R1L

Los sectores tienen asignadas dos *posiciones de control*. Cuando un sector se encuentra operativo las dos posiciones de control deben ser cubiertas por un controlador cada una. Tendremos un controlador *ejecutivo*, que se encarga de gestionar el tráfico aéreo comunicando a los pilotos las instrucciones pertinentes para evitar posibles situaciones de conflicto entre las aeronaves dentro del sector.

El controlador *planificador* es el responsable de anticiparse a las posibles situaciones de conflicto y comunicárselas al controlador ejecutivo para que este resuelva la situación. Principalmente, actúan en los límites del espacio aéreo del sector asignado, vigilando las entradas de aeronaves de los sectores colindantes.

Los sectores, además de pertenecer a uno o varios núcleos, también se dividen por tipos. Existen los sectores de *aproximación* y los sectores de *ruta*. Los sectores de aproximación se encuentran cercanos a centros de control, por lo que exigen una mayor atención y nivel de conocimientos para la gestión del tráfico aéreo en estos. Por otro lado, los sectores de ruta se encuentran más alejados de los centros de control, se suelen realizar menos maniobras y, por tanto, se gestionan más fácilmente.

Al igual que en otros trabajos, los controladores tienen turnos. En concreto, tienen tres tipos de turno: turno de mañana, tarde y noche. Los turnos de mañana y tarde tienen

dos variantes: los turnos cortos y los turnos largos. Estos últimos se solapan con el turno de noche, aumentando ligeramente la duración del turno. Cada centro de control regula la hora de inicio y fin de sus turnos. En la Figura 3.6 podemos ver el caso del centro de control de Barcelona.

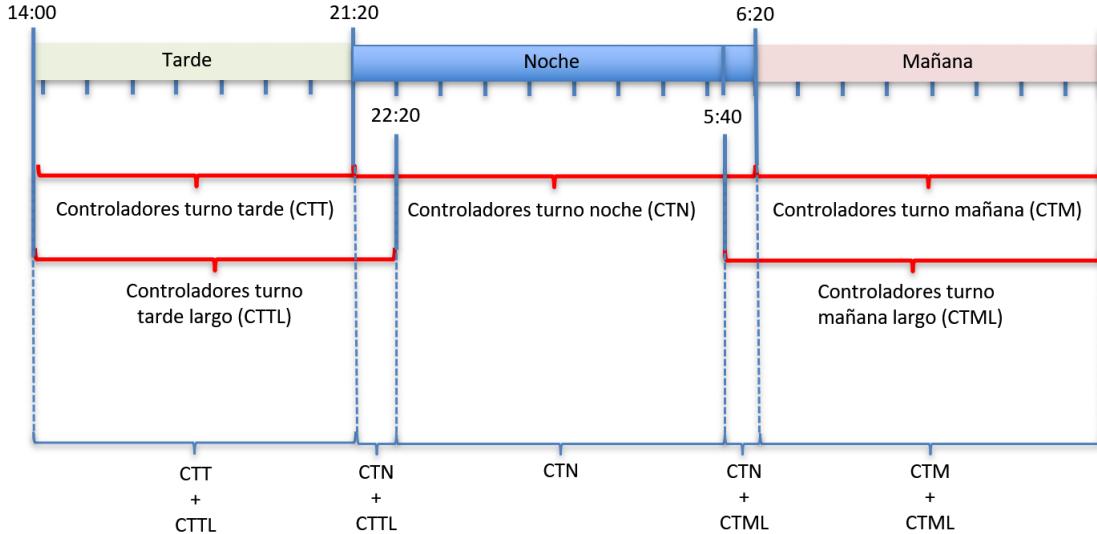


Figura 3.6: Turnos de trabajo del centro de control de Barcelona

Otro término que debemos explicar es el de *sectorización*. Denominamos sectorización al conjunto de sectores operados en cada período de tiempo a lo largo de un turno, la cual nos indica la lista de los sectores abiertos y los intervalos de tiempo que estos permanecen abiertos durante el turno. La sectorización de un turno para un aeropuerto depende del tráfico aéreo en el mismo. La sectorización para cada turno se realiza mediante una estimación del tráfico aéreo con veinticuatro horas de antelación. Encontramos un ejemplo de una sectorización en la Figura 3.7.

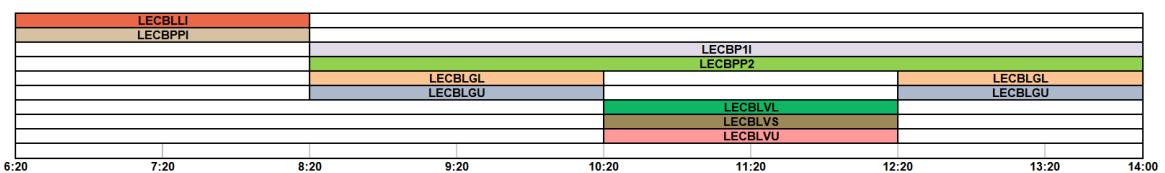


Figura 3.7: Ejemplo de sectorización

Esta sectorización corresponde a un turno de mañana del centro de control de Barcelona: El turno abarca desde las 6:20 hasta las 14:00 y durante este turno se involucran nueve sectores distintos. Desde las 6:20 a las 8:20 encontramos abiertos dos secto-

res (LECBLLI, LECBPPI), de 8:20 a 10:20 contamos con cuatro sectores (LECBLGL, LECBLGU, LECBPP2, LECBP1I), de 10:20 a 12:20 tenemos cinco sectores abiertos (LECBLVL, LECBLVS, LECBLVU; además de los sectores LECBPP2, LECBP1I que permanecen abiertos) y de 12:20 a 14:00 encontramos cuatro sectores (volviendo a la configuración abierta de 8:20 a 10:20 con los sectores: LECBLGL, LECBLGU, LECBPP2, LECBP1I).

Los controladores aéreos (*Air Traffic Controllers*, ATCos) son los encargados de operar en los distintos sectores. Cada controlador estará asignado a un único núcleo de sectores y únicamente podrá ser asignado a los sectores pertenecientes al núcleo. Cada uno tendrá una capacitación que podrá ser PTD (Ruta y Aproximación) o CON (Ruta). Esta capacitación permitirá a los controladores estar asignados a unos sectores u otros.

El objetivo de este problema es asignar a los controladores turnos de trabajo que cumplan con sus restricciones laborales y les permita cubrir todos los sectores abiertos del espacio aéreo en todo momento. Este tipo de problemas son conocidos como problemas de *timetabling*, que es un caso especial de los problemas de asignación.

A continuación, explicaremos los tres tipos distintos de problemas de *timetabling* relacionados con la asignación de controladores que se han resuelto en esta Tesis Doctoral.

Es importante recordar que en los tres problemas considerados se ha contado con la colaboración de expertos de CRIDA, que nos ha proporcionado la información experta requerida y todos los datos necesarios para la perfecta comprensión del problema, así como de las instancias resueltas de cada uno de los tres problemas.

### 3.1. Primera aproximación al problema

A continuación vamos a explicar una primera aproximación al problema. El motivo de que no se resolviera el problema completo desde el inicio es la elevada complejidad del mismo. Las numerosas y complejas restricciones, además del tamaño del problema, han sido factores importantes. Esta primera aproximación, la cual fue propuesta por CRIDA, nos ha permitido comprender el problema en profundidad y ha facilitado su posterior resolución. También debemos indicar que el problema corresponde a la fase pre-táctica.

En la primera aproximación, podríamos destacar tres puntos fundamentalmente. El

### **3.1. PRIMERA APROXIMACIÓN AL PROBLEMA**

---

primero es que disponemos de un conjunto de restricciones menos estricto que el problema real. El segundo punto hace referencia al tamaño del problema, para la primera aproximación se resuelve un día completo (24 horas), mientras que en la versión completa resolvemos el problema de turno en turno. Y por último, el tercero, disponemos de un número elevado de controladores para resolver el problema.

El principal objetivo para la resolución del problema es obtener una solución que satisfaga el conjunto de restricciones del problema, independientemente del número de controladores necesarios para hacerlo. A su vez, se tiene como objetivo minimizar el número de controladores utilizados y una vez cumplido este objetivo, también se propone equilibrar la carga de trabajo entre los turnos de trabajo de los controladores.

#### **3.1.1. Restricciones**

Los controladores tienen un amplio conjunto de condiciones laborales (restricciones) reguladas e impuestas por ley, las cuales deben ser respetadas. Además de las restricciones establecidas, existen restricciones lógicas y prácticas asociadas a la naturaleza del problema.

Para esta primera aproximación al problema, se tiene un conjunto de restricciones menos estricto, permitiendo la obtención de una solución factible de una forma sencilla.

Para el segundo problema, el conjunto es más estricto lo cual provoca, como se verá más adelante, que los esfuerzos de cómputo se centren en la obtención de una solución factible, es decir, que cumpla con todas las restricciones del problema.

Algunas de las restricciones de los controladores aéreos vienen reguladas por el Real Decreto 1001/2010 y la Ley 9/2010, que regulan los proveedores de servicios de tráfico aéreo, en concreto las restricciones utilizadas para la aproximación al problema son las siguientes:

1. Un controlador no puede trabajar más de ocho horas.
2. Un controlador tiene un único turno de trabajo asignado y no puede trabajar fuera de este, con independencia de la duración.
3. Un controlador debe descansar el 25 % del total de su jornada laboral mientras sea

en un turno de día, y un 50 % en el caso de tratarse de un turno de noche.

4. Un controlador no puede trabajar más de dos horas consecutivas.
5. El descanso debe tener una duración mínima de media hora.
6. Un controlador debe permanecer en el mismo sector y misma posición durante al menos media hora.
7. Un controlador no puede cambiar más de dos veces de sector, independientemente de la posición dentro del sector, en un mismo turno entre sectores no afines.
8. Los tiempos de ocupación de las posiciones en los puestos deben estar compensados.
9. En cada slot de tiempo, un controlador solo puede ocupar una posición de un único sector.
10. En cada slot de tiempo un sector tiene que tener asignado un único controlador por cada posición, teniendo el sector asociadas dos posiciones.
11. No se permite cambiar de sector sin descanso, a menos que se produzca un cambio de configuración y que los sector sean afines, en cuyo caso, se permite cambiar pero se exige permanecer un mínimo de quince minutos. En caso contrario, se necesita realizar un descanso para poder cambiar de sector.
12. Los sectores que se abran deben permanecer abiertos durante veinte minutos (esta restricción afecta indirectamente a la solución, ya que afecta a la entrada del problema).

### 3.1.2. Modelización matemática

Para el mejor entendimiento del problema, se procedió a la definición de las variables binarias y la modelización matemática mediante programación lineal de las restricciones y función objetivo del problema. No se ha conseguido modelizar por programación lineal todas las restricciones, dado que existen restricciones como la restricción que involucra el cambio de posición entre sectores afines que no han podido ser modelizadas.

### 3.1. PRIMERA APROXIMACIÓN AL PROBLEMA

---

Para la resolución de un día de trabajo, 24 horas, se divide el día en intervalos de 5 minutos (slots), habiendo un total de 288 slots.

- Entre los slots 1 y 88, ambos incluidos, (de 14:00 a 21:20) trabajan los controladores del turno de tarde (CTT) y los controladores del turno de tarde largo (CTTL).
- Entre los slots 89 y 100, ambos incluidos, (de 21:20 a 22:20) trabajan los controladores del turno de tarde largo (CTTL) y los controladores del turno de noche (CTN).
- Entre los slots 101 y 188, ambos incluidos, (de 22:20 a 5:40) trabajan los controladores del turno de noche (CTN).
- Entre los slots 189 y 196, ambos incluidos, (de 5:40 a 6:20) trabajan los controladores del turno de noche (CTN) y los controladores del turno de mañana largo (CTML).
- Entre los slots 197 y 288, ambos incluidos, (de 6:20 a 14:00) trabajan los controladores del turno de mañana largo (CTML) y los controladores del turno de mañana corto (CTM).

Dicho de otra forma:

- Los CTT pueden trabajar entre los slots 1 y 88, ambos incluidos (de 14:00 a 21:20).
- Los CTTL pueden trabajar entre los slots 1 y 100, ambos incluidos (de 14:00 a 22:20).
- Los CTN pueden trabajar entre los slots 89 y 196, ambos incluidos (de 21:20 a 6:20).
- Los CTML pueden trabajar entre los slots 189 y 288, ambos incluidos (de 5:40 a 14:00).
- Los CTM pueden trabajar entre los slots 197 y 288, ambos incluidos (de 6:20 a 14:00).

La longitud de los turnos puede variar dependiendo del centro de control, pero el período completo siempre será de 288 slots.

Consideremos las variables  $x_{ijklm_l}$ , donde  $i$  ( $i \in \{1, 2, 3, \dots, c\}$ ) denota al controlador,  $j$  ( $j \in \{1 = T, 2 = TL, 3 = N, 4 = ML, 5 = M\}$ ) al tipo de turno que se trata,  $k$  ( $k \in \{1 = \text{descansando}, 2 = \text{ejecutivo}, 3 = \text{planificador}\}$ ) al estado del controlador,  $l$  ( $l \in \{1 = (0, 5], 2 = (5, 10], \dots, 288 = (1435, 1440]\}$ ) es el slot de tiempo considerado y  $m_l$  ( $m_l \in \{1, 2, \dots, s_l\}$ ) es el número de sectores abiertos en el slot de tiempo  $l$ .

$s_l$  es el número de sectores abiertos en el slot de tiempo  $l$ . Son valores conocidos asociados a la configuración de sectores del turno examinado.

Estas variables son binarias, valiendo 1 cuando el controlador  $i$  se encuentra trabajando en el turno  $j$ , en el estado  $k$ , en el período  $l$ , en el sector  $m_l$  y 0 en caso contrario. En total, tenemos  $c \times 5 \times 3 \times s_1 \times \dots \times s_{288}$  variables de decisión.

El objetivo del problema es minimizar el número de controladores necesarios, lo que es equivalente a minimizar el número de descansos para  $c$  controladores:

$$\min z = \sum_{i=1}^c \sum_{j=1}^5 \sum_{l=1}^{288} \sum_{m_l=1}^{s_l} x_{ijlm_l}.$$

A continuación, procedemos con la modelización de las restricciones:

1. Un controlador no puede trabajar más de ocho horas (96 slots de cinco minutos).

$$\sum_{j=1}^5 \sum_{k=1}^3 \sum_{l=1}^{288} \sum_{m_l=1}^{s_l} x_{ijklm_l} \leq 96, \forall i.$$

El número de restricciones es  $c$ .

2. Un controlador tiene un único turno de trabajo asignado y no puede trabajar fuera de este, con independencia de la duración.

$$3y_{ij} \leq \sum_{k=1}^3 \sum_{l=1}^{288} \sum_{m_l=1}^{s_l} x_{ijklm_l} \leq 288y_{ij}, \forall i, j, \quad \sum_{j=1}^5 y_{ij} = 1, \forall i.$$

El número de restricciones es  $2c + 5$ .

3. Un controlador debe descansar el 25 % del total de su jornada laboral mientras sea en un turno de día, y un 50 % en el caso de tratarse de un turno de noche.

### 3.1. PRIMERA APROXIMACIÓN AL PROBLEMA

---

Si consideramos que el turno de tarde abarca del slot 1 al 88, ambos incluidos:

$$0.25 \sum_{k=2}^3 \sum_{l=1}^{88} \sum_{m_l=1}^{s_l} x_{i1klm_l} \leq \sum_{l=1}^{88} \sum_{m_l=1}^{s_l} x_{i11lm_l}, \forall i.$$

Se procede análogamente para el resto de turnos. El número de restricciones es  $c \times 5$ .

4. Un controlador no puede trabajar más de dos horas consecutivas.

Para comprobar esta restricción consideramos un período  $T_r$  de 25 slots consecutivos, obteniendo un tiempo de 2 horas y 5 minutos. Vamos a ilustrarlo con el turno de tarde ( $TT$ ). Desde  $r = 1$  hasta 64 ( $T_{64} = [64, 88]$ ) comprobando que el número de slots de trabajo en  $T_r$  sea menor a 25 para cada controlador.

$$\sum_{k=2}^3 \sum_{l=1}^{25} \sum_{m_l=1}^{s_l} x_{i1klm_l} < 25, \quad \forall i \quad (r = 1),$$

...

$$\sum_{k=2}^3 \sum_{l=64}^{88} \sum_{m_l=1}^{s_l} x_{i1klm_l} < 25, \quad \forall i \quad (r = 64).$$

Debemos repetir el proceso para los turnos restantes. Para el turno de tarde largo con  $T_1 = [1, 25]$  y  $T_{76} = [76, 100]$ , añadiendo 76 restricciones. Para el turno de noche,  $T_1 = [89, 113]$  y  $T_{172} = [172, 196]$ , añadiendo 88. En el turno de mañana,  $T_1 = [197, 221]$  y  $T_{264} = [264, 288]$ , con 76 restricciones. Y por último el turno de mañana largo con  $T_1 = [189, 213]$  y  $T_{264} = [264, 288]$ , con 84.

Debemos comprobar todo el conjunto de restricciones para cada controlador  $i$ , por lo que el número de restricciones totales es  $(64 + 76 + 88 + 76 + 84) \times c = 388 \times c$ .

5. El descanso debe tener una duración mínima de media hora.

Para comprobar esta restricción verificamos para cada controlador que si se encuentra trabajando en el slot  $l$  y además en el slot  $l + 1$  se encuentra descansando, se debe cumplir que del slot  $l + 1$  al  $l + 5$  el controlador se encuentre en posición de descanso (6 slots, 30 minutos). Vamos a ilustrar este proceso con el turno de tarde, que comprende del slot 1 al 88 ambos incluidos. La condición es la siguiente:

Si  $\sum_{m_l=1}^{s_l} x_{i12lm_l} + \sum_{m_l=1}^{s_l} x_{i13lm_l} > 0$  (el controlador se encuentra trabajando en el slot  $l$ ) y  $\sum_{m_{l+1}=1}^{s_{l+1}} x_{i11l+1m_{l+1}} > 0$  (el controlador se encuentra descansando en el slot  $l+1$ ), entonces  $\sum_{r=l+1}^{l+5} \sum_{k_r=1}^{s_r} x_{i11rk_r} - 6 \geq 0$  (el controlador descansa, como mínimo, 30 minutos).

Es decir:

$$\begin{aligned} & - \sum_{m=l+1}^{l+5} \sum_{k_m=1}^{s_m} x_{i11mk_m} + 6 \leq M_1 y_{il1}, \forall i, l, \\ & \sum_{k_l=1}^{s_l} x_{i12lk_l} + \sum_{k_l=1}^{s_l} x_{i13lk_l} \leq M_1(1 - y_{il1}), \forall i, l, \\ & \sum_{k_{l+1}=1}^{s_{l+1}} x_{i11l+1k_{l+1}} \leq M_1(1 - y_{il1}), \forall i, l, \quad y_{il1} \in \{0, 1\} \end{aligned}$$

El valor de  $M_1$  es elegido de la siguiente manera:

$$\begin{aligned} & - \sum_{m=l+1}^{l+5} \sum_{k_m=1}^{s_m} x_{i11mk_m} + 6 \leq M_1, \forall i, l, \\ & \sum_{k_l=1}^{s_l} x_{i12lk_l} + \sum_{k_l=1}^{s_l} x_{i13lk_l} \leq M_1, \forall i, l, \quad \sum_{k_{l+1}=1}^{s_{l+1}} x_{i11l+1k_{l+1}} \leq M_1, \forall i, l. \end{aligned}$$

El número de restricciones es  $c \times 288 \times 3$ .

6. Un controlador debe permanecer en el mismo sector y misma posición durante al menos media hora.

Esta restricción se comprueba de manera similar a la anterior. Vamos a ilustrar el proceso con el turno de tarde. La condición que se tiene que comprobar es la siguiente: Si  $\sum_{m_l=1}^{s_l} x_{i11lm_l} > 0$  (el controlador  $i$  descansa en  $l$ ) y  $\sum_{m_{l+1}=1}^{s_{l+1}} x_{i12l+1m_{l+1}} > 0$  (el controlador trabaja en posición de ejecutivo en el slot  $l+1$ ). Entonces,  $\sum_{r=l+1}^{l+5} \sum_{m_r=1}^{s_r} x_{i12rm_r} - 6 \geq 0$  (el controlador tiene que permanecer trabajando como ejecutivo en el mismo sector durante media hora como mínimo).

En definitiva:

$$- \sum_{m=l+1}^{l+5} \sum_{k_m=1}^{s_m} x_{i12mk_m} + 6 \leq M_2 y_{il2}, \quad \sum_{k_l=1}^{s_l} x_{i11lk_l} \leq M_2(1 - y_{il2}), \forall i, l,$$

### 3.1. PRIMERA APROXIMACIÓN AL PROBLEMA

---

$$\sum_{k_{l+1}=1}^{s_{l+1}} x_{i12l+1k_{l+1}} \leq M_2(1 - y_{il2}), \forall i, l, \quad y_{il2} \in \{0, 1\}.$$

El valor de  $M_2$  debe ser escogido de la siguiente manera:

$$- \sum_{m=l+1}^{l+5} \sum_{k_m=1}^{s_m} x_{i12mk_m} + 6 \leq M_2, \forall i, l,$$

$$\sum_{k_l=1}^{s_l} x_{i11lk_l} \leq M_2, \forall i, l, \quad \sum_{k_{l+1}=1}^{s_{l+1}} x_{i12l+1k_{l+1}} \leq M_2, \forall i, l.$$

Para los controladores en la posición de planificador se realiza análogamente:

$$- \sum_{m=l+1}^{l+5} \sum_{k_m=1}^{s_m} x_{i13mk_m} + 6 \leq M_3 y_{il3}, \quad \sum_{k_l=1}^{s_l} x_{i11lk_l} \leq M_3(1 - y_{il3}), \forall i, l,$$

$$\sum_{k_{l+1}=1}^{s_{l+1}} x_{i13l+1k_{l+1}} \leq M_3(1 - y_{il3}), \forall i, l, \quad y_{il3} \in \{0, 1\}.$$

El valor de  $M_3$  debe ser escogido de la siguiente manera:

$$- \sum_{m=l+1}^{l+5} \sum_{k_m=1}^{s_m} x_{i13mk_m} + 6 \leq M_3, \forall i, l,$$

$$\sum_{k_l=1}^{s_l} x_{i11lk_l} \leq M_3, \forall i, l, \quad \sum_{k_{l+1}=1}^{s_{l+1}} x_{i13l+1k_{l+1}} \leq M_3, \forall i, l.$$

El número de restricciones es  $288 \times c \times 3$ . Este valor debe ser multiplicado por 2 debido a que deben considerarse las posiciones de ejecutivo y planificador.

7. Un controlador no puede cambiar más de dos veces de sector independientemente de la posición dentro del sector, en un mismo turno entre sectores no afines.

Para el turno de tarde:

$$\sum_{j=1}^5 \sum_{k=2}^3 \sum_{l=1}^{288} x_{ijkl1} \leq M_4 z_{i1}, \forall i \quad \dots \quad \sum_{j=1}^5 \sum_{k=2}^3 \sum_{l=1}^{288} x_{ijklmax_l} \leq M_4 z_{imax}, \forall i,$$

$$\sum_{k=1}^{max} z_{ik} \leq 2, \forall i.$$

Y análogamente se realiza con el resto de turnos.

8. Los tiempos de ocupación de las posiciones deben estar equilibrados en un 40-60 % entre las posiciones de planificador y ejecutivo.

Para el turno de tarde:

$$\left| \sum_{l=1}^{88} \sum_{m_l=1}^{s_l} x_{i13lm_l} - \sum_{l=1}^{88} \sum_{m_l=1}^{s_l} x_{i12lm_l} \right| \leq 0,2 \times \sum_{k=2}^3 \sum_{l=1}^{88} \sum_{m_l=1}^{s_l} x_{i1klm_l}, \forall i.$$

Se procede análogamente con el resto de turnos. El número de restricciones es  $c \times 10$ .

9. En cada slot de tiempo, un controlador solo puede ocupar una posición de un único sector:

$$\sum_{j=1}^5 \sum_{k=2}^3 \sum_{m_l=1}^{s_l} x_{ijk1lm_l} \leq 1, \quad \forall i, l.$$

El número de restricciones es  $c \times 288$ .

10. En cada slot de tiempo un sector tiene que tener asignado un único controlador por cada posición, teniendo el sector asociadas dos posiciones:

$$\sum_{i=1}^c \sum_{j=1}^5 x_{ij1lm_l} = 1, \quad \sum_{i=1}^c \sum_{j=1}^5 x_{ij2lm_l} = 1, \quad \forall l, m_l \in \{1, 2, \dots, s_l\}.$$

El número de restricciones es  $2 \times 288 \times (s_1 + s_2 + \dots + s_{288})$ .

11. No se permite cambiar de sector sin descanso, a menos que se produzca un cambio de configuración y que los sectores sean afines, en cuyo caso, se permite cambiar pero se exige permanecer un mínimo de quince minutos. En caso contrario, se necesita realizar un descanso para poder cambiar de sector.

Si no tenemos en cuenta la afinidad entre sectores por simplificar la modelización, las restricciones son las siguientes:

$$x_{ij2l+1m_{l+1}} \leq 1 - x_{ij3l+1m_{l+1}}, \forall i, j, \quad x_{ij3l+1m_{l+1}} \leq 1 - x_{ij2l+1m_{l+1}}, \forall i, j.$$

### 3.1. PRIMERA APROXIMACIÓN AL PROBLEMA

---

El número de restricciones es  $2(c + 5)$ .

12. Los sectores que se abran deben permanecer abiertos durante veinte minutos (esta restricción afecta indirectamente a la solución, ya que afecta a la entrada del problema).

Si tomamos como referencia la sectorización del turno mostrado en la Figura 3.8, y teniendo en cuenta que la cota superior del número de controladores que se tiene es 40 ( $c = 3 \times 3 \times 4.347$ ), el número de variables de decisión del problema lineal 0-1 a resolver sería:

$$c \times 5 \times 3 \times (s_1 + s_2 + \dots + s_{288}) = 40 \times 5 \times 3 \times (1 \times 52 + 2 \times 12 + 4 \times 12 + 6 \times 20 + 7 \times 48 + 6 \times 20 + 5 \times 40 + 6 \times 36 + 5 \times 16 + 3 \times 12 + 1 \times 20) = 40 \times 5 \times 3 \times 1252 = 751200$$

variables binarias.

A estas variables habría que sumar todas las variables binarias auxiliares que se han utilizado para la modelización de las restricciones.

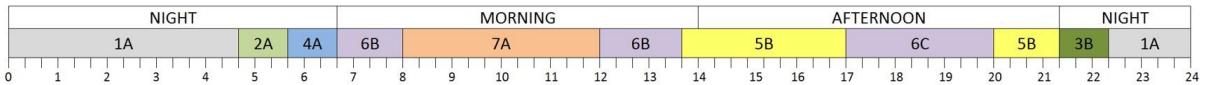


Figura 3.8: Ejemplo de sectorización

Por otro lado, el número total de restricciones asociadas a cada una de las condiciones de contorno establecidas sería:

1. 5
2.  $c \times 6 = 240$
3.  $c \times 288 = 11.520$
4.  $(s_1 + s_2 + \dots + s_{288}) \times 2 = 2.504$
5.  $c \times 5 = 200$
6.  $c \times 388 = 11.520$
7.  $c \times 1104 = 44.160$

$$8. c \times 2208 = 88.320$$

$$9. c \times 7 = 2.800$$

$$10. c \times 10 = 400$$

siendo el número total de restricciones 161.669.

### 3.2. Versión completa del problema

Una vez resuelta la aproximación al problema, comenzamos con la resolución del problema completo. La principal diferencia con la aproximación al problema se encuentra en que las condiciones que se deben cumplir son más restrictivas, y además, tenemos un número de controladores fijo, lo cual puede llevarnos a una solución infactible en casos extremos. Por esto último, no es tan sencillo partir de una solución tan cercana a la factibilidad como se realiza en la aproximación. También hay que destacar que la dimensión del problema es ligeramente menor debido a que la resolución se realiza turno por turno y no de un día completo.

Al igual que para la aproximación al problema expuesta previamente, es imprescindible que la solución proporcionada cumpla todas las restricciones. A lo anteriormente mencionado, debemos utilizar una perspectiva multiobjetivo, ya que añadimos un conjunto de objetivos que se deben optimizar:

1. El primer objetivo es cumplir las siguientes condiciones:

- a) El tiempo óptimo de trabajo continuado de un controlador que se encuentra trabajando en el mismo sector y posición (ejecutivo y planificador) es de 45 minutos.
- b) El tiempo óptimo de trabajo continuado entre descansos de un controlador independientemente del sector y posición que ocupe es de 90 minutos.
- c) El porcentaje de tiempo que un controlador trabaja en posiciones ejecutivas debe estar comprendido entre el 40 % y el 60 % del trabajo total realizado (sin contar con los descansos).

### **3.2. VERSIÓN COMPLETA DEL PROBLEMA**

---

2. El segundo objetivo es mantener una estructura o apariencia similar a la de los estadillos o plantillas usadas anteriormente para ofrecer las soluciones.
3. El tercer objetivo está compuesto por dos sub-objetivos, los cuales están relacionados con los cambios en sala:
  - a) Por un lado, se trata de minimizar el número de intervalos de descanso de los controladores aéreos.
  - b) Por otro, maximizar el número de sectores que influyen en las acreditaciones (*sectores elementales*), es decir, que los controladores ocupen posiciones en varios sectores determinados para evitar que pierdan las acreditaciones por falta de práctica.
4. El cuarto objetivo es realizar una distribución de la carga de trabajo equilibrada entre todos los controladores.

#### **3.2.1. Restricciones**

Algunas restricciones expuestas en la primera aproximación al problema (Sección 3.1.1) se mantienen en la versión completa, en concreto las restricciones 4, 5, 9 y 12. Además de estas restricciones, debemos incorporar las siguientes (algunas de las cuales son modificaciones de las consideradas en la primera aproximación):

1. Los controladores aéreos solo pueden operar en sectores que pertenezcan al núcleo en el que estén acreditados.
2. Los controladores aéreos con acreditación CON solo pueden operar en sectores de tipo ruta.
3. Los controladores aéreos deben descansar un 25 % de su turno cuando este es un turno de día (TT, TTL, TM, TML). En el turno de noche estos deben descansar un 33 %.
4. Los sectores abiertos durante toda la noche deben ser operados por cuatro controladores.

5. Los controladores aéreos deben descansar un mínimo de treinta minutos cada dos horas de trabajo. Estas no tienen porqué ser consecutivas, lo que quiere decir que, en una ventana de trabajo de dos horas y media, se tiene que descansar como mínimo media hora.
6. Los cambios de los controladores entre sectores cuando se encuentran trabajando en posición de ejecutivo, no están permitidos sin un descanso excepto que se produzca un cambio de configuración y los sectores sean afines.
7. El período mínimo de trabajo de un controlador aéreo entre dos descansos es de quince minutos.
8. La duración de los períodos de descanso de un controlador aéreo debe ser de al menos quince minutos.
9. Los controladores aéreos deben permanecer en el mismo sector y posición, un mínimo de quince minutos, antes de poder cambiar a otro sector o posición.
10. Los controladores aéreos no pueden trabajar en más de tres sectores no afines en un único turno.
11. Todos los controladores aéreos deben tener un turno de trabajo asignado y todos los turnos de trabajo deben estar asignados a un controlador.
12. Ningún controlador aéreo puede descansar durante el turno completo, por lo que como mínimo trabajará quince minutos.

### **3.3. Problema táctico**

El problema debe resolverse el mismo día de operación durante la fase táctica, cuando el turno ya ha comenzado. En ese momento, surge un imprevisto, por ejemplo, un aumento del tráfico aéreo esperado debido al cierre de otro aeropuerto por empeoramiento de las condiciones climatológicas o una baja por enfermedad de un controlador.

Este problema se caracteriza por la necesidad de resolverlo en un tiempo mínimo, debido a la naturaleza del problema.

### 3.3. PROBLEMA TÁCTICO

---

En este caso, debemos optimizar la asignación de turnos de trabajo previamente realizada en ese mismo momento, desde el momento que se produce el imprevisto hasta el final del turno, pero teniendo en cuenta la parte del turno ya ejecutada. El objetivo principal es el de minimizar el número de condiciones laborales incumplidas, y dando solución al problema sin causar situaciones de peligro.

Las condiciones laborales de los controladores son las mismas que las expuestas en el segundo problema, en la Sección 3.2.1.

La principal diferencia con los otros dos problemas expuestos, se encuentra en que el problema se debe resolver durante la fase táctica, cuando ya ha comenzado el turno de trabajo. Esto nos obliga a proporcionar una solución rápida, y a tener en consideración el trabajo realizado previamente por los controladores antes del imprevisto surgido. Otra diferencia destacable, es la disminución del tamaño del problema, debido a que solo se realizan modificaciones en la parte del turno posterior al imprevisto.

Al igual que en el resto de problemas expuestos previamente, el objetivo más importante es obtener una solución factible. Debido a tener que considerar el trabajo realizado previamente por los controladores, ofrecer una solución factible no es una tarea sencilla.

A esto, debemos añadir una serie de objetivos que se deben optimizar:

1. Ofrecer una solución que cumpla con el mayor número de condiciones laborales posibles.
2. Minimizar el número de cambios de los controladores a otras posiciones de control en el momento del imprevisto. El objetivo es minimizar el riesgo de generar conflictos en la sala de control causados por una gran cantidad de cambios simultáneos.
3. Mantener en la medida de lo posible la estructura del horario inicialmente utilizado, con el objetivo de facilitar el trabajo a los controladores.



---

## Capítulo 4

# Problemas de asignación y extensiones

---

Los problemas de asignación o *scheduling problems* pueden ser entendidos como la asignación de recursos, ya sean personas, máquinas, clases o trabajos, a un patrón de espacio o tiempo, verificando un conjunto de restricciones y optimizando uno o varios objetivos.

Los problemas de asignación son problemas combinatorios, y factores como el tamaño y la complejidad en este tipo de problemas los convierte en problemas difíciles de resolver, en ocasiones, incluso imposible con métodos exactos. Atendiendo a su complejidad, según la teoría de la complejidad computacional, este tipo de problemas se encuentran dentro de la clase de complejidad *NP-Complete*, es decir, no se conoce ningún algoritmo capaz de resolverlo en tiempo polinómico.

Se han utilizado algunos métodos exactos para la resolución de estos problemas, pero cuando la dimensión del problema aumenta o las restricciones o la función objetivo lo convierten en un problema no lineal, los métodos exactos deben ser descartados. Por esto, en la literatura se encuentran numerosos ejemplos de metaheurísticas utilizadas para la resolución de problemas de asignación.

En diversos estudios podemos encontrar un gran número de problemas categorizados como problemas de asignación. Tan solo distinguir los diferentes tipos de problemas de asignación puede resultar una tarea compleja. Con el propósito de facilitar esta tarea mostramos en la Figura 4.1, una clasificación de los distintos tipos de problemas que abordaremos a lo largo del capítulo. En [103] se define la diferencia entre estos problemas, lo cual puede servir de apoyo a la hora de clasificarlos. En primer lugar, se debe considerar el objetivo de los problemas de asignación como la resolución práctica de un problema

relacionado con la colocación/asignación de una serie de recursos, sujetos a restricciones, en el espacio y el tiempo, utilizando o desarrollando las herramientas que se consideren oportunas.

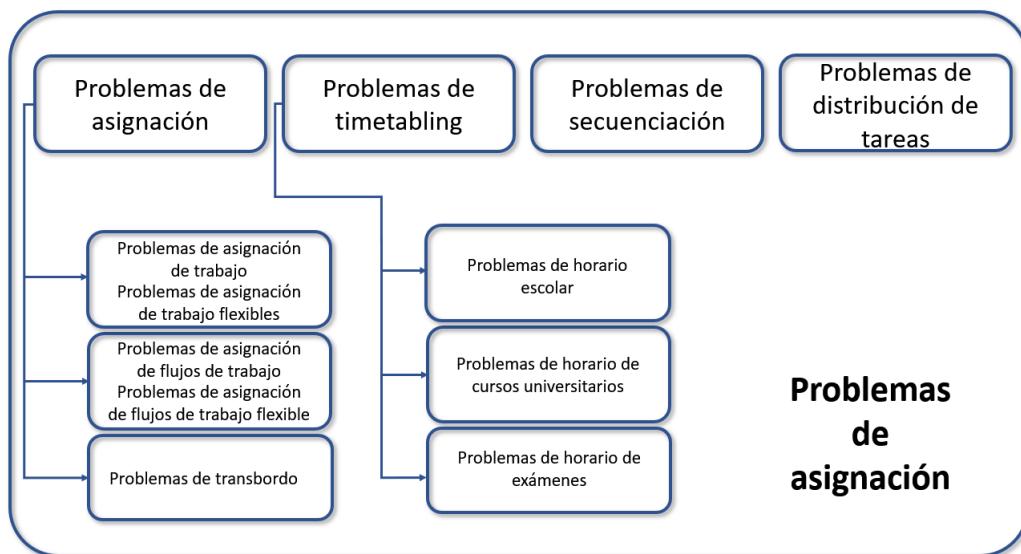


Figura 4.1: Clasificación de los tipos de problemas de asignación

Igualmente, puede considerarse la asignación y la creación de horarios como actividades separadas. En este sentido, el término de asignación es utilizado como un término genérico que engloba todos estos tipos de problemas, y pueden considerarse los problemas de horarios, secuenciación y distribución de tareas como casos especiales de la actividad genérica de asignación.

Los problemas de horarios o *timetabling problems* se entienden como aquellos problemas en los que un evento debe asignarse a un lugar, aunque esto no implica necesariamente la asignación de recursos. Por ejemplo, en un problema de horarios de autobuses solo debemos mostrar el horario que tiene cada ruta, independientemente de los vehículos o conductores que se necesiten. Esta última parte podría considerarse como actividad de asignación. Para los problemas de horarios escolares, únicamente se tiene en cuenta qué clases se imparten y a qué hora. Un ejemplo sencillo sería simular una escuela de educación infantil, donde un profesor imparte todas las asignaturas y solo existe una única aula. Otros casos con mayor complejidad serían los problemas de horarios universitarios, los cuales implican la asignación de aulas entre otros recursos. Se podría considerar que este tipo de problemas tienen actividades de asignación.

#### 4.1. PROBLEMAS DE ASIGNACIÓN O *SCHEDULING*

---

En [103] se muestran otros dos tipos de problemas similares, aunque estos se distinguen con mayor facilidad. Se trata de los problemas de secuenciación o *sequencing problems* y de los problemas de distribución de tareas *rostering problems*.

Los *problemas de secuenciación* [9] son problemas en los que solo se debe asignar un orden a una serie. Podría ser el orden de visita de lugares o el orden de tareas a realizar. Los objetos de la serie se ordenan con el objetivo de optimizar los costes o recursos. El ejemplo más conocido es el problema del viajante (*Traveling Salesman Problem*, TSP), en el cual se deben visitar un número determinado de lugares con el objetivo de minimizar un coste, el cual puede ser la distancia que se recorre, el tiempo que se consume o el precio del desplazamiento derivado de visitar estos lugares en el orden establecido.

Los *problemas de distribución de tareas* [18, 36] tienen como objetivo la distribución de una serie de recursos en lugares determinados, estando sujetos a un conjunto de restricciones, debiendo distribuirse optimizando un objetivo (tiempo o coste), o simplemente alcanzando una solución factible. Uno de los problemas más conocidos es el *problema de n-reinas* [87], que consiste en asignar  $n$  reinas en un tablero de ajedrez de  $n$  filas y  $n$  columnas sin que las reinas comparten ninguna fila, columna o diagonal.

Algunos de los problemas pueden encajar en más de una de las definiciones anteriormente expuestas. Varios de ellos tienen características comunes y es habitual utilizar indistintamente cualquiera de los términos expuestos para referirse a estos problemas.

#### 4.1. Problemas de asignación o *scheduling*

Los *problemas de asignación* [17, 25] tienen multitud de variantes. A continuación procederemos a explicar los problemas de asignación de trabajo o *job shop scheduling problems*. El objetivo de este problema es elaborar una serie de productos en el menor tiempo posible, y para cumplir este objetivo disponemos de un conjunto de  $M$  máquinas, las cuales son utilizadas para realizar tareas u operaciones. Las operaciones ( $O$ ) serán cada uno de los pasos necesarios para finalizar el producto. Por otro lado, llamaremos trabajo ( $T$ ), a la secuencia ordenada de operaciones, las cuales llevan asociadas un coste de tiempo.

De acuerdo a la clasificación de problemas de asignación, en el grupo de los problemas

de asignación determinísticos encontramos dos ramas: los problemas de asignación de trabajo y los problemas de asignación de flujos de trabajo o *flow shop scheduling problems*.

- Los *problemas de asignación de trabajo* son problemas donde tenemos un conjunto de operaciones o tareas, y cada operación es realizada por una única máquina, la cual tarda un tiempo determinado. Existe una generalización llamada problemas de asignación de trabajo flexibles o *flexible job shop scheduling problems*, en la que una máquina puede realizar un conjunto de operaciones.

Para la resolución de estos problemas se han utilizado multitud de algoritmos. En [27] utilizan algoritmos genéticos para la resolución de estos problemas, mientras que en [21] hacen uso de *sistemas de hormigas* [28] el cual aplican al problema de asignación de trabajo y en [101] utilizan *recocido simulado* y comparan los resultados con los obtenidos en otras publicaciones que abordan este mismo problema.

En el caso de los problemas de asignación de trabajo flexibles podemos apreciar que los métodos utilizados son similares. Ejemplo de ello es [63], donde aplican *algoritmos genéticos* para la resolución del problema, mientras que en [104] proponen un enfoque híbrido utilizando *recocido simulado* junto a *optimización por enjambres de partículas*.

- Los *problemas de asignación de flujos de trabajo* son similares a los problemas de asignación de trabajo, pero la diferencia radica en que las operaciones deben seguir un orden establecido y se dispone de una máquina para cada una de las tareas.

Se han utilizado distintos métodos para la resolución de estos problemas, por ejemplo en [82] se utiliza un *algoritmo discreto de colonias de abejas*, el cual comparan con otros métodos, como los *algoritmos genéticos*. En [77] analizan el rendimiento de los *algoritmos genéticos* en comparación con otros métodos, como *búsqueda tabú* y *recocido simulado*. Además, proponen dos hibridaciones distintas de los *algoritmos genéticos* con búsqueda local y con *recocido simulado*, siendo el objetivo mejorar su rendimiento.

Existe una variante como el problema de asignación de flujos de trabajo flexible o *flexible flow shop scheduling problem*. En ella, cabe la posibilidad de tener un

## 4.2. PROBLEMAS DE HORARIOS O *TIMETABLING*

---

mayor número de máquinas en paralelo destinadas a una sola tarea. También es conocido como problema híbrido de asignación de flujo de trabajo (*Hybrid Flow shop Scheduling Problem*, HFSP). En [69] y [88] encontramos un amplio conjunto de técnicas utilizadas para su resolución.

Por ejemplo, en [88] recogen algunos de los algoritmos exactos utilizados para resolver los HFSP como son el *algoritmo de ramificación y acotación* (*Branch & Bound*, B&B) [68]. Estos algoritmos son expuestos haciendo mención a la incapacidad de estos para la resolución de instancias medianas o grandes o problemas complejos reales. También hace una revisión de un gran número de artículos en los que se propone el uso de heurísticas, programación matemática y metaheurísticas aplicado al problema. Las metaheurísticas más utilizadas son *búsqueda tabú*, *recocido simulado*, *algoritmos genéticos* o *sistemas de colonias de hormigas*, entre otras.

Otra categoría dentro de los problemas de asignación son los *problemas de transbordo* (*transshipment problems*) [55, 59]. Para este problema disponemos de un conjunto de nodos de origen ( $m$ ) y destino ( $n$ ). El objetivo del problema es satisfacer la demanda de los nodos de destino minimizando los costes de transporte del producto entre los distintos nodos. Se pueden realizar envíos entre todos los nodos, ya sean de origen o destino, pero el coste del envío no dependerá de la cantidad enviada, sino que el coste es independiente y únicamente depende del tiempo de transporte entre nodos.

## 4.2. Problemas de horarios o *timetabling*

Tal como sucede en los problemas de asignación, los problemas de horarios han ido desarrollando variantes a lo largo de los años. Ha sido tal el desarrollo, que ha sido necesaria la creación de clasificaciones donde agruparlos. Una de las clasificaciones más utilizadas la encontramos en [89], divide los problemas de horarios en tres grandes áreas:

- Los problemas de horario escolar (*school timetabling problems*) [3, 20]. Para la resolución de estos problemas se trata de generar el horario de clases semanal de un colegio, evitando que dos profesores coincidan en la misma clase en el mismo momento y que dos clases las imparta el mismo profesor al mismo tiempo. La versión

básica de este problema no contempla que alguna clase o profesor no esté disponible en ciertos momentos, pero existen variantes que introducen este tipo de restricciones. Además, encontramos multitud de variantes que contienen restricciones de mayor complejidad, como el uso de salas especiales, más de un profesor por asignatura o clases simultáneas.

A continuación, se muestran algunos de los métodos utilizados para la resolución de estos problemas. En [3] se resuelve un problema de horario escolar utilizando el algoritmo de recocido simulado de forma secuencial, y añaden algunas mejoras de paralelización en la asignación de aulas cuando el número de aulas disponible no esté próximo a cero.

Encontramos otros métodos como los *algoritmos genéticos* y la *búsqueda tabú* propuestos para la resolución de estos problemas en [20] donde se comparan junto con el *recocido simulado*. Además, en este artículo se comparan los resultados obtenidos por los algoritmos genéticos y los obtenidos por una variante añadiendo un proceso de *búsqueda local* a los *algoritmos genéticos*. Entre todos los métodos utilizados, los mejores resultados se obtienen por parte de los *algoritmos genéticos* con *búsqueda local*.

En [5] proponen el uso de un método de resolución dividido en tres fases para la resolución de este tipo de problemas. Primero, utilizan varias heurísticas en paralelo para la creación de soluciones iniciales. A continuación, proponen la metaheurística de *búsqueda tabú* para la obtención de horarios que cumplan todas las restricciones. Finalmente, mejoran las soluciones de la segunda fase para generar horarios más compactos en esta última fase.

Otro ejemplo de resolución lo podemos encontrar en [94], donde se propone el uso de un enfoque híbrido para la resolución de estos problemas. En concreto, el uso del *algoritmo GRASP (Greedy Randomized Adaptive Search Procedure)* para la construcción de una solución inicial, y con el objetivo de mejorar esta solución, se utiliza la *búsqueda tabú*. En el caso de que las soluciones generadas no sean factibles se propone un proceso llamado *Intraclasses-Interclasses* para tratar de obtener una solución factible.

## 4.2. PROBLEMAS DE HORARIOS O *TIMETABLING*

---

En [38, 39] proponen un algoritmo híbrido compuesto de tres pasos secuenciales. Primero, proponen utilizar *Kingston high school timetabling engine (KHE)*, es una plataforma que genera soluciones iniciales rápidamente, la plataforma realiza primero una fase estructural, después la asignación de tiempo y finalmente la asignación de recursos. A continuación, proponen el uso del algoritmo de *búsqueda en entornos variables* con seis entornos distintos para mejorar la solución hasta su estancamiento. Por último, utilizan una metaheurística para optimizar la solución hasta cumplir con el límite de tiempo de ejecución.

- Los problemas de horario de cursos universitarios (*university course timetabling problems*). El problema consiste en generar el horario de una semana de conferencias de un conjunto de cursos en una universidad, minimizando el número de conferencias que se solapan con alumnos comunes a estas.

La principal diferencia entre los problemas escolares y los universitarios, reside en que las clases pueden tener distintos grupos de estudiantes, y por lo tanto, clases con estudiantes en común pueden entrar en conflicto.

Debemos señalar que los profesores de escuela suelen impartir más de una asignatura, mientras que en la universidad esto no sucede. Otro factor que puede añadir dificultad al problema es la introducción del tamaño de las aulas, ya que un aula puede no ser suficientemente grande para impartir una clase. Podemos encontrar variantes en las que los tiempos que duran las clases no sean uniformes, o que algunas clases se imparten varias veces a la semana debido a un número elevado de alumnos.

Existen numerosos métodos utilizados para resolver este tipo de problemas. Por ejemplo, en [56] utilizan la *búsqueda tabú* para la resolución de problemas de horarios universitarios a gran escala. En [34] se propone el uso de *algoritmos genéticos* para la resolución de estos problemas cuando presentan un complejo conjunto de restricciones.

- Los problemas de horario de exámenes (*exam timetabling problems*). Consisten en generar el horario de una semana de exámenes evitando que se solapen exámenes

con alumnos comunes y separando lo máximo posible los exámenes de los alumnos.

Algunos autores consideran y formulan de la misma manera este problema y el anterior, pero tienen ciertas diferencias. Por ejemplo, un alumno puede saltarse una clase porque coincide con otra, pero esto nunca puede suceder en un examen. También pueden existir otro tipo de restricciones para que los alumnos no tengan más de un examen el mismo día, o que se necesite más de un profesor por aula.

En [85] podemos observar un estudio de los últimos artículos publicados sobre los distintos métodos utilizados para la resolución de los problemas de horarios de exámenes. En este estudio, se muestran métodos de resolución basados en grafos, técnicas basadas en restricciones como la *programación lógica de restricciones* o técnicas de satisfacción de restricciones. Igualmente, se analizan algunos métodos de resolución basados en distintas metaheurísticas como *búsqueda tabú*, *recocido simulado* o *búsqueda en entornos variables*. Otros métodos propuestos son los algoritmos evolutivos. En este artículo también se recogen otros métodos como *algoritmos meméticos*, *sistemas de colonias de hormigas*, *técnicas multicriterio*, *hiperheurísticas* o *técnicas de clustering*. Por último, se exponen los *benchmarks* utilizados en los distintos artículos y se compara el resultado de los métodos expuestos.

Además de la clasificación propuesta anteriormente podemos encontrar otras clasificaciones en [15, 48].

### 4.3. Problemas de *timetabling* en el ámbito del control del tráfico aéreo

Como ya sabemos, en el control del tráfico aéreo lo que prima es garantizar la seguridad aérea en todo momento. Para esto, es fundamental que los controladores aéreos se encuentren trabajando en uso de sus plenas facultades y se cumplan todos los estándares de seguridad.

Pese a esta premisa, encontramos multitud de problemas de asignación resueltos y estudios sobre la gestión del tráfico aéreo, pero muy pocos sobre la asignación de controladores aéreos a sus puestos de trabajo.

#### 4.3. PROBLEMAS DE *TIMETABLING* EN EL ÁMBITO DEL CONTROL DEL TRÁFICO AÉREO

---

En [97] encontramos un amplio análisis sobre la planificación de recursos humanos asociado al trabajo de los controladores. También presenta un estudio completo sobre los distintos aspectos a tratar en la distribución de tareas. Además, expone las ventajas y limitaciones que pueden surgir de la automatización del proceso de asignación de controladores aéreos mediante software.

Más tarde, en [6] se presenta un estudio complejo de manejo de turnos en el ámbito de los controladores aéreos. Presentan las características del sistema de trabajo por turnos, los distintos tipos de turnos y cómo estas pueden afectar a los trabajadores. Introducen algunos estándares que deben seguirse a la hora de diseñar los turnos para garantizar la productividad, eficiencia y seguridad exigida en este tipo de trabajos.

Existen algunas herramientas y *frameworks* diseñadas para la resolución de problemas de asignación en este ámbito, pero la mayoría de estas no son de libre acceso y los detalles no están disponibles. En [54] se recogen algunas de estas herramientas.

En [22] se hace mención a un algoritmo voraz utilizado para la creación de horarios de trabajo para los controladores aéreos con una granularidad de treinta minutos y un conjunto más laxo de restricciones. Esta Tesis Doctoral utiliza una granularidad menor (cinco minutos), además en [22] el problema ha sido resuelto en base al reglamento de Reino Unido utilizando un conjunto de condiciones menos restrictivo, mientras en este trabajo el conjunto de condiciones es mucho más amplio y aplicamos la normativa española.

En [95] y en [96] encontramos dos estudios sobre la asignación de trabajo a controladores aéreos. En [95] se proponen tres métodos de optimización (*propositional satisfiability problem*, *constraint satisfaction problems* y *integer linear programming*) para la resolución del problema, mientras que en [96] utilizan un enfoque híbrido con el *algoritmo de escalada* (*hill climbing*) y *propositional satisfiability problem*. No obstante, este problema abarca períodos de tiempo más grandes (meses) y se utiliza una granularidad más gruesa, una hora, por lo que las restricciones a tener en cuenta son distintas, y esto deriva en un problema distinto.

Por ejemplo, esta Tesis Doctoral se centra en resolver un único turno de trabajo con una discretización del espacio de tiempo de 5 minutos. La principal diferencia con [95, 96] se encuentra en la discretización y el tipo de restricciones que se aplican. Restricciones como el número máximo de turnos de trabajo que un controlador puede trabajar seguidos

no es un factor a tener en cuenta en esta Tesis Doctoral. Mientras que en el caso de [95, 96] no deben preocuparse por restricciones como el descanso o trabajo mínimo consecutivo.

Otro punto que diferencia ambos trabajos es la incorporación por parte de [95, 96] de las preferencias de los controladores aéreos en los objetivos del problema, mientras que en nuestro caso, el conjunto de objetivos a optimizar se basa en formar un horario fácilmente comprensible, reparta el trabajo equitativamente y facilite el trabajo de los controladores.

---

## Capítulo 5

# Técnicas/Herramientas utilizadas

---

Los problemas de asignación de trabajo y de creación de horarios (*timetabling*) pueden abordarse de múltiples formas. Los métodos clásicos son una de ellas, garantizan alcanzar la solución óptima, pero estos problemas no siempre son lineales. Otro factor a tener en cuenta para descartar los métodos clásicos es el excesivo tiempo de cómputo necesario para resolver los problemas. Normalmente, los problemas de asignación o de creación de horarios son problemas combinatorios con un elevado número de variables y de restricciones, y por lo tanto, estos quedan descartados.

Para la resolución de estos problemas también pueden usarse *heurísticas*, pero con las heurísticas tenemos otros problemas como es la posibilidad de quedarnos atrapados en un óptimo local en el proceso de búsqueda. Otra opción es el uso de *metaheurísticas*, las cuales son algoritmos de optimización que gracias a su flexibilidad son aplicables a un gran número de problemas de optimización. Estos algoritmos realizan una búsqueda inteligente en el espacio de búsqueda del problema y se aplican donde las heurísticas y los métodos clásicos no resultan eficaces. Las metaheurísticas no garantizan encontrar la solución óptima pero sí una solución suficientemente próxima a la óptima.

Las *matheurísticas* o las *hiperheurísticas* son otras técnicas que pueden ser utilizadas. Las matheurísticas son una hibridación entre métodos de programación matemática y las metaheurísticas. En cambio, las hiperheurísticas son métodos que buscan la automatización del proceso de selección, combinación y uso de las heurísticas más adecuada para la resolución de un problema. Para realizar el proceso de selección y combinación de las heurísticas, las hiperheurísticas suelen utilizar técnicas de aprendizaje automático.

A lo largo de los años se han desarrollado numerosas metaheurísticas con las que

trabajar, por lo que vamos a realizar un breve resumen de las más conocidas. Existen múltiples criterios para clasificar las metaheurísticas, por ejemplo, pueden clasificarse por el uso de un único entorno o varios; otro criterio para clasificar las metaheurísticas, es si se han inspirado en la naturaleza o no; y también podemos basarnos en si utilizan mecanismos de memoria o no. En esta ocasión, utilizaremos un criterio en base al tratamiento de las soluciones, utilizando este criterio podemos clasificar las metaheurísticas en tres grandes grupos: de búsqueda global o trayectoriales, poblacionales y constructivas.

- Las *metaheurísticas trayectoriales* utilizan una única solución por cada iteración del algoritmo, y consideran una nueva solución obtenida a partir de la anterior en cada iteración.

Dentro de este grupo, encontramos algoritmos como: el *recocido o enfriamiento simulado* [67, 16], la *búsqueda tabú* [42] o la *búsqueda en entornos variables (Variable Neighborhood Search, VNS)* [75], entre otros.

- Las *metaheurísticas poblacionales* utilizan un conjunto de soluciones, al que denominan población, en cada iteración del algoritmo. El objetivo es realizar una amplia exploración del espacio de búsqueda, mediante la manipulación de dicha población.

Alguno de los ejemplos más representativos son: los *algoritmos genéticos (Genetic Algorithms, GA)*, propuestos en [58], desde su aparición se han desarrollado numerosas variantes y se han aplicado a multitud de problemas, por ejemplo, en [64] se aplican diferentes variantes de los GA al problema de enrutamiento de vehículos y se comparan los resultados con los de otros algoritmos, como la *búsqueda tabú* o los *sistemas de colonias de hormigas*.

Los *algoritmos meméticos (Memetic Algorithms, MA)* [76] son otro ejemplo de metaheurística poblacional. Estos algoritmos han sido aplicados a multitud de problemas y se han desarrollado a lo largo del tiempo. En [79] podemos observar numerosas variantes y aplicaciones de los MA.

Los *algoritmos de enjambres de partículas (Particle Swarm Optimization algorithm, PSO)* [66] también se clasifican dentro de este grupo, en [105] se muestra un análisis exhaustivo de distintas versiones del algoritmo, hibridaciones con otros métodos co-

## 5.1. METAHEURÍSTICAS TRAYECTORIALES

---

mo la *búsqueda tabú* o los *algoritmos genéticos* y aplicaciones a diferentes problemas como problemas de asignación o el problema del viajante.

El algoritmo de *búsqueda dispersa* (*Scatter Search*, SS) fue propuesto en [41, 45], y se basa en la combinación ponderada de una población de soluciones más reducida que en los GA, e integra la combinación de soluciones con procesos de búsqueda local. En [47] encontramos un análisis del SS y una recopilación de los problemas resueltos con el mismo, como puede ser el entrenamiento de *redes neuronales* [65] o el enrutamiento de vehículos [7].

- Las *metaheurísticas constructivas* parten de una solución inicial incompleta y añaden componentes iterativamente hasta obtener la solución deseada.

En este grupo podemos encontrar algoritmos como: los *sistemas de colonias de hormigas* (*Ant Colony Optimization*, ACO) propuestos en [28], los cuales están inspirados en el comportamiento de las hormigas, y suelen aplicarse a problemas de enrutamiento o de asignación. No obstante, la lista no se limita a estos, en [29] encontramos una lista de algunas variantes del algoritmo y de sus aplicaciones a distintos problemas. El algoritmo *Greedy Randomized Adaptive Search Procedures* (GRASP) también es una metaheurística constructiva, fue propuesto en [37] y cuenta con multitud de aplicaciones. En [86] se muestran algunas, junto con una comparativa entre otros algoritmos como la *búsqueda en entornos variables* o la *búsqueda tabú*.

En nuestro caso, nos centraremos en las metaheurísticas trayectoriales debido a que son las que ofrecen los resultados más prometedores para la resolución de este tipo de problemas.

### 5.1. Metaheurísticas trayectoriales

Como hemos mencionado anteriormente, las metaheurísticas trayectoriales utilizan una solución en cada iteración, y consideran una nueva solución obtenida a partir de la anterior en cada iteración. Se centran en la búsqueda probabilística, y tienen como objetivo no quedar atrapadas en óptimos locales. Para ello, se apoyan en distintos recursos dependiendo de qué metaheurística se use. Estas metaheurísticas necesitan un método o

heurística que construya una solución inicial y pueden verse fuertemente afectadas por esta solución inicial, por lo que se debe elegir cuidadosamente.

Para evitar quedar atrapados en óptimos locales, se consideran tres opciones: comenzar de nuevo la búsqueda desde otra solución inicial, modificar la estructura de entornos y permitir movimientos hacia peores soluciones.

Las principales metaheurísticas trayectoriales son, el *recocido simulado*, la *búsqueda en entornos variables* y la *búsqueda tabú*; que describimos a continuación en detalle, siendo las tres metaheurísticas que se han utilizado en esta Tesis Doctoral para resolver los problemas propuestos.

## 5.2. Recocido Simulado (SA)

El *recocido o enfriamiento simulado* (*Simulated Annealing*, SA) está inspirado en la metalurgia y la termodinámica. En la metalurgia, se observa que si se realiza un enfriamiento lento, se origina un sólido de entropía mínima. Si esta idea es aplicada en el ámbito de la optimización, se puede realizar una comparación entre la entropía mínima del sólido y la solución óptima del problema. Esta metaheurística fue introducida por Kirkpatrick, Gelatt y Vecchi [67] y por Cerny [16]. En ambos estudios se aplicó el *recocido simulado* al problema del viajero.

El proceso consiste en generar una solución inicial, y a partir de esta generar otras soluciones de forma iterativa mientras se desciende la temperatura, hasta obtener una solución óptima o próxima a la óptima. Siguiendo con la analogía de la metalurgia, la temperatura del sólido se equipara con la temperatura del proceso de optimización, y la entropía mínima del sólido es equivalente a la solución óptima.

El efecto de la temperatura sobre el algoritmo es esencial. Cuando la temperatura es alta, al inicio de la ejecución se permitirá con una probabilidad alta movernos hacia soluciones peores, evitando quedarnos atrapados en óptimos locales, y a medida que las iteraciones aumentan, la temperatura desciende y la probabilidad de aceptar soluciones peores también disminuye con esta.

Una ventaja del recocido simulado es la facilidad para conseguir un equilibrio entre exploración y explotación, el cual necesitan todas las metaheurísticas para su correcto

## 5.2. RECOCIDO SIMULADO (SA)

---

funcionamiento. Esto lo consigue con una temperatura inicial elevada, favoreciendo la exploración y evitando quedar estancado en óptimos locales y su descenso progresivo permitiendo al algoritmo una mayor explotación en las fases avanzadas, favoreciendo así la búsqueda de los óptimos globales.

Aarts y Korst [2] demostraron la convergencia del algoritmo al óptimo global según la temperatura se aproxima a cero con probabilidad uno, con un estudio basado en la teoría de las cadenas de Markov. No obstante, para esto se necesita un enfriamiento infinitamente lento, lo que implica un tiempo infinito, y en la práctica, no se puede garantizar la obtención de una solución óptima.

La creación de la solución inicial es un tema muy relevante del que puede depender que el algoritmo converja o no. En algunas ocasiones se utiliza un *recocido simulado multicomienzo*, es decir, se genera más de una solución inicial y se aplica el *recocido simulado* a estas. Aplicar un algoritmo multicomienzo no tiene por qué suponer un tiempo adicional, ya que se puede paralelizar el trabajo para cada una de las soluciones iniciales si se cuentan con los medios adecuados.

Otro punto que necesitamos definir es la búsqueda de soluciones del entorno de una solución a partir de movimientos. Un *movimiento* es un cambio en una solución que genera otra solución (del entorno). Por ello, las soluciones del entorno de una solución serán todas las soluciones a las que se pueden llegar realizando un único movimiento. Es importante que los movimientos ofrezcan soluciones cercanas entre sí, es decir, de características similares. En cualquier caso, la definición de soluciones del entorno y los movimientos son un factor clave para la convergencia del algoritmo.

El *recocido simulado* tiene una serie de parámetros indispensables para que funcione correctamente, y es muy dependiente de ellos. Aquí es donde encontramos una de las principales desventajas del *recocido simulado*. La estimación de dichos parámetros es compleja ya que muchos de estos se tienen que estimar teniendo en cuenta las relaciones existentes entre ellos. Por esta razón, la estimación de los parámetros no puede realizarse de forma independiente. También se necesita tomar una serie de decisiones, como la elección de una condición de parada o la definición de un entorno de soluciones, lo cual afectará al funcionamiento de este.

Existen ciertas técnicas que permiten realizar una estimación de alguno de estos pará-

metros para calcular cuáles serían los valores idóneos, pero en otras ocasiones, es necesario estimarlos mediante prueba y error. Por ejemplo, en [61, 1, 102, 91] se muestran varios métodos y aproximaciones para asignar valores a estos parámetros lo mejor posible, muchas de las cuales mencionaremos posteriormente.

Existen numerosas versiones y variantes del *recocido simulado* que se han ido proponeiendo a lo largo de los años [60, 31]. Podemos dividir las distintas versiones entre las que solo tienen un único objetivo (uniobjetivo), o las que tienen varios objetivos generalmente en conflicto o contradictorios (multiobjetivo).

### 5.2.1. Recocido Simulado Uniobjetivo

El *recocido simulado uniobjetivo* es la versión más básica del *recocido simulado* (pseudocódigo en el Algoritmo 1), y partiendo de esta, se pueden añadir modificaciones y mejoras que pueden funcionar mejor o peor dependiendo del problema a resolver.

---

#### Algoritmo 1 Recocido o Enfriamiento Simulado (SA) para un problema de maximización

**Entrada:**  $s = s_0, s_0 \in S$  {Solución Inicial} siendo  $S$  el espacio de soluciones

**Salida:** Mejor solución encontrada.

```
1:  $T_1 = t < 0$  {Elegir una temperatura inicial}  
    $0 < \alpha < 1$  {Elegir una tasa de enfriamiento}  
2: repetir  
3:    $s' = \text{SoluciónAleatoria}(E(s))$  {Escoge una solución aleatoria  $s'$  del entorno de  $s$ }  
4:   si ( $f(s') < f^*$ ) {Siendo  $f()$  la función objetivo (< o > dependerá, si minimizamos  
o maximizamos)} entonces  
5:      $s = s'$   
      $f^* = f(s')$   
6:   si no  
7:     Generar  $u = U(0, 1)$  y  $p = e^{\frac{-(f(s') - f(s))}{T_i}}$   
8:     si ( $p > u$ ) entonces  
9:        $s = s'$   
10:    fin si  
11:  fin si  
12:   $T_{i+1} = \alpha T_i$   
13: hasta que Criterio de parada
```

---

## 5.2. RECOCIDO SIMULADO (SA)

---

### Generación de soluciones iniciales

En el sentido estricto, la generación de soluciones iniciales no pertenece al *recocido simulado*, aunque sí es cierto que es un factor que puede afectar en gran medida al funcionamiento del algoritmo. Como la mayoría de los parámetros y funciones del SA, no existe una única regla que garantice el mejor funcionamiento, si no que dependerá del problema en cuestión que estemos resolviendo.

Puede utilizarse un algoritmo completamente aleatorio para generar las soluciones iniciales, lo cual facilita la exploración del espacio de búsqueda completo sin perder soluciones. En otros casos, es recomendable utilizar un algoritmo voraz (*greedy*) que nos proporcione una solución determinista, pero la ventaja es que la solución proporcionada tendrá un mejor valor objetivo desde el inicio del recocido.

Este proceso pese a ser parte independiente del recocido, tiene una gran importancia en el proceso de convergencia, pudiendo ser la diferencia entre encontrar una solución óptima o no.

### Función objetivo

La *función objetivo* es una función utilizada para evaluar las soluciones, y al resultado obtenido de esta función se le denomina *valor objetivo*.

Definir una buena función objetivo es de suma importancia, ya que si la función objetivo no es adecuada, no proporciona suficiente información para poder mejorar el problema. Es altamente recomendable que la función objetivo tenga las siguientes características:

1. Que la función objetivo no esté formada por valles. Si una función objetivo tiene valles, quiere decir que su valor es el mismo para una solución del entorno que para otra. Esto no es recomendable ya que, además de no aportar ninguna información sobre cuál elegir, el algoritmo no tiene información suficiente para poder avanzar en una dirección u otra.
2. Es preferible que la función objetivo sea continua. Una función objetivo discontinua dificulta la exploración del espacio de soluciones. Este problema no es sencillo de resolver, pero se pueden mitigar sus inconvenientes modificando los entornos

de soluciones, la representación de las soluciones o permitiendo movimientos hacia soluciones infactibles.

3. Es preferible que los movimientos generen soluciones similares y estas tengan un valor aproximado, es decir, los saltos grandes en la función objetivo no son recomendables. Un movimiento debe generar una solución parecida a la actual y esta, por lo tanto, al generar su valor, no debe variar en exceso. También es importante que un cambio pequeño en una solución genere un pequeño cambio en el valor objetivo de esta. En algunos casos están justificados movimientos que provoquen grandes cambios para intentar escapar de óptimos locales.

### Temperatura inicial

En [10] se recomienda que la temperatura inicial sea iniciada con un valor suficientemente grande para que la probabilidad de aceptación de la transición hacia soluciones peores que la actual sea igual a un valor específico (normalmente cercano al 80-90 %).

Una temperatura demasiado alta puede aumentar los tiempos de convergencia o incluso provocar un mal funcionamiento del algoritmo. Por el contrario, una temperatura baja provoca una convergencia prematura que se puede traducir en un aumento de las posibilidades de que el algoritmo se estanque en un óptimo local.

En [10] se propone una fórmula recursiva, que finaliza cuando el valor  $X(T_n)$  se acerca al valor de  $X_0$ , obteniendo como temperatura inicial  $T_n$ . La fórmula propuesta es:

$$T_{n+1} = T_n \left( \frac{\ln(X(T_n))}{\ln(X_0)} \right)^{1/p}, \quad (5.1)$$

siendo  $p$  un número real mayor o igual a 1. Aún por definir encontramos  $X_0$ , al cual se asigna el valor de la probabilidad de aceptación deseada, y por último  $X(T_n)$  el cual se define en la siguiente fórmula:

$$X(T) = \frac{\sum_{t \in S} \exp(-E_{j_t}/T)}{\sum_{t \in S} \exp(-E_{i_t}/T)}, \quad (5.2)$$

donde se utiliza un conjunto  $S$  de transiciones positivas generado previamente, siendo  $i$  la solución de partida y  $j \in N(i)$  la nueva solución propuesta del entorno  $N(i)$  y donde  $E_{j_t}$

## 5.2. RECOCIDO SIMULADO (SA)

---

y  $E_{i_t}$  son los valores en la función objetivo de las soluciones generadas en la iteración  $t$ .

Otra forma de estimar la temperatura inicial ( $T_0$ ) propuesta en [61, 62] es a partir de porcentaje de aceptación inicial ( $X_0$ ), es decir, el porcentaje de soluciones que se debe aceptar durante las primeras iteraciones del algoritmo. Además, se usa la media de incrementos (para un problema de maximización, en caso de ser un problema de minimización serán los decrementos) de la función objetivo ( $\Delta$ ), calculada solo con los valores positivos, es decir, cuando se obtiene una solución mejor; usando la ecuación

$$T_0 = -\Delta/\ln(X_0). \quad (5.3)$$

Es importante tener en cuenta que estos datos tienen que ser independientes de la solución inicial, por lo que es necesario depender de un generador aleatorio de soluciones para el uso correcto de la fórmula.

Existen otros métodos que necesitan el uso de generadores aleatorios de soluciones iniciales, estos son expuestos en [1, 102] utilizando otras fórmulas para el cálculo de la temperatura inicial. Estos métodos, tienen otros inconvenientes ya que dependen del parámetro de porcentaje de aceptación  $X_0$ , que se estima entre el 90 - 100 % pero del cual no se conoce un valor exacto.

En [67] se propone la fórmula

$$T_0 = \delta E_{max}, \quad (5.4)$$

para obtener la temperatura inicial, donde  $\delta E_{max}$  es la diferencia máxima entre los valores de la función objetivo de cualquier par de soluciones vecinas.

### Función de enfriamiento

La función de *enfriamiento* es una función que permite disminuir la temperatura inicial del algoritmo mientras se realizan las iteraciones sobre este. Esto se traduce en reducir la probabilidad de elegir soluciones que empeoran el valor objetivo y, por tanto, quedarnos con las mejores. La función más conocida es la de *enfriamiento proporcional*, sugerida por Kirkpatrick en [67]. A esta función se le denomina *enfriamiento geométrico*, siendo el valor de  $k$  la iteración actual

$$T_k = \alpha T_{k-1}, \quad (5.5)$$

y  $0 < \alpha < 1$  la *tasa o ratio de enfriamiento*.

Generalmente, el ratio de enfriamiento se sitúa en  $\alpha = 0,95$  según la referencia de Hajek [49], pero se proponen otros métodos para calcularlo como

$$\alpha = (T_M/T_0)^{1/(M-1)}, \quad (5.6)$$

siendo  $M$  el número de iteraciones máximo.

Otro ejemplo de función de enfriamiento, aún más sencillo si cabe, lo encontramos en la función de enfriamiento lineal

$$T_k = T_{k-1} - \alpha. \quad (5.7)$$

En [2], los autores se basan en la condición asintótica de convergencia del *recocido simulado*, pero ya que en la práctica los tiempos de cómputo no serían aceptables en ningún caso, añaden un factor de aceleración para el enfriamiento. La función propuesta utiliza un  $\alpha \geq 1$ ,

$$T_k = \frac{T_0}{1 + \alpha \log(1 + k)}. \quad (5.8)$$

También encontramos en [49] la función

$$T_k = \frac{c}{\log(1 + k)}, \quad (5.9)$$

donde la condición de convergencia es que el valor de  $c$  sea mayor a la profundidad máxima necesaria para salir de cualquier óptimo local.

Podemos encontrar en la literatura multitud de variaciones sobre esta función las cuales englobamos dentro del grupo de las *funciones multiplicativas*, tales como,

$$T_k = \frac{T_0}{1 + \alpha k}, \quad (5.10)$$

## 5.2. RECOCIDO SIMULADO (SA)

---

$$\text{y } T_k = \frac{T_0}{1 + \alpha k^2}. \quad (5.11)$$

También existen otros ejemplos más conocidos como el criterio de Boltzmann

$$T_k = \frac{T_0}{1 + \log(k)}, \quad (5.12)$$

y el esquema de Cauchy

$$T_k = \frac{T_0}{1 + k}. \quad (5.13)$$

En otro grupo, al cual podemos denominar *aditivas*, encontramos otras funciones como las siguientes, que representan un descenso lineal de la temperatura o la función cuadrática, entre otras:

$$T_k = T_M + (T_0 - T_M) \left( \frac{M - k}{M} \right), \quad (5.14)$$

$$T_k = T_M + (T_0 - T_M) \left( \frac{M - k}{M} \right)^2. \quad (5.15)$$

El problema de estas funciones es que incorporan la temperatura final ( $T_M$ ) del algoritmo, la cual en muchas ocasiones no se conoce.

Hasta ahora solo hemos visto funciones de enfriamiento rígidas o inadaptables, aunque también existen funciones que incorporan los valores objetivos de las soluciones en la misma. Una de estas funciones la encontramos en [70], la cual permite adaptar la temperatura a la evolución de las soluciones:

$$T_k = \alpha (f(s_k) - f^* + T_{k-1})^g, \alpha > 0, g > 0. \quad (5.16)$$

En [4] se propone la *técnica de recalentado*, consistente en el uso de dos parámetros  $\alpha$  y  $\beta$  (ambos menores que 1) para disminuir y aumentar la temperatura respectivamente con las fórmulas mostradas en la expresión 5.17. La fase de enfriamiento termina cuando no existe una mejora del valor objetivo de la solución en un pequeño número de iteraciones, mientras que la fase de calentamiento finaliza cuando el valor objetivo se modifica (ya sea empeorando o mejorando el actual). También tiene la posibilidad de quedarse estancado en

ciclos, por lo que los autores exponen un criterio de parada para evitar el comportamiento cíclico.

$$T_{k+1} = \begin{cases} \alpha T_k, & \text{si Enfriar} \\ T_k/\beta, & \text{si Calentar} \end{cases}. \quad (5.17)$$

Existen otros estudios en la literatura, los cuales repasan las distintas funciones de enfriamiento, tales como [19, 4].

Se han realizado algunos estudios comparando todo el conjunto de funciones de enfriamiento publicadas sobre distintos problemas [81, 73, 71, 31].

En [31] muestran que acorde a sus resultados, la elección de una función de enfriamiento frente a otra no presenta una ventaja significativa. En cambio, es importante no desperdiciar tiempo de cómputo en fases donde la temperatura es demasiado alta o baja. En [81] miden la calidad de las funciones de enfriamiento en base a la entropía producida, y comparando varias funciones, llegan a la conclusión de que el rendimiento de la función de enfriado logarítmica produce una entropía muy elevada en comparación con la lineal, exponencial, TDS1 y TDS2 (TDS1 y TDS2 son dos funciones propuestas en el artículo, basadas en la constante de la velocidad termodinámica).

En la Tabla 5.1 mostramos un resumen de las funciones de enfriamiento.

### Número de iteraciones en el que se mantiene constante la temperatura ( $L$ )

Este parámetro, junto con la temperatura inicial y la función de enfriamiento, son los que establecen el desarrollo del algoritmo, permitiendo una ejecución más rápida o más lenta.

El número de iteraciones debe ser proporcional al tamaño del entorno de soluciones (cadenas adaptativas), es decir, al número de soluciones del entorno de una solución, aunque también se puede estimar un valor fijo (cadenas estáticas).

Una mejora que se aplica con frecuencia en estos casos es el *cutoff*, consistente en introducir un contador del número de soluciones aceptadas y, cuando este supere un umbral, la temperatura puede descender antes de lo establecido. Gracias a esto se pueden mejorar los tiempos, sobre todo con el uso de temperaturas muy elevadas.

## 5.2. RECOCIDO SIMULADO (SA)

---

Tabla 5.1: Comparativa de las distintas funciones de enfriamiento

Función		Características	
		Ventajas	Desventajas
<b>Lineal</b>	$T_k = T_{k-1} - \alpha$	Sencillez	Muy baja exploración
<b>Geométrica</b>	$T_k = \alpha T_{k-1}$	Más estudiada y utilizada	-
<b>Logarítmica 1</b>	$T_k = \frac{c}{\log(1+k)}$	Alta exploración	Possible estancamiento
<b>Logarítmica 2</b>	$T_k = \frac{T_0}{1+\alpha \log(1+k)}$	Alta exploración	Possible estancamiento
<b>Multiplicativa 1</b>	$T_k = \frac{T_0}{1+\alpha k}$	Rápida	Baja exploración
<b>Multiplicativa 2</b>	$T_k = \frac{T_0}{1+\alpha k^2}$	Rápida	Baja exploración
<b>Cauchy</b>	$T_k = \frac{T_0}{1+k}$	Rápida	Baja exploración
<b>Boltzmann</b>	$T_k = \frac{T_0}{1+\log(k)}$	Alta exploración	Possible estancamiento
<b>Aditiva 1</b>	$T_k = T_M + (T_0 - T_M) \left(\frac{M-k}{M}\right)^g$	Alta exploración	Lenta y necesidad de estimar $M$
<b>Aditiva 2</b>	$T_k = T_M + (T_0 - T_M) \left(\frac{M-k}{M}\right)^2$	Equilibrio exploración explotación	Necesidad de estimar $M$
<b>Adaptativa 1</b>	$T_k = \alpha (f(s_k) - f^* + T_{k-1})^g$	Alta exploración	Elevado tiempo de cómputo
<b>Adaptativa 2</b>	$T_{k+1} = \begin{cases} \alpha T_k, & \text{si Enfriar} \\ T_k / \beta, & \text{si Calentar} \end{cases}$	Alta exploración	Possible comportamiento cíclico

## Soluciones del entorno y movimientos

Las soluciones de un entorno de una solución  $x$  son todas las soluciones que se pueden obtener aplicando un movimiento a dicha solución. Un *movimiento* es el cambio que se realiza en una iteración sobre una solución para formar otra de su entorno. Los movimientos son conceptos muy genéricos, por lo que, a continuación, veremos un ejemplo para un mejor entendimiento.

Imaginemos un problema en el que tenemos que ordenar unas fichas, pero solo podemos mover la primera con una de las siguientes. Una definición de movimiento sería el intercambio entre la primera ficha con alguna de las anteriores. A partir de la solución  $(2,3,5,1)$ , podríamos generar las soluciones del entorno aplicando los movimientos, siendo estas  $(3,2,5,1)$ ;  $(5,3,2,1)$  y  $(1,3,5,2)$ . Dependiendo del problema, tendremos distintos tipos de movimientos y soluciones del entorno.

En la mayoría de casos, la elección del movimiento no se realiza de forma aleatoria, y se intenta elegir movimientos que tiendan a soluciones más eficientes. Existen varias

propuestas en esta área, pero gran parte de las mismas tienen una fuerte dependencia con el problema en concreto que se quiera resolver. Se necesita conocer el problema y aplicar conocimiento del dominio para que, de esta forma, los movimientos sean eficientes.

### Función de aceptación

La función de aceptación más usada para saber si el recocido simulado acepta la solución nueva o se queda con la actual, es el *criterio de metrópolis*. El criterio de metrópolis [74], es una regla de aceptación consistente en lo siguiente: si la nueva solución mejora el valor objetivo, esta es aceptada; en caso contrario, se genera un número aleatorio  $r$  entre 0 y 1, y se rechaza la nueva solución salvo que  $e^{\frac{-\Delta}{T_i}} > r$ . La distribución descrita se denomina *distribución de Boltzman*, que viene dada por la siguiente expresión:

$$e^{\frac{-(f(x') - f(x))}{T_i}}. \quad (5.18)$$

El uso de tablas de valores aproximados de la función exponencial en lugar de usar la distribución de Boltzman supone un ahorro en tiempo considerable.

No obstante, aún existiendo otras distribuciones o reglas de aceptación [11], en la mayoría de casos la distribución de Boltzman es la más rápida y efectiva.

Por ejemplo, en [24] se propone el *algoritmo de Creutz* o *enfriamiento microcanónico*. Este algoritmo se basa en la idea de que la energía se mantiene constante ( $E_{total} = E_p + E_c$ ). La energía potencial se considera el valor de la función objetivo, mientras que la energía cinética es usada como temperatura en el SA. El algoritmo se basa en aceptar todas las soluciones que disminuyan la energía potencial y solo acepta las soluciones que aumenten la energía potencial cuando el aumento es menor que la energía cinética,  $\Delta E < E_c$ .

Otra de estas funciones de aceptación la encontramos en [30]. En este artículo presentan el *método del umbral de aceptación*. La principal característica de este método reside en que las soluciones que empeoran el valor objetivo son aceptadas siempre que la diferencia entre los valores  $\Delta f$  no superen un umbral  $T$ , el cual desciende progresivamente.

## 5.2. RECOCIDO SIMULADO (SA)

---

### Condición de parada

Se han propuesto numerosas condiciones de parada, como el número de iteraciones, número de iteraciones sin mejora de la solución, tiempo de ejecución, temperatura final, etc.

En [61] se propone otra regla de parada conocida como *la regla de parada de Johnson*. Esta regla consiste en que cada vez que descienda la temperatura, tenemos que observar el número de iteraciones en las que se ha mejorado y, si este no llega a un mínimo preestablecido, se sumará uno a un contador. Cuando este contador llegue al valor elegido se ejecutará la condición de parada.

Otra regla de parada utilizada usualmente es el *porcentaje de mejora*, es decir, si cada cierto número de iteraciones la función objetivo no ha mejorado un porcentaje establecido, la búsqueda finaliza.

En [91] se propone el uso de cuatro reglas de parada no excluyentes que en el caso que se cumpla al menos una de estas, el algoritmo llega a su fin. Las condiciones de parada son las siguientes:

- El algoritmo se detiene cuando no existan mejoras de la solución durante las últimas cuatro etapas de temperatura (lo que es equivalente a  $4L$  iteraciones).
- El algoritmo se detiene cuando la temperatura llegue a un umbral estipulado.
- El algoritmo se detiene cuando se supera un umbral máximo de iteraciones multiplicado por las dimensiones del problema  $k \geq U \times n$ , siendo  $k$  la iteración actual,  $U$  el umbral y  $n$  un valor representativo del espacio de búsqueda del problema.
- El algoritmo se detiene cuando el mejor valor objetivo encontrado hasta el momento no se mejora un porcentaje establecido durante  $L$  iteraciones. Esta función es ligeramente más compleja que la anterior debido a que tiene en cuenta los valores máximos y mínimos de las variables. Solo se puede aplicar cuando estamos resolviendo problemas con variables continuas.

### 5.2.2. Recocido Simulado Multiobjetivo

En el año 1983, Kirkpatrick presentó la metaheurística, pero no fue hasta la década de los 90 donde pudimos encontrar las primeras publicaciones de algoritmos MOSA (*Multi-Objective Simulated Annealing*).

En [90] podemos ver cómo se aplica a problemas multiobjetivo. Para la adaptación del recocido simulado a problemas multiobjetivo, se propone modificar el criterio de aceptación en base a las dominancias de las soluciones, es decir, guardamos las soluciones formando un frente de Pareto siempre y cuando una solución no esté dominada por las soluciones pertenecientes al conjunto de Pareto.

Antes de definir el concepto de frente de Pareto, debemos entender el concepto de dominancia. Este término indica que una solución domina a otra cuando ninguno de los criterios de la solución dominada es mejor al de la solución dominante.

El *frente o frontera de Pareto* (también denominado *conjunto eficiente*) está formado por soluciones *no dominadas, eficientes, o pareto óptimas*, que son soluciones tales que no existe otra solución que tome un valor mejor en algún objetivo sin causar un empeoramiento simultáneo en al menos otro objetivo. En la Figura 5.1 tenemos un ejemplo de un frente de Pareto para un problema de minimización con dos objetivos,  $f_1$  y  $f_2$ , donde podemos apreciar que solo las soluciones no dominadas son las que forman el frente de Pareto.

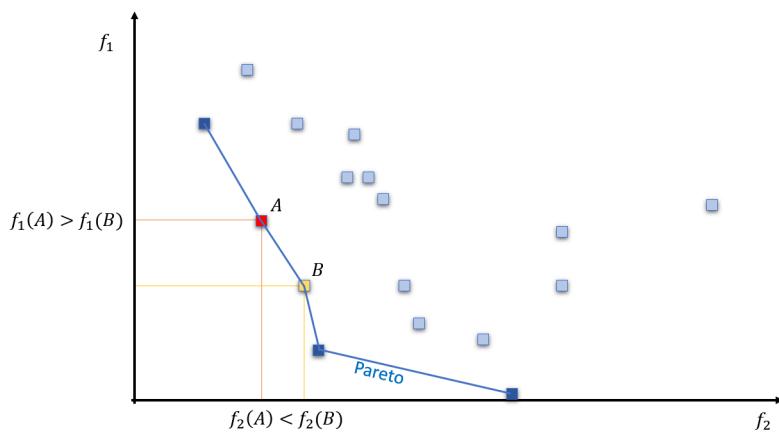


Figura 5.1: Ejemplo de frente de Pareto

## 5.2. RECOCIDO SIMULADO (SA)

---

### Función de aceptación

La aceptación de soluciones en el *recocido simulado multiobjetivo* es más compleja que en comparación con el *recocido uniobjetivo*. Existen numerosas investigaciones [78, 90] que combinan todos los objetivos en una suma ponderada para la obtención de un valor objetivo:

$$f(x) = \sum_{k=0}^D f_k(x) \times w_k, \quad (5.19)$$

siendo  $D$  el número de objetivos,  $w_k$  la ponderación para el objetivo  $k$  y  $f_k$  la función objetivo.

En este caso podemos usar la función básica de aceptación del *recocido simulado uniobjetivo*. No obstante, con este enfoque encontramos algunos inconvenientes: no existe un método claro para la elección de los pesos, y en los casos en el que el frente de Pareto no es convexo, ciertas zonas del frente de Pareto resultan inaccesibles con determinados pesos [26].

Si queremos evitar el uso de ponderaciones, la función más sencilla se basa en aceptar los individuos no dominados por la solución anterior y guardarla en caso de que pertenezca al conjunto de Pareto. En caso contrario, es decir, cuando esté dominada por la solución anterior, la solución se aceptará con cierta probabilidad. Una de las funciones de aceptación más usadas (para problemas de minimización) es

$$e^{-\frac{\sum_{k=0}^D (f_k(x') - f_k(x))}{T_i}}, \quad (5.20)$$

siendo  $D$  el número de objetivos y  $f_k$  la función objetivo  $k$ -ésima.

Otra variante del algoritmo se encuentra en [8]. Esta variante se denomina AMOSA (*Archive Multi-Objective Simulated Annealing*). La principal novedad se encuentra en el uso de un *archivo* (de ahí su nombre) en el que se van a introducir y eliminar soluciones no dominadas y dominadas, respectivamente, según avanza el proceso de búsqueda y se vayan analizando nuevas soluciones, de tal forma que al final del proceso de búsqueda en dicho *archivo* tendremos una aproximación del *conjunto eficiente* o *frontera Pareto*. Cuando el número de soluciones almacenadas llega a un umbral determinado  $HL$ , se utiliza un

proceso de *clustering* para reducir el número hasta un umbral inferior  $SL$  eliminando las soluciones dominadas. A continuación, explicamos el criterio de aceptación utilizado por el algoritmo.

1. Caso 1: La solución actual domina a la nueva solución. En este caso, se acepta la solución con una probabilidad  $p$ , la cual viene dada por la expresión

$$p = \frac{1}{1 + e^{\Delta dom_{avg} T_i}}, \quad (5.21)$$

donde

$$\Delta dom_{avg} = ((\sum_{s=0}^S (\Delta dom_{x_s, x'})) + \Delta dom_{x, x'}) / (S + 1), \quad (5.22)$$

y

$$\Delta dom_{x, y} = \prod_{k=0}^M \left( \frac{|f_k(x) - f_k(y)|}{R_k} \right), \quad (5.23)$$

siendo  $x, y$  dos soluciones distintas y  $R_k$  el rango de valores del objetivo  $k$ .

2. Caso 2: La nueva solución y la solución actual no están dominadas respectivamente.
  - a) La nueva solución no está dominada por ninguna de las soluciones almacenadas en el *archivo*. En este caso, la solución nueva sustituye a la actual y se incluye en el *archivo*.
  - b) La nueva solución domina  $k$  soluciones de las almacenadas en el *archivo*. En este caso, la solución nueva también sustituye a la actual y se incluye en el *archivo*, eliminando las  $k$  soluciones a las que domina en el *archivo*.
  - c) La nueva solución está dominada por  $k$  soluciones del *archivo*. En este caso, se acepta la nueva solución con una probabilidad  $p$  dada por la expresión

$$p = \frac{1}{1 + e^{\Delta dom_{avg} \times T_i}}, \quad (5.24)$$

donde  $\Delta dom_{avg} = (\sum_{s=0}^S (\Delta dom_{x_s, x'})) / S$ .

3. Caso 3: La nueva solución domina la solución actual.

### 5.3. BÚSQUEDA EN ENTORNOS VARIABLES (VNS)

---

- a) La nueva solución no está dominada por ninguna de las almacenadas en el *archivo*. En este caso, la solución nueva sustituye a la actual y se incluye en el *archivo*.
- b) La nueva solución domina  $k$  soluciones de las almacenadas en el *archivo*. En este caso, la solución nueva también sustituye a la actual y se incluye en el *archivo*, eliminando las  $k$  soluciones a las que domina en el *archivo*.
- c) La nueva solución está dominada por  $k$  soluciones del *archivo*, por lo que no es incluida en el *archivo*. En este caso, se acepta la nueva solución con una probabilidad  $p$  dada por la expresión

$$p = \frac{1}{1 + e^{-\Delta dom_{min}}}, \quad (5.25)$$

donde  $\Delta dom_{min} = \min_S(\Delta dom_{x_s, x'})$ .

## 5.3. Búsqueda en Entornos Variables (VNS)

La idea básica de la *búsqueda en entornos variables* (VNS, *Variable Neighborhood Search*), propuesta en [75], se basa en la exploración de un conjunto de sucesivos entornos para obtener una mejor solución. El VNS explora de forma aleatoria o sistemática un conjunto de distintos entornos para obtener un conjunto de varios óptimos locales. El VNS está basado en la idea de modificar el entorno, permite movernos por distintos óptimos locales y facilita alcanzar el óptimo global.

Existen numerosas variantes de la búsqueda básica, algunas variantes modifican el orden de elección de los distintos entornos, otras se basan en la aceptación de soluciones peores a la actual en base a una medida de distancia, favoreciendo así la exploración. Otras variantes más complejas son el *VNS reducido* [50], el *VNS con descomposición*[53] o el *VNS sesgado*, entre otras.

### 5.3.1. Búsqueda en Entornos Variables Básica

La primera variante es el VNS básico del cual tenemos el pseudocódigo en el Algoritmo 2. Esta búsqueda comienza generando una solución aleatoria en el primer entorno. A partir

de esta solución, se ejecuta una búsqueda local en este entorno para encontrar el óptimo. Una vez obtenemos la nueva solución, se compara con la mejor solución actual. Si la nueva solución es mejor, esta es escogida, y se comienza de nuevo con el primer entorno; en caso contrario, se modifica el entorno actual por el siguiente y generamos una nueva solución aleatoria a partir de la mejor solución actual para comenzar la búsqueda local en el nuevo entorno. El motivo de generar una nueva solución aleatoria cada vez que cambiamos de entorno es evitar el estancamiento del algoritmo debido al uso de reglas determinísticas en la búsqueda local.

---

**Algoritmo 2** Búsqueda en Entornos Variables Básica (VNS) para problemas de optimización de minimización

---

**Entrada:** Conjunto de los distintos entorno  $N_k$  for  $k = 1, \dots, k_{max}$

$s = s_0$  {Solución Inicial}

**Salida:** Mejor solución encontrada.

```
1: repetir
2:    $k = 1$ 
3:   repetir
4:      $s' = \text{SoluciónAleatoria}(N_k(s))$  {Escoge una solución aleatoria  $s'$  del entorno  $N_k(s)$ }
       $s'' = \text{BúsquedaLocal}(N_k(s'))$  {Ejecuta el algoritmo de búsqueda local sobre el entorno  $N_k(s')$  de  $s'$ }
5:     si ( $f(s'') < f(s)$ ) entonces
6:        $s = s''$ 
7:        $k = 1$ 
8:     si no
9:        $k = k + 1$ 
10:    fin si
11: hasta que  $k = k_{max}$ 
12: hasta que Criterio de parada
```

---

### 5.3.2. Búsqueda en Entornos Variables Descendente

La *búsqueda en entornos variables descendente* se propuso en [57]. Consiste en realizar una búsqueda voraz en los distintos entornos. El pseudocódigo de esta variante se encuentra en el Algoritmo 3. En primer lugar, se escoge el orden de los distintos entornos y se realiza una búsqueda local con el primero. Si el óptimo local encontrado es mejor que la mejor solución actual, se repite la búsqueda con el primer entorno, y en caso contrario, pasamos al siguiente. Con esta búsqueda, la probabilidad de alcanzar un óptimo global es

### 5.3. BÚSQUEDA EN ENTORNOS VARIABLES (VNS)

---

mayor cuantos más entornos se tengan, debido a que la solución será un óptimo local de cada uno de los respectivos entornos.

---

**Algoritmo 3** Búsqueda en Entornos Variables Descendente (VND) para problemas de optimización de minimización

---

**Entrada:** Conjunto de los distintos entornos  $N_k$  for  $k = 1, \dots, k_{max}$   
 $s = s_0$  {Solución Inicial}

**Salida:** Mejor solución encontrada.

```
1: repetir
2:    $k = 1$ 
3:   repetir
4:      $s' = \text{BúsquedaVoraz}(N_k(s))$  {Ejecuta el algoritmo de búsqueda voraz sobre el
       entorno  $N_k(s)$  de  $s$ }
5:     si ( $f(s') < f(s)$ ) entonces
6:        $s = s'$ 
7:        $k = 1$ 
8:     si no
9:        $k = k + 1$ 
10:    fin si
11: hasta que  $k = k_{max}$ 
12: hasta que Criterio de parada
```

---

#### 5.3.3. Búsqueda en Entornos Variables Reducida

La principal diferencia de la *búsqueda en entornos variables reducida* [50] con la anterior se debe a que la búsqueda descendente se centra en la intensificación o explotación, mientras que la búsqueda reducida introduce una mayor aleatoriedad permitiendo una mayor exploración.

Esta variante se basa en el VNS básico, la principal diferencia es que no se realiza una búsqueda local iterativa, sino que busca en un entorno de forma aleatoria un número máximo de iteraciones entre dos mejoras. Si se encuentra una solución mejor, se escoge esta. En caso contrario, busca en el siguiente entorno y repite la búsqueda aleatoria. Tenemos el pseudocódigo de este método en el Algoritmo 4.

Como criterio o condición de parada, generalmente se utiliza un número máximo de iteraciones entre dos mejoras.

---

**Algoritmo 4** Búsqueda en Entornos Variables Reducida (RVNS) para problemas de optimización de minimización

---

**Entrada:** Conjunto de los distintos entornos  $N_k$  for  $k = 1, \dots, k_{max}$

$s = s_0$  {Solución Inicial}

**Salida:** Mejor solución encontrada.

```
1: repetir
2:    $k = 1$ 
3:   repetir
4:     repetir
5:        $s' = \text{SoluciónAleatoria}(N_k(s))$  {Escoge una solución aleatoria  $s'$  del entorno  $N_k(s)$ }
6:     hasta que  $f(s') < f(s) \parallel$  n° máximo iteraciones
7:     si ( $f(s') < f(s)$ ) entonces
8:        $s = s'$ 
9:        $k = 1$ 
10:      si no
11:         $k = k + 1$ 
12:      fin si
13:    hasta que  $k = k_{max}$ 
14:  hasta que Criterio de parada
```

---

### 5.3.4. Búsqueda en Entornos Variables con Descomposición

La versión del VNS expuesta en [53], se centra en descomponer el problema en un subproblema más pequeño. Encontramos el pseudocódigo en Algoritmo 5, el cual es muy similar al del VNS básico, aunque la principal diferencia la encontramos al generar una solución aleatoria a partir de la anterior. En ese momento, se fijan las características distintas entre ambas soluciones  $y = s'/s$ , y en el proceso de búsqueda local únicamente se optimizan las características que se encuentran en el individuo aleatorio y no en el anterior  $s'' = s/s' \cup y'$ , es decir, se optimizan únicamente la parte que difiere de la solución anterior con respecto a la nueva.

La principal ventaja reside en la posibilidad de dividir el espacio de búsqueda en un subespacio más pequeño, lo cual permite una búsqueda local más rápida.

### 5.3.5. Búsqueda en Entornos Variables Sesgada

El algoritmo básico del VNS hace mención a la posibilidad de aceptar soluciones peores a la mejor solución hasta el momento, en base a la distancia de la solución.

### 5.3. BÚSQUEDA EN ENTORNOS VARIABLES (VNS)

---

**Algoritmo 5** Búsqueda en Entornos Variables con Descomposición (VNDS) para problemas de optimización de minimización

---

**Entrada:** Conjunto de los distintos entornos  $N_k$  for  $k = 1, \dots, k_{max}$

$s = s_0$  {Solución Inicial}

**Salida:** Mejor solución encontrada.

```
1: repetir
2:    $k = 1$ 
3:   repetir
4:      $s' = \text{SoluciónAleatoria}(N_k(s))$  {Escoge una solución aleatoria  $s'$  del entorno  $N_k(s)$ }
5:      $s'' = \text{BúsquedaLocal}(N_k(s'))$  {Ejecuta el algoritmo de búsqueda parcial sobre un subconjunto del entorno  $N_k(s')$  de  $s'$ }
```

5: **si** ( $f(s'') < f(s)$ ) **entonces**

```
6:        $s = s''$ 
7:        $k = 1$ 
8:     si no
9:        $k = k + 1$ 
10:      fin si
11:    hasta que  $k = k_{max}$ 
12:  hasta que Criterio de parada
```

---

La variante del VNS, llamada *búsqueda en entornos variables sesgada (skewed VNS, SVNS)* [51], utiliza una función de similitud entre la solución actual y la nueva solución generada a partir de esta, si las soluciones no se parecen, es decir, están alejadas, existe la posibilidad de sustituir la solución actual por esta nueva solución incluso empeorando el valor de la función objetivo, pero solo en el caso de que no lo empeore en exceso.

En algunos espacios de búsqueda encontramos grandes valles que dificultan el avance de la mayoría de algoritmos, como el VNS básico. No obstante, el SVNS, del cual encontramos el pseudocódigo en Algoritmo 6, resulta muy atractivo para la resolución de estos problemas.

#### 5.3.6. Búsqueda en Entornos Variables Paralela

Esta variante del VNS propuesta en [52], es distinta al resto, ya que influyen factores de procesado y CPUs. Está claro que el VNS tiene diferentes entornos y es un algoritmo que permite una parallelización sencilla. Asimismo, existen numerosas búsquedas locales que permiten parallelización e incluso, puede generarse más de una solución aleatoria y generar múltiples búsquedas locales con distintas soluciones. Todas estas opciones tienen

---

**Algoritmo 6** Búsqueda en Entornos Variables Sesgada (SVNS) para problemas de optimización de minimización

---

**Entrada:** Conjunto de los distintos entornos  $N_k$  for  $k = 1, \dots, k_{max}$

$s = s_0$  {Solución Inicial}

**Salida:** Mejor solución encontrada.

```
1: repetir
2:    $k = 1$ 
3:   repetir
4:      $s' = \text{SoluciónAleatoria}(N_k(s))$  {Escoge una solución aleatoria  $s'$  del entorno  $N_k(s)$ }
5:      $s'' = \text{BúsquedaLocal}(N_k(s'))$  {Ejecuta el algoritmo de búsqueda local sobre el entorno  $N_k(s')$  de  $s'$ }
6:     si ( $f(s'') < f(s)$ ) entonces
7:        $s = s''$ 
8:        $k = 1$ 
9:     si no, si ( $s'' - \alpha \times f(s, s'') < s$ ) entonces
10:       $s = s''$ 
11:       $k = 1$ 
12:    si no
13:       $k = k + 1$ 
14:    fin si
15:  hasta que  $k = k_{max}$ 
16: hasta que Criterio de parada
```

---

ventajas e inconvenientes, por lo que debe examinarse individualmente en función del problema a resolver.

## 5.4. Búsqueda Tabú

La *búsqueda Tabú* (*Tabu Search*, TS) se engloba dentro de las metaheurísticas de búsqueda con memoria, y es usada para evitar el anclaje en un óptimo local. La metaheurística es publicada por primera vez en [42], por Glover. Del mismo autor podemos encontrar otras publicaciones sobre la búsqueda tabú como [43, 44, 46], en las que se muestran algunas variantes de la metaheurística y aplicaciones de la misma.

La TS utiliza heurísticas para la búsqueda dentro del entorno, permitiendo movimientos que no mejoren la solución actual, y evita que se repitan movimientos realizados. Para esto, los últimos movimientos son almacenados en una lista llamada *lista tabú*. La lista tabú provee de un carácter exploratorio al método y a su vez evita que se produzcan ciclos innecesarios en la búsqueda de soluciones.

## 5.4. BÚSQUEDA TABÚ

---

Otro elemento relevante es el *criterio de aspiración*, por el cual si un movimiento genera una solución mejor que cualquier otra conocida, se permite la aplicación del movimiento pese a encontrarse en la lista tabú. Esto nos permite intensificar cuando es necesario.

En consecuencia, la lista tabú nos restringe ciertos movimientos mientras que el criterio de aspiración permite realizar movimientos pertenecientes a la lista tabú cuando se den ciertas condiciones.

En el Algoritmo 7 encontramos la versión básica de la TS. Sin embargo, existen numerosas variantes acerca de cómo tratar esta lista tabú y los movimientos permitidos en cada iteración. Por ejemplo, podemos encontrar TS con memoria de corto plazo o de largo plazo.

---

### Algoritmo 7 Búsqueda Tabú (TS)

---

**Entrada:**  $s = s_i, i = 0, s_i \in S$  {Solución Inicial} siendo  $i$  la iteración actual y  $S$  el espacio de soluciones

**Salida:** Mejor solución encontrada.

- 1:  $N(s)$  {Determinar el entorno de  $s$ }
  - $LT(s, i)$  {Determinar la lista tabú}
  - $CA(s)$  {Definir Criterio de Aspiración}
  - 2: **repetir**
  - 3:    $s' = \text{MejorSolución}(N(s))$  {Escoge la mejor solución  $s'$  del entorno  $N(s)$ , teniendo en cuenta  $LT$  y  $CA$ .}
  - 4:   **si** ( $f(s') < f^*$ ) {Siendo  $f()$  la función objetivo (< o > dependerá, si minimizamos o maximizamos)} **entonces**
  - 5:      $s = s'$
  - $f^* = f(s')$
  - 6:   **fin si**
  - 7:   Actualizar( $LT$ ) {Actualizar la lista tabú ( $LT$ )}
  - $i = i + 1$
  - 8: **hasta que** Criterio de parada
- 

El objetivo de la memoria a corto plazo es penalizar los movimientos que generan soluciones visitadas recientemente. De esta forma, se almacenan estos movimientos en la lista tabú durante un número determinado de iteraciones. Una vez que pasa este número de iteraciones (el cual no debe ser excesivamente elevado) el movimiento queda liberado y se permite de nuevo su uso.

La memoria a largo plazo no se basa en las soluciones más recientes sino que se basa en las soluciones más frecuentes. Este tipo de memoria es ligeramente más compleja y completa, ya que almacena un número mayor de movimientos o soluciones, y puede

utilizarse de dos formas distintas. La primera es con fines de exploración, mediante la cual se penalizan las soluciones más visitadas para explorar espacios de búsqueda menos conocidos. La segunda es el uso con fines de intensificación, que consiste en buscar la explotación de espacios de búsqueda ya conocidos en los que se han encontrado las mejores soluciones.

Además de las variantes anteriormente nombradas, podemos encontrar multitud de investigaciones [13, 56, 23] sobre búsqueda tabú y sus distintas aplicaciones a problemas de asignación y *timetabling* cada vez más complejos.

## 5.5. Expresiones regulares

Las expresiones regulares o Regex [40] son cadenas de texto especiales utilizadas para describir patrones de búsqueda. Regex está compuesto por meta-caracteres tales como (), [] and {}; \A (inicio de una cadena), \s (espacio en blanco) o \d (dígitos), y cuantificadores \* (0 o más), + (1 o más) y ? (0 o 1).

La principal aplicación en lo que a programación se refiere, es la automatización de procesos de búsquedas y análisis empleados en múltiples ocasiones. Existen numerosas herramientas que utilizan Regex para búsqueda de patrones: Java (RegEx, ver Package java.util.regex); JavaScript desde la versión 1.2; Perl; C y C ++ (la librería PCRE); PHP; Python y lenguajes .Net.

Los beneficios del uso de expresiones regulares son la alta velocidad de comprobación y su modularidad debido a su simplicidad y claridad. Gracias a esto, se permite una implementación de un modelo claro y eficaz.

Las expresiones regulares tienen un gran número de aplicaciones, incluyendo identificación de patrones en secuencias de ADN [72], en aprendizaje automático para clasificación de comentarios en redes sociales [83] o para la extracción de información en texto no estructurado con procesamiento de lenguajes naturales [93].

La *minería de textos* es un área con muchas líneas de investigación abiertas y las expresiones regulares son una herramienta muy potente en este ámbito. Existen numerosos estudios sobre minería de textos y expresiones regulares, en ámbitos como las redes sociales [35], filtrado de spam [92, 84] y comprobación de parámetros, entre otros.

## 5.5. EXPRESIONES REGULARES

---

Un ejemplo de expresiones regulares es el siguiente patrón

`coche(s)?`

que sirve para reconocer las palabras *coche* y *coches*. Otro ejemplo de patrón utilizado para reconocer direcciones de correo en un texto sería el siguiente:

`(\w+@[a-zA-Z]+\.[a-zA-Z]{2,6}) .`

Este patrón permite reconocer distintos correos, la expresión comienza con \w lo que permite reconocer caracteres de *a – z, A – Z, 0 – 9* incluyendo el símbolo *\_ underscore*. La expresión continua con @, reconociendo el mismo símbolo, y los caracteres [a – z, A – Z] +, lo que permite el reconocimiento de cualquier conjunto de caracteres alfabéticos. A esto le sucede un punto (.) y la expresión [a – z, A – Z]{2,6} que permite reconocer un conjunto de 2 a 6 caracteres alfabéticos. Por lo que este patrón podría reconocer correos como: *Faustino\_Tello\_01@email.es, F@e.es, o F01Tello\_01@MailMail.online.*

En nuestro caso, para el problema de creación de horarios de trabajo para controladores aéreos, Regex es usado para comprobar las condiciones laborales de los controladores que forman parte de las restricciones de los problemas que se van a resolver. A pesar de que no todas las restricciones laborales son comprobadas con Regex, sí lo son la gran mayoría de ellas. Las restricciones no comprobadas con Regex fueron implementadas desde el inicio por su alta complejidad. Todos los patrones utilizados para la comprobación de restricciones se encuentran explicados en el Apéndice A.

---

## CAPÍTULO 5

---

## Capítulo 6

# Metodología de solución propuesta

---

En este capítulo introducimos las distintas metodologías propuestas para la resolución de los tres tipos de problemas *timetabling* de asignación de controladores aéreos a puestos de control considerados en la presente Tesis Doctoral.

En primer lugar, presentamos la modelización de las soluciones y su codificación que es común a los tres problemas de optimización considerados. A continuación, en las siguientes secciones, abordamos cada uno de los tres problemas, recordando brevemente las características de los mismos, descritos en detalle en el Capítulo 3, describiendo en detalle la metodología de solución propuesta en cada caso, e ilustrándola mediante la resolución de distintas instancias de los problemas.

### 6.1. Representación de la solución

Los problemas de optimización *timetabling* que estamos resolviendo consisten en la asignación de controladores a sectores operativos mientras se respetan ciertas condiciones laborales. Los controladores cubren los sectores gestionando el tráfico aéreo en este área y, ante todo, garantizando la seguridad del tráfico aéreo.

Como paso previo a la explicación de la metodología de resolución utilizada en cada uno de los problemas, procedemos a mostrar una serie de términos y elementos que usaremos en el proceso de resolución del problema para representar las soluciones obtenidas.

La *matriz de turnos* o *matriz de trabajo* será donde se representen los sectores que deben ser cubiertos por los controladores en cada momento. Las filas de la matriz se corresponden con los turnos de trabajo de los controladores, mientras que las columnas

representan los intervalos de tiempo o slots en los que se divide el turno. Adicionalmente, cada una de las filas de la matriz es asignada a un controlador.

Cada celda de la matriz de turnos contiene información sobre el sector que cubre el controlador de dicha fila durante el intervalo de tiempo asociado a su columna y la posición de trabajo en dicho sector (planificador o ejecutivo). También es posible que se encuentre en un período de descanso. Esta información se representa por tres símbolos.

Los descansos de los controladores se representan con la combinación 111, los controladores que se encuentran fuera de turno se representan con la combinación 000 (esta representación únicamente es necesaria en el tercio de los problemas considerados, en fase táctica) y, mientras que los controladores se encuentren trabajando, utilizamos tres letras que identifican el sector en el cual el controlador se encuentra trabajando. Cuando las letras se encuentran en mayúscula, denotan que el controlador se encuentra trabajando en posición de ejecutivo, cuando están en minúscula, denotan que el controlador se encuentra trabajando en posición de planificador.

El tamaño del intervalo de tiempo o slot, se ha establecido en cinco minutos. Esta decisión se ha llevado a cabo debido a que un tamaño más pequeño aumenta considerablemente las dimensiones del problema. Se ha consultado con expertos en la materia pertenecientes a CRIDA, los cuales estiman adecuado un slot de cinco minutos para una correcta resolución del problema, considerando que todas las restricciones del problema se adecúan a esta discretización del tiempo.

Para facilitar la comprensión de la matriz se han utilizado distintos colores para cada uno de los sectores y el color blanco para representar los descansos.

En la Figura 6.1 se muestra un ejemplo de la matriz de turnos. En este ejemplo podemos observar que en la solución hay veintiocho controladores (filas) y que se corresponde con un turno de 7 horas y 20 minutos (88 slots) de duración. El primer controlador, comienza con un período de descanso de 16 slots (1 hora y 20 minutos), representado por 111, seguido por un período de trabajo en el sector *abb* en posición de planificador y posteriormente en el mismo sector, pero en posición de ejecutivo (*ABB*). También podemos observar como todos los períodos de descanso se representan con 111 y se utiliza el color blanco, mientras que los períodos de trabajo, utilizan el código asociado al sector y cada uno tiene un color asignado.

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

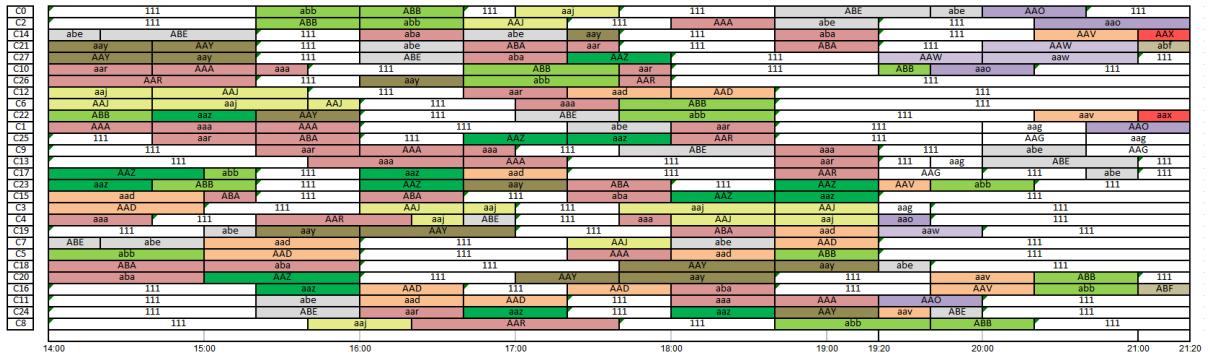


Figura 6.1: Matriz de turnos

## 6.2. Primera aproximación al problema. Fase pre-táctica

El problema al que nos enfrentamos consiste en la creación de horarios de trabajo minimizando el número de controladores utilizado a la vez que respetamos sus condiciones laborales. Para la resolución del problema contamos con un elevado número de controladores, el cual se irá reduciendo en la medida de lo posible debido a que el problema tiene un amplio conjunto de restricciones, expuestas en la Sección 3.1.1, y deben cumplirse para la correcta resolución del problema.

Para la resolución del problema hemos propuesto una metodología dividida en dos fases, una primera fase que utiliza una heurística basada en el uso de plantillas para la construcción de soluciones iniciales factibles, mientras que la segunda fase utiliza un algoritmo multicomienzo basado en el *recocido simulado* para optimizar las soluciones generadas en la primera fase. En esta segunda fase el objetivo es minimizar el número de controladores necesarios (manteniendo la factibilidad de las soluciones visitadas).

### 6.2.1. Fase 1: Creación de soluciones iniciales factibles

Como ya se ha mencionado, la creación de soluciones iniciales para el algoritmo de recocido simulado puede ser un factor decisivo en la resolución satisfactoria del problema.

El algoritmo de inicialización de soluciones se basa en el uso de una plantilla y el ajuste de la misma para cumplir las condiciones del entorno. El objetivo principal de esta fase es encontrar una solución inicial factible. Se ha hecho un estudio de las posibles plantillas que se podrían usar, y se ha optado por la plantilla que explicaremos a continuación, la

cual nombramos como 3x1.

La plantilla 3x1 consta de tres controladores que se utilizan para cubrir un sector en las posiciones de ejecutivo y planificador. A cada controlador se le asigna de forma repetitiva un período de descanso y dos de trabajo (planificador y ejecutivo), tal y como se muestra en la Figura 6.2.

<b>ATC 1</b>	Working	Resting	Working	Working	Resting	Working	Working	...
<b>ATC 2</b>	Resting	Working	Working	Resting	Working	Working	Resting	...
<b>ATC 3</b>	Working	Working	Resting	Working	Working	Resting	Working	...

Figura 6.2: Plantilla 3x1 de referencia para la creación de soluciones iniciales

La longitud del turno depende del problema que vamos a resolver, y la plantilla siempre abarca todo el turno, por lo que su longitud es variable. En la plantilla, la longitud del trabajo siempre será el doble que la del descanso, por lo que esta plantilla tiene un porcentaje mínimo de descanso del 33 %. Esto provoca que, en casos en que se requiera un 50 % del turno de descanso, se tenga que utilizar otra plantilla específica.

Debido a las condiciones del problema y la estructura de la plantilla, el tiempo de descanso siempre es superior o igual al tiempo de descanso mínimo, e inferior o igual a la mitad del tiempo de trabajo máximo, es decir,

$$\text{descanso\_mínimo} \leq \text{descanso} \leq \text{trabajo\_máximo\_consecutivo}/2. \quad (6.1)$$

Las condiciones laborales establecen un descanso mínimo de seis slots (treinta minutos) y de trabajo máximo de veinticuatro slots (dos horas), por lo que el rango de descanso estaría comprendido en el intervalo [6,12] (se debe recordar que el descanso siempre será, a lo sumo, la mitad que el tiempo máximo de trabajo). Por tanto, tenemos siete posibles longitudes distintas de intervalos. En función de la longitud que seleccionemos, la plantilla será distinta y la solución inicial obtenida también. De esta forma, utilizando distintas longitudes disponemos de un algoritmo de multicomienzo.

Supongamos seleccionada una longitud ( $L=6$ ) para los intervalos, dando lugar a un intervalo de descanso de seis slots y un intervalo de trabajo de doce slots. La plantilla correspondiente se muestra en la Figura 6.3.

Una vez descrita la estructura de las plantillas que vamos a utilizar, procedemos a

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

<b>Trabajador 1</b>	T	D	t	T	D	t	T	D	t	T	D	t	T	D	t	T
<b>Trabajador 2</b>	D	t	T	D	t	T	D	t	T	D	t	T	D	t	T	D
<b>Trabajador 3</b>	t	T	D	t	T	D	t	T	D	t	T	D	t	T	D	t
	t	t	t	t	t	t	T	T	T	T	T	T	T	T	T	T
	T	T	T	T	T	T	D	D	D	D	D	D	D	D	D	D

Figura 6.3: Plantilla de referencia para  $L=6$

explicar el funcionamiento de la heurística para la construcción de la solución inicial.

Se ha dividido la heurística en tres fases, durante la primera fase (introducción de plantillas) se construye la matriz de turnos mediante el uso de plantillas  $3 \times 1$ . Se utiliza una plantilla para cada uno de los sectores abiertos y en caso de que el sector se cierre en algún momento, se rellena el resto de la plantilla con descansos. Una vez finalizada esta fase se tiene una solución generalmente infactible, por no cumplir con la restricción de trabajo mínimo consecutivo. En la segunda fase (reparación de soluciones), se utiliza un algoritmo para transformar las soluciones infactibles en soluciones factibles, lo que no siempre es posible. Este algoritmo realiza traspasos de trabajo entre los distintos turnos con el objetivo de cumplir las restricciones que se estuvieran incumpliendo. Por último, la tercera fase (comprobación de restricciones) únicamente comprueba la factibilidad de las soluciones generadas y desecha las soluciones infactibles.

### Introducción de plantillas

En esta fase de la heurística nos aseguramos de que todos los sectores abiertos se encuentran cubiertos. Este proceso, se encuentra descrito en el Algoritmo 8.

En primer lugar, se obtienen todos los sectores que han sido abiertos en algún momento del turno. Los sectores solo aparecerán una vez en la lista, sin que estos se repitan (*Sectors*). También obtenemos la longitud del turno en función del número de slots (*nSlots*).

El Algoritmo 8 comienza iterando la lista de sectores abiertos. Para cada uno de los sectores, se introduce una plantilla vacía en la matriz de turnos con la función *Introducir-PlantillaVacia()*. Una vez se introduce la plantilla, iteramos la lista de todos slots en los que se divide el turno, mientras se comprueba si el sector  $i$  se encuentra abierto en el slot

**Algoritmo 8** Algoritmo de asignación de plantillas

---

```

1: para todo  $i \in Sectors$  hacer
2:    $P[][] = \text{IntroducirPlantillaVacía}()$ 
3:   para todo  $j \in nSlots$  hacer
4:      $o = \text{AbiertoEn}(i,j)$  Indica si el sector  $i$  se encuentra abierto en el slot  $j$ 
5:     si ( $o == \text{true}$ ) entonces
6:        $P = \text{CompletarPlantillaTrabajo}(P, j, i)$  Introduce en la plantilla vacía el nom-
      bre del sector  $i$  en la columna del slot  $j$ .
7:     si no
8:        $P = \text{CompletarPlantillaDescanso}(P, i)$  Introduce en la plantilla vacía el iden-
      tificador de descanso en la columna del slot  $j$ .
9:     fin si
10:    fin para
11:  fin para

```

---

$j$ , esto se realiza utilizando la función  $\text{Abierto}(i,j)$  del algoritmo.

En caso de que el sector esté abierto, se pasa al Paso 6 del algoritmo (función  $\text{CompletarPlantillaTrabajo}(P, j, i)$ ), donde se introduce la información del sector. En el caso contrario (el sector se encuentra cerrado), pasamos al Paso 7 (función  $\text{CompletarPlantillaDescanso}(P, i)$ ), donde se introduce el identificador del descanso. Una vez se termina la iteración de la lista de sectores, el algoritmo de asignación de plantillas finaliza.

En el Paso 6, es decir, en la función  $\text{CompletarPlantillaTrabajo}$  puede suceder lo siguiente: si el sector se encuentra abierto en el slot  $j$  y este se cierra en  $j + 1$ , puede que alguno de los controladores que estaban trabajando en  $j$  dejen de trabajar por el cierre de sectores, y por tanto, no cumplan la restricción de trabajo mínimo consecutivo.

Vamos a mostrar un ejemplo para ilustrar el funcionamiento de esta primera parte del algoritmo. La sectorización usada se muestra en la Figura 6.4.

2:00	2:30	3:00	3:30	4:00	4:30	5:00	5:30	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30
Sector X															
		Sector Y													
		Sector Z						Sector Z							

Figura 6.4: Ejemplo de sectorización

Es una sectorización con tres sectores abiertos, uno se mantiene abierto todo el turno (ocho horas), otro solo se abre durante 3:30 horas en medio del turno y el último se abre y cierra dos veces. En este ejemplo cada celda representará un total de seis slots, al igual

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

que se explica en la Figura 6.3.

1. Una vez tenemos la lista de sectores abiertos, en este caso tres, comenzamos con el primer sector, el Sector X. Lo primero será introducir una plantilla vacía como dice en el punto dos del algoritmo.
2. Continuamos en el punto tres, por lo que comenzamos la iteración por slots. Para el primer sector  $i$  y el primer slot  $j$  se comprueba si el sector está abierto (punto cuatro). Dado que, efectivamente sí lo está, pasamos al punto seis y se introduce en el slot correspondiente de la plantilla modificando la  $T$  general de trabajo por el identificador del sector  $i$  (Figura 6.5).

	Sector X											
	Sector Y						Sector Z					
	Sector Z						Sector Z					
C1	X											
C2	1											
C3	x											

Figura 6.5: Explicación del algoritmo de inicialización 1/4

3. Se sigue introduciendo la plantilla mientras que el sector permanezca abierto.
4. Una vez se ha recorrido todo el array de slots pasamos al siguiente sector. Y por tanto se introduce una nueva plantilla vacía como se indica en el punto dos.
5. En este sector repetimos el proceso. Al estar cerrado durante las primeras horas, lo cual se comprueba por cada slot en el punto cinco, se introducen descansos en el turno hasta que este sector sea abierto (Figura 6.6).

	Sector X											
	Sector Y						Sector Z					
	Sector Z						Sector Z					
C1	X	1	x	X	X	1	x	X	1	x	X	1
C2	1	x	X	1	x	X	1	x	X	1	x	X
C3	x	X	1	x	X	1	x	X	1	x	X	1
C4	1	1	1	1								
C5	1	1	1	1								
C6	1	1	1	1								

Figura 6.6: Explicación del algoritmo de inicialización 2/4

6. Cuando el sector Y se abre, se asigna el slot perteneciente a la plantilla en la posición en la que nos encontramos, y así se continúa iterando sobre los slots del mismo modo. Cabe destacar, que el hecho de que el sector estuviera cerrado con anterioridad, no implica que se comience con la plantilla de cero, si no que se introducirán los valores correspondientes al slot en el que nos encontramos.
7. Si el sector se cierra, se continúa el proceso, dejando de introducir los datos que corresponden a la plantilla para introducir únicamente descansos hasta que este se vuelva a abrir o se termine el turno (Figura 6.7).

	Sector Y															
	Sector Z								Sector Z							
	X	1	x	X	1	x	X	1	x	X	1	x	X	1	x	X
C1	X	1	x	X	1	x	X	1	x	X	1	x	X	1	x	X
C2	1	x	X	1	x	X	1	x	X	1	x	X	1	x	X	1
C3	x	X	1	x	X	1	x	X	1	x	X	1	x	X	1	x
C4	1	1	1	1	1	y	Y	1	y	Y	1	1	1	1	1	1
C5	1	1	1	1	y	Y	1	y	Y	1	y	1	1	1	1	1
C6	1	1	1	1	Y	1	y	Y	1	y	Y	1	1	1	1	1

Figura 6.7: Explicación del algoritmo de inicialización 3/4

8. Una vez terminado el recorrido completo de todo el conjunto de sectores que forman parte de la sectorización, el algoritmo ha finalizado dando como resultado la matriz de trabajo, la cual se muestra en la Figura 6.8.

	Sector X															
	Sector Y								Sector Z							
	X	1	x	X	1	x	X	1	x	X	1	x	X	1	x	X
C1	X	1	x	X	1	x	X	1	x	X	1	x	X	1	x	X
C2	1	x	X	1	x	X	1	x	X	1	x	X	1	x	X	1
C3	x	X	1	x	X	1	x	X	1	x	X	1	x	X	1	x
C4	1	1	1	1	1	y	Y	1	y	Y	1	1	1	1	1	1
C5	1	1	1	1	y	Y	1	y	Y	1	y	1	1	1	1	1
C6	1	1	1	1	Y	1	y	Y	1	y	Y	1	1	1	1	1
C7	1	1	1	1	1	z	Z	1	1	1	1	z	Z	1	1	1
C8	1	1	1	1	z	Z	1	z	Z	1	1	z	Z	1	1	1
C9	1	1	1	1	Z	1	z	Z	1	1	Z	1	z	1	1	1

Figura 6.8: Explicación del algoritmo de inicialización 4/4

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

### Reparación de soluciones

Retomando lo mencionado anteriormente acerca de que no todas las matrices de turnos generadas son factibles, una vez que son generadas, se aplica un algoritmo que elimina algunas de las infactibilidades que puedan existir. Tenemos el pseudocódigo asociado a esta tarea en el Algoritmo 9. Este se centra en eliminar las infactibilidades producidas por no cumplir los trabajos mínimos, ya que son las más usuales cuando generamos las matrices durante la introducción de las plantillas.

El Algoritmo 9 comienza recorriendo la matriz de turnos, la cual recorremos por filas ( $Ctrls$ ), estando cada fila asignada a un controlador. Lo primero que se realiza es comprobar si la fila  $i$  cumple la condición de trabajo mínimo en todo momento, lo cual se realiza con la función *ComprobarTrabajoMinimo()*. En el caso que esta no se cumpla se recorre la fila identificando las infracciones y procediendo a su resolución.

El primer paso que se realiza es comprobar si la carga de trabajo que genera la infactibilidad se puede traspasar, esto se realiza en la función *ComprobarTraspasoTrabajo()*. Para esto, comprobamos que exista un controlador trabajando en ese mismo sector justo en el slot anterior o posterior al que nuestro controlador se encuentra trabajando en el sector. Además, debemos comprobar que se cumplan las siguientes restricciones (las restricciones se encuentran descritas en la Sección 3.1.1) en ambos controladores para evitar generar nuevas infactibilidades:

1. Comprobar que se cumpla la condición de trabajo máximo (restricción 4).
2. Comprobar que se cumpla la condición de descanso mínimo (restricción 5).
3. Comprobar que se cumpla la condición de trabajo mínimo (restricción 6).

En el caso de que se pueda realizar el traspaso de trabajo, se realiza mediante la función *RealizarTraspasoTrabajo()*. En caso contrario, se intentará realizar una adquisición para solucionar la infactibilidad. Para comprobar si se puede realizar la adquisición se utiliza la función *ComprobarAdquisicionTrabajo()*, en esta función debemos realizar las mismas comprobaciones que al comprobar el traspaso. La diferencia reside en que en el caso del traspaso, el controlador que incumple la restricción traspasa el trabajo y en el caso de la

adquisición, es el controlador que no incumple la restricción el que traspasa la carga de trabajo.

---

**Algoritmo 9** Algoritmo de reparación de soluciones

---

```

1: para todo  $i \in CtrlS$  hacer
2:    $o = \text{ComprobarTrabajoMinimo}(i)$  Comprueba si la fila  $i$  cumple la restricción de
   trabajo mínimo
3:   si ( $o! = \text{true}$ ) entonces
4:     para todo  $j \in CtrlS$  hacer
5:       si ( $j! = i$ ) entonces
6:          $m = \text{ComprobarTraspasoTrabajo}(j, i)$  Comprueba si es posible traspasar el
         conjunto de slots que causan la infactibilidad
7:         si ( $m == \text{true}$ ) entonces
8:            $\text{RealizarTraspasoTrabajo}(j, i)$  Realiza el traspaso de los slots indicados
9:         si no
10:           $n = \text{ComprobarAdquisicionTrabajo}(j, i)$  Comprueba si es posible adquirir
            un conjunto de slots y unirlos a los que causan la infactibilidad (aumen-
            tando el trabajo para cumplir la restricción)
11:          si ( $n == \text{true}$ ) entonces
12:             $\text{RealizarAdquisicionTrabajo}(j, i)$  Realiza la adquisición de los slots indi-
              cados
13:          fin si
14:        fin si
15:      fin si
16:    fin para
17:  fin si
18: fin para

```

---

En las Figuras 6.9 y 6.10 mostramos un ejemplo de cómo el método *RealizarTraspasoTrabajo* puede realizar un traspaso de la carga de trabajo entre dos controladores, y en las Figuras 6.11 y 6.12 mostramos un ejemplo de cómo funciona el método *RealizarAdquisicionTrabajo*.

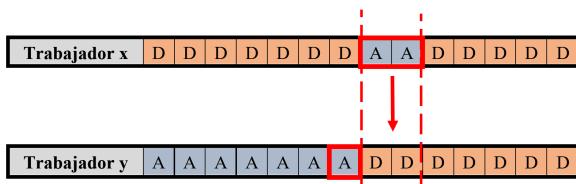


Figura 6.9: Traspaso de fragmento de trabajo con unión anterior

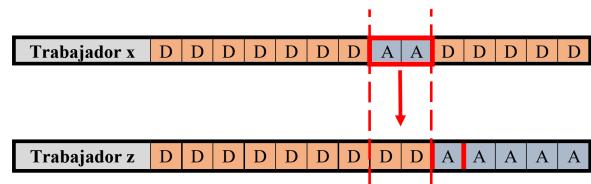


Figura 6.10: Traspaso de trabajo con unión posterior

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

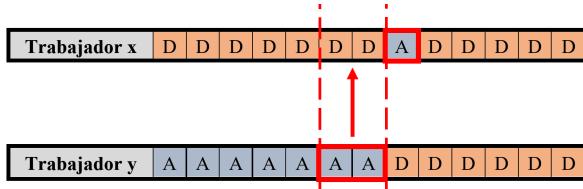


Figura 6.11: Adquisición de fragmento de trabajo con unión anterior

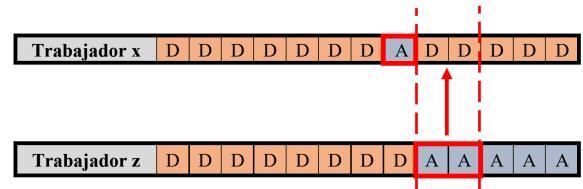


Figura 6.12: Adquisición de fragmento de trabajo con unión posterior

### Comprobación de la factibilidad

En esta fase de creación de soluciones, únicamente nos encargamos de comprobar que las soluciones generadas cumplan con todas las restricciones. En caso contrario, las soluciones infactibles serían desechadas.

Es posible, aunque poco probable, el no encontrar soluciones factibles, además de ser posible que no exista ninguna solución factible para el problema con una determinada sectorización.

Para comprobar la robustez y el correcto funcionamiento de este algoritmo, se ha realizado un estudio por el cual se han generado un conjunto de problemas pseudo-aleatorios, y se ha aplicado el algoritmo de creación de soluciones.

El objetivo de este experimento es medir el porcentaje de soluciones factibles generadas, las restricciones que se incumplen y el tiempo de ejecución empleado en cada una de las fases del algoritmo. Los datos de las pruebas realizadas se muestran en la Tabla 6.1. Se han utilizado tres escenarios, que varían en el número máximo de sectores abiertos simultáneamente, que son 3, 5 y 7, respectivamente.

Todos los escenarios comparten las siguientes características: el número máximo de sectores que se pueden abrir o cerrar simultáneamente es de tres y ningún sector puede permanecer abierto menos de veinte minutos. Para cada uno de los escenarios, se han creado cien problemas pseudo-aleatorios y los resultados mostrados son la media de estos para cada una de las soluciones generadas. Recordemos que se generan siete soluciones con distintas longitudes de intervalos, pero la diferencia de tiempo entre estos no es significativa por lo que mostramos únicamente la media.

En la primera columna de la Tabla 6.1 se muestra el porcentaje de soluciones factibles generadas en función de las distintas plantillas utilizadas (longitud de los intervalos de

Tabla 6.1: Resultados del algoritmo de creación de soluciones

	Porcentaje de soluciones factibles	Media de restricciones incumplidas	Tiempo de cómputo (s)		
			Sec. 1	Sec. 2	Sec. 3
Escenario 1	[48, 64, 55, 55, 57, 0, 63] %	2.14	0.0898	0.00086	0.00214
Escenario 2	[41, 55, 48, 49, 48, 0, 56] %	2.42	1.277	0.001	0.005
Escenario 3	[22, 54, 30, 34, 34, 0, 58] %	2.78	9.185	0.001	0.0091

descanso asociados a cada plantilla [6,7,8,9,10,11,12]). La segunda columna muestra el número medio de restricciones incumplidas por solución. Por último, se muestran los tiempo de ejecución de las distintas partes del algoritmo de creación de soluciones iniciales factibles.

Una vez analizamos los resultados, vimos que algunos intervalos tienen mayor probabilidad que otros para obtener soluciones factibles. En el caso del intervalo con una longitud de once slots no obtiene soluciones factibles, por lo que decidimos desecharlo. También observamos que el algoritmo funciona correctamente independientemente del tamaño del problema. No obstante, se aprecia un ligero descenso de la factibilidad a medida que el tamaño aumenta.

Por último, el aumento del tiempo de cómputo asociado al aumento del tamaño del problema no supone ningún inconveniente ya que el aumento no es relevante respecto al resto de tiempos.

### 6.2.2. Fase 2: SA multicomienzo para obtención de soluciones óptimas

Para la ejecución de esta segunda fase, se utiliza un algoritmo basado en el *recocido simulado*, del cual encontramos su pseudocódigo en el Algoritmo 1 (Sección 5.2.1). Se utiliza un algoritmo multicomienzo ya que nos permite una fácil paralelización y contamos con más de una solución inicial. Aunque el algoritmo esté basado en el SA básico, se deben tomar decisiones de diseño para ajustarnos al problema a resolver.

### Función objetivo

El objetivo es minimizar el número de controladores necesarios para cubrir la sectorización de un turno determinado. La función objetivo utilizada es

$$p = \sum_{k=0}^{nATC-1} \left( \frac{(k+1) \times h_k}{nATC^2} \right), \quad (6.2)$$

donde,  $nATC$  es el número de controladores y  $h_k$  es el número de slots de trabajo del controlador  $k$  (en la fila  $k$ -ésima de la codificación de la solución).

El motivo de utilizar esta función y no utilizar el número de controladores como función es muy sencillo, la gran mayoría de las soluciones dentro del entorno tendrían el mismo valor objetivo, mientras que de esta manera, siempre existe una diferencia entre el valor objetivo de dos soluciones.

Es importante tener en consideración que en las soluciones que vamos analizando, los controlores están ordenados en la codificación de menor o mayor carga de trabajo (en las primeras filas los de menor carga y en las últimas los de mayor carga). El motivo de esta ordenación es reducir progresivamente la carga de trabajo de las filas superiores con menor carga, traspasando esta a las filas inferiores, hasta que se elimine completamente la carga de trabajo de estas filas, y así poder eliminarlas de la matriz. En la Figura 6.13 encontramos una solución obtenida utilizando esta función objetivo, en la cual se aprecia la ordenación de las filas en función de la carga de trabajo.

### Temperatura inicial, función de enfriamiento y número de iteraciones hasta el descenso de la temperatura

Se realizó un conjunto de pruebas básico para ajustar el valor de los parámetros del algoritmo. Para calcular la temperatura inicial, buscamos un valor suficientemente elevado para que el porcentaje de aceptación de las soluciones peores que la actual siempre sea mayor al 90 %.

Para el ajuste de la temperatura inicial se realiza una ejecución previa donde se genera un conjunto de soluciones dentro del entorno de la solución inicial, a partir de estas soluciones se modifica progresivamente la temperatura inicial hasta obtener un porcentaje de aceptación medio cercano al 90 %.

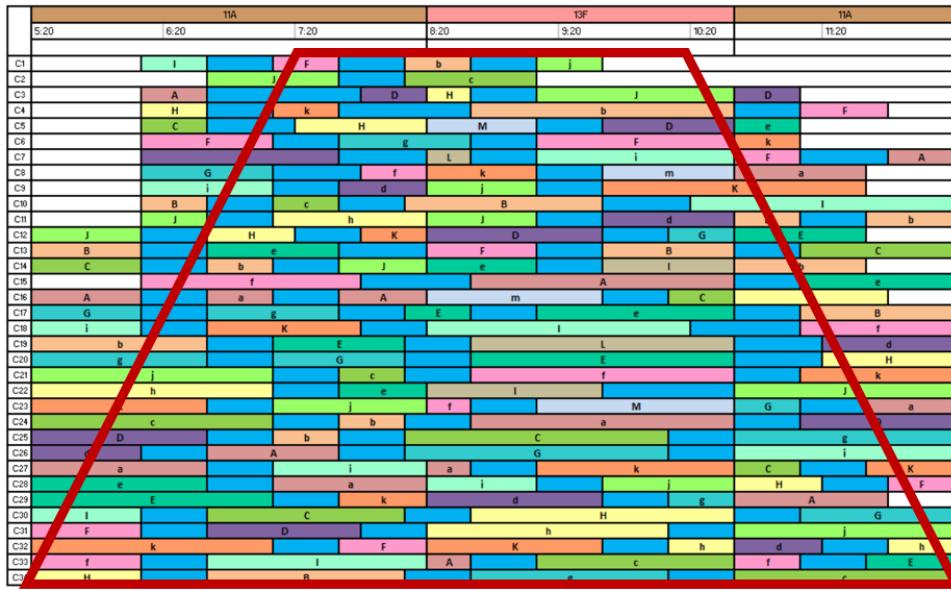


Figura 6.13: Solución resultante de la aplicación de la segunda fase

La función de enfriamiento utilizada es la función de enfriamiento geométrica, de la cual se debe ajustar el parámetro  $\alpha$ . Otro parámetro que debe ajustarse para el correcto funcionamiento del algoritmo es el número de iteraciones hasta el descenso de la temperatura ( $L$ ).

Estos parámetros deben ajustarse para el óptimo funcionamiento del algoritmo, son parámetros que requieren cierta experimentación y un meticuloso proceso de ajuste, en el que deben obtener y analizar datos del progreso y comportamiento del algoritmo. Todo esto se realiza buscando un equilibrio entre la exploración y la explotación en la ejecución, además de la robustez del algoritmo al aplicarlo en distintas instancias del problema.

En la Figura 6.14 se muestra un ejemplo de una instancia concreta de la evolución del porcentaje de aceptación para obtener un correcto equilibrio entre exploración y explotación del algoritmo.

### Definición del entorno de una solución

Otro elemento de gran importancia en el funcionamiento de la metaheurística es la elección de un entorno de soluciones adecuado.

Una de las primeras acciones que se realizaron fue llevar a cabo un estudio para tomar una de las decisiones de diseño más importantes: permitir al algoritmo generar soluciones

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

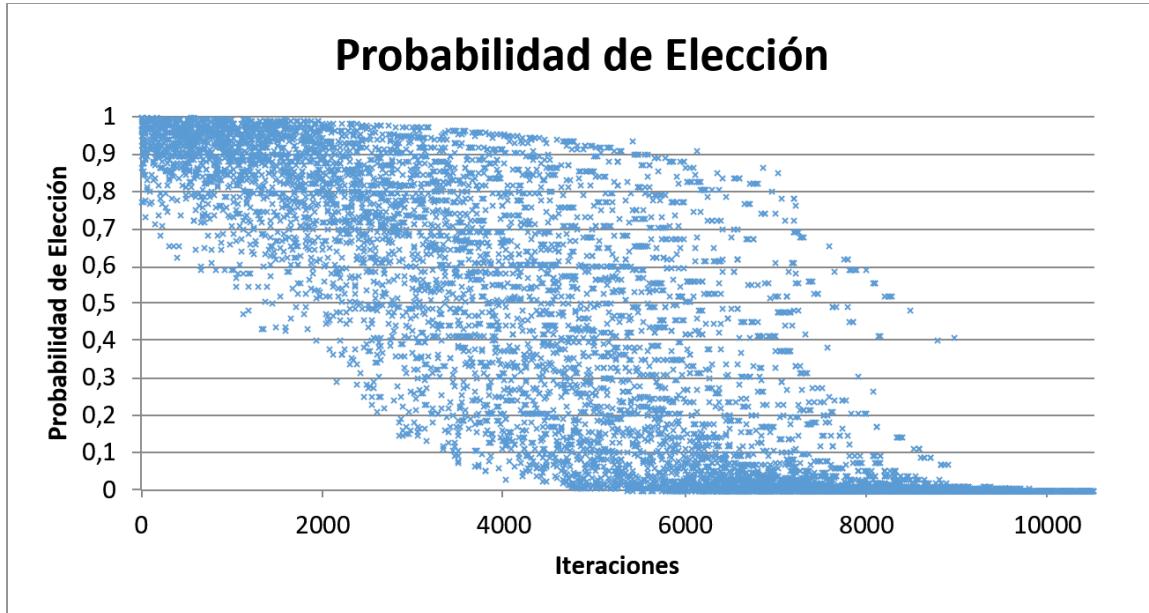


Figura 6.14: Ejemplo de evolución del porcentaje de aceptación

infactibles o solo utilizar soluciones factibles.

Para comprobar si la generación de soluciones infactibles aportaba alguna mejora en los resultados del algoritmo se modificó la función objetivo introduciendo una penalización por las restricciones incumplidas. Una vez que analizamos las trazas generadas por el algoritmo en múltiples ejecuciones y distintos problemas observamos lo siguiente: se incrementa enormemente el espacio de búsqueda, los tiempos de ejecución y provoca peores resultados en las soluciones del algoritmo.

Por estos motivos se escogió realizar el proceso de búsqueda dentro de la región o espacio factible durante toda la ejecución del algoritmo y en el caso que se genere alguna solución infactible esta queda descartada. Se han probado los siguientes entornos:

1. *Entorno 1*: Para la generación de soluciones dentro de este entorno se utiliza el movimiento que se explica a continuación:

- a) Se escoge un primer controlador aleatoriamente, el cual cederá un intervalo de trabajo a un segundo.
- b) Se obtienen sus períodos de trabajo continuados y se escoge uno aleatoriamente.

Si la longitud de este es mayor a la granularidad máxima (parámetro establecido en doce slots), únicamente se escogen los doce primeros, en otro caso se escoge

el intervalo completo.

- c) En el siguiente paso, se obtiene un segundo controlador de forma aleatoria. Al generar la solución se comprueba que los dos turnos que han sufrido modificaciones no incumplan ninguna de las condiciones laborales expuestas en la Sección 3.1.1. En caso de que esta incumplan alguna de estas restricciones, se está generando un individuo no válido, y repite el proceso seleccionando de nuevo el segundo controlador hasta obtener una solución que la cumpla o haber probado con todos los controladores.
- d) Si no se obtiene ningún individuo válido, es decir, que cumpla la condición anterior, se reduce en uno el número de slots del intervalo y se repite el proceso. Esto se realiza sucesivamente hasta una granularidad mínima (parámetro establecido en dos slots).
- e) En el improbable caso de que ninguna de las soluciones generadas sea válida, se selecciona de nuevo aleatoriamente el primer controlador y se repite nuevamente desde el paso b).

2. *Entorno 2*: Este entorno es muy similar al *Entorno 1*, en este caso, el movimiento que se realiza consta de los siguientes pasos:

- a) Este paso es idéntico al paso a) del *Entorno 1*.
- b) Se obtienen sus períodos de trabajo continuados, y se escoge uno aleatoriamente. Posteriormente, se modifica aleatoriamente la longitud del intervalo entre una granularidad mínima y máxima (parámetros establecidos en dos y doce slots, respectivamente). En el caso de que la longitud del período sea menor que la escogida, esta no varía. En caso contrario, se obtendrán únicamente los primeros slots del período de trabajo.
- c) Este paso es idéntico al paso c) del *Entorno 1*.
- d) Si no se obtiene ningún individuo válido, es decir, que cumpla la condición anterior, se modifica el tamaño del intervalo escogiendo nuevamente una longitud aleatoria entre la granularidad mínima y máxima (eliminado el número escogido anteriormente se evita repetir el mismo proceso), y volvemos al paso

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

c). En el improbable caso de que se hayan probado todos los posibles tamaños del intervalo y no se haya obtenido una solución válida, se reinicia el proceso repitiéndolo desde el paso a) (eliminando los controladores ya escogidos).

### Condición de parada

Distintas condiciones de parada han sido probadas. En este caso, hemos preferido utilizar una de las condiciones de parada más comúnmente usadas en la literatura, que consiste en finalizar la ejecución cuando la mejor solución encontrada no se mejora en un porcentaje determinado durante un número de iteraciones.

### Extensión de la Fase 2: Obtención de soluciones balanceadas

Un factor importante a incorporar en el problema identificado por los expertos de CRI-DA fue el equilibrio en la carga de trabajo de los controladores aéreos, con el objetivo de que fuese lo más equilibrada posible, evitando soluciones en las que algunos controladores tuviesen una carga de trabajo muy alta y otros muy baja.

Teniendo esto en cuenta se decidió incorporar en la función objetivo la desviación típica de la carga de trabajo de los controladores en las soluciones analizadas. De esta forma, una vez identificado el número mínimo de controladores necesarios para cierta sectorización, en una fase posterior, se sigue el proceso de búsqueda en el espacio factible, pero buscando soluciones lo más equilibradas posible (para ese número mínimo de controladores identificados).

### 6.2.3. Ejemplo de aplicación

Una vez descrita la metodología de resolución para la primera aproximación al problema, procedemos a exponer un ejemplo de la aplicación de la misma para una instancia del problema.

La instancia escogida pertenece a un turno de mañana que abarca de 5:20 a 12:20, y tiene la sectorización del turno representada en la Figura 6.15. La sectorización está formada por la configuración 11A de 5:20 a 8:20, después se modifica a una configuración 13A de 8:20 a 10:40, y por último, regresa a la configuración 11A de 10:40 hasta el final

del turno (12:20).

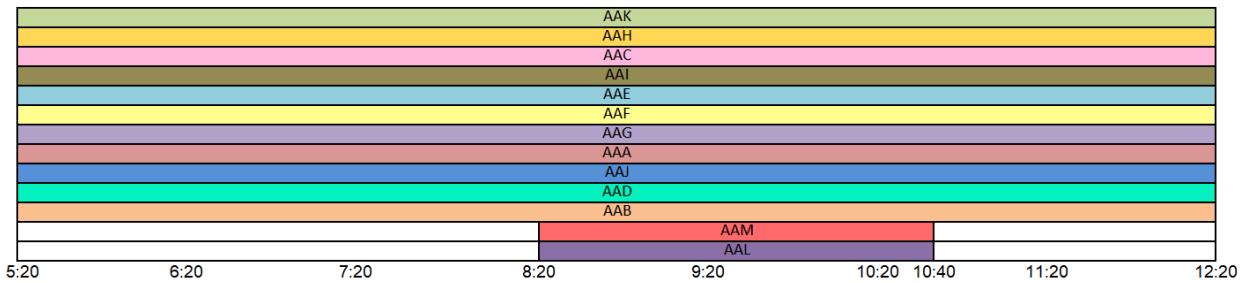


Figura 6.15: Sectorización del turno

Primero se ejecuta la primera fase de la metodología, en la que se construye un conjunto de siete soluciones iniciales factibles utilizando las distintas plantillas 3x1 con distintas longitudes de intervalos de descansos (6,12). Al utilizar plantillas 3x1 y contar con 13 sectores, todas las soluciones iniciales utilizan un total de 39 controladores. La Figura 6.16 muestra una de las soluciones iniciales generadas en esta primera fase.

Una vez finaliza la primera fase se ejecuta la segunda fase para la optimización de las soluciones iniciales. En esta fase se reduce el número de controladores utilizado en las soluciones. Se lleva a cabo la ejecución del algoritmo multicomienzo basado en el *recocido simulado* con los siguientes parámetros (ajustados mediante un proceso de experimentación y análisis del comportamiento de la metaheurística en el proceso de búsqueda):

1. La temperatura inicial se ajusta para que el porcentaje de soluciones peores a la solución actual aceptado sea superior al 90 %. En este caso, el valor obtenido es  $T_0 = 0.75$ .
2. La función de enfriamiento utilizada es la función de enfriamiento geométrica, con un parámetro  $\alpha$  igual a 0.9.
3. El número de iteraciones hasta el descenso de la temperatura ha sido establecido en  $L = 500$ .
4. La condición de parada utilizada finaliza la ejecución en el caso que la mejor solución encontrada no mejore en al menos un 0.05 % durante 1750 iteraciones.
5. El entorno utilizado ha sido el *Entorno 2*.

## 6.2. PRIMERA APROXIMACIÓN AL PROBLEMA. FASE PRE-TÁCTICA

---

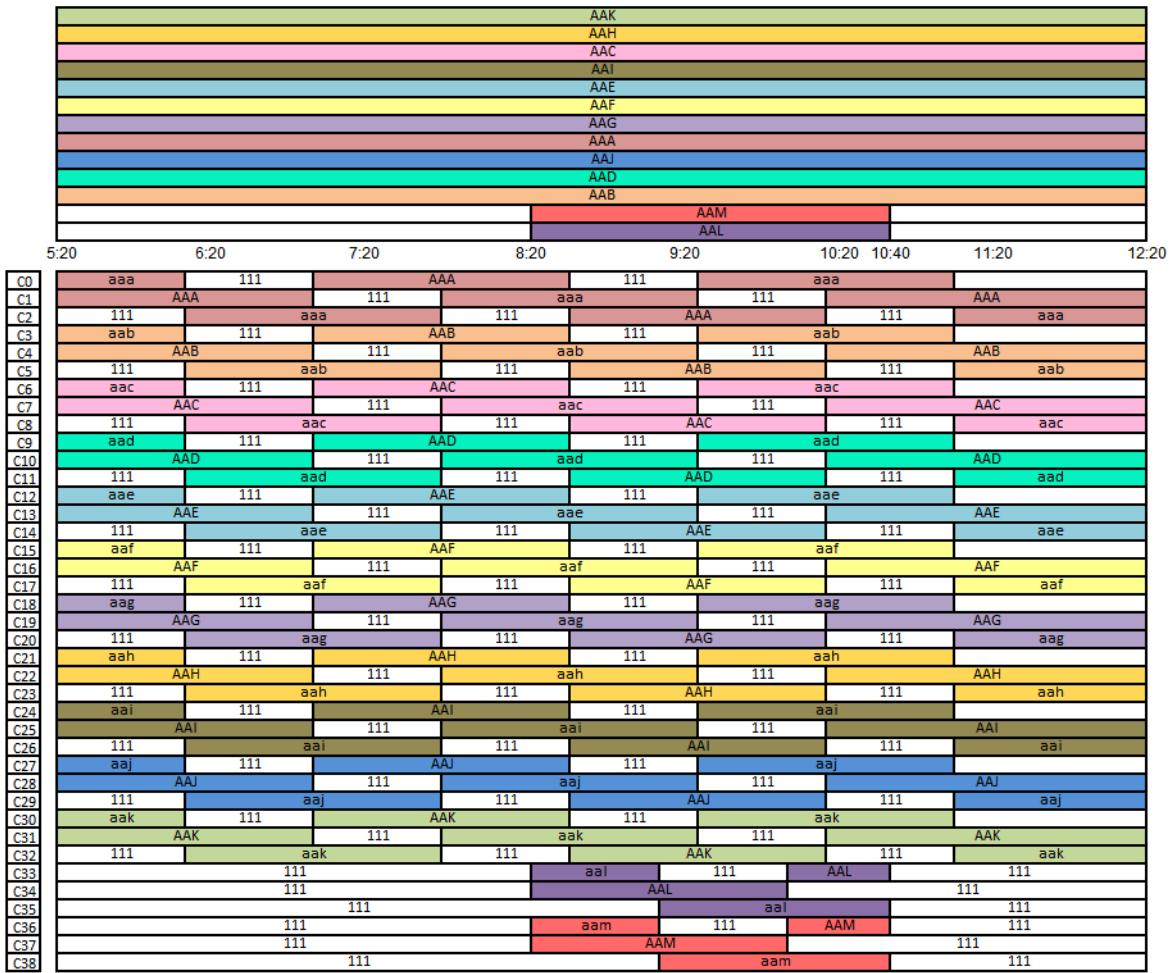


Figura 6.16: Solución inicial

Una vez finaliza la ejecución del algoritmo, obtenemos siete soluciones optimizadas con el siguiente número de controladores respectivamente  $\{35, 34, 34, 35, 34, 34, 34\}$ . Vamos a coger como ejemplo, esta solución se encuentra representada en la Figura 6.17, la que necesita de 34 controladores y tienen una desviación típica de 64.28.

En la Figura 6.17 se puede apreciar que la carga de trabajo no está repartida equitativamente entre todos los controladores. A continuación, se lleva a cabo la extensión de la segunda fase, para el balanceo de la carga de trabajo entre los controladores. Una vez se realiza este proceso, ya se dispone de la solución óptima balanceada, que utiliza 34 controladores y una desviación típica que se ha reducido de 64.28 a 16.17. Esta solución se encuentra representada en la Figura 6.18.

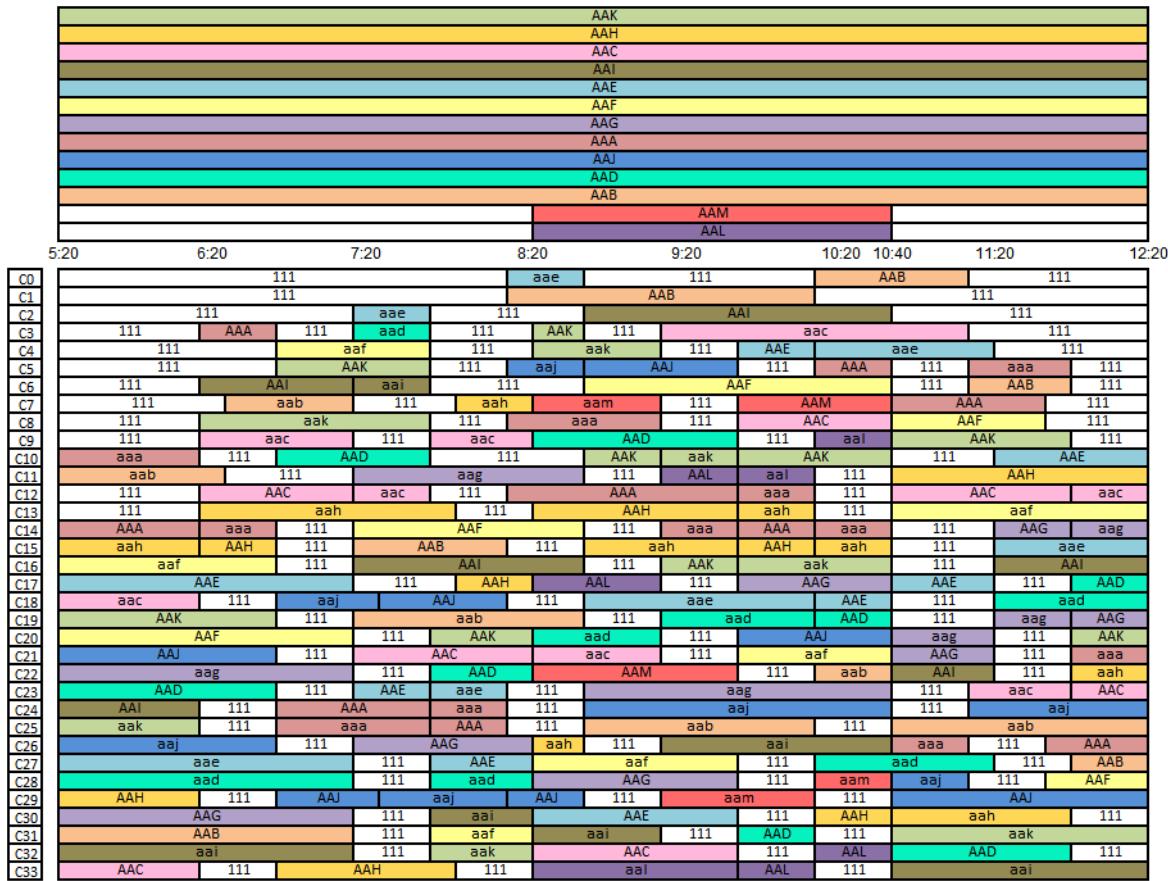


Figura 6.17: Solución óptima

### 6.3. Versión completa del problema

El problema completo consiste en la creación de horarios de trabajo para un número de controladores de tráfico aéreo fijo, a la vez que se cumplen las condiciones laborales de los controladores y optimizamos una serie de objetivos, los cuales describiremos en detalle más adelante.

Si lo comparamos con el problema anterior existen algunas diferencias importantes, por ejemplo, el conjunto de condiciones laborales del que disponemos para la resolución del problema es más restrictivo que en el problema anterior. También debemos destacar que el objetivo del problema anterior es alcanzar una solución factible minimizando el número de controladores utilizados, mientras que en este problema el número de controladores es fijo y la dificultad la encontramos en obtener una solución factible, además de disponer de un amplio conjunto de objetivos que debemos optimizar.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

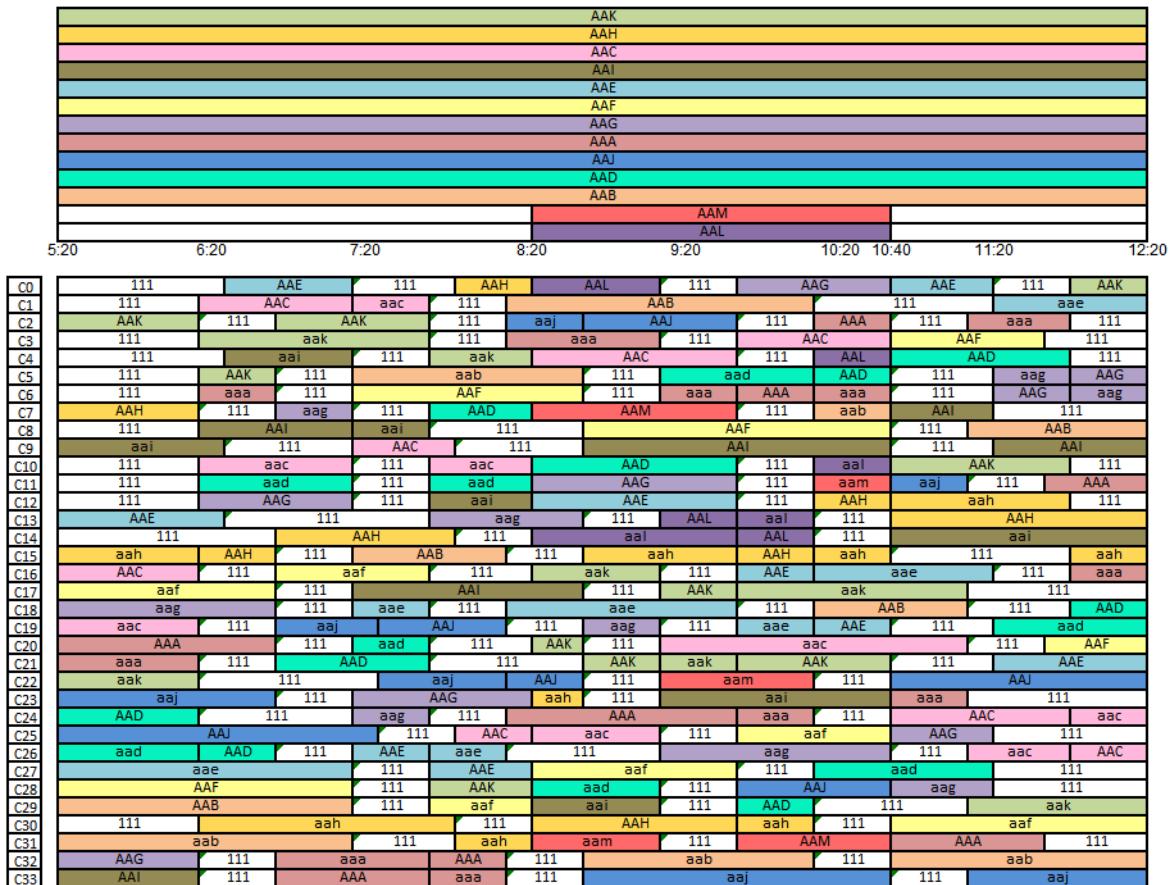


Figura 6.18: Solución óptima balanceada

El método de resolución se compone de tres fases.

En la primera fase, utilizamos una heurística para la construcción de un conjunto de soluciones iniciales. Se utiliza una plantilla optimizada y se modifican las longitudes de los descansos para crear las distintas soluciones. Estas soluciones iniciales pueden incumplir algunas de las condiciones laborales y utilizar más controladores de los disponibles.

En la segunda fase, utilizamos un algoritmo multicomienzo basado en *recocido simulado* (SA). Como entrada del algoritmo se utiliza el conjunto de soluciones iniciales infactibles que provienen de la primera fase para la generación de soluciones factibles. En esta fase utilizamos una función objetivo que penaliza el incumplimiento de las restricciones y tiene como objetivo alcanzar una solución factible, además también penaliza el uso de más controladores que los realmente disponibles para la resolución del problema.

Por último, en la tercera fase, se aplica de nuevo un algoritmo multicomienzo basado en SA. La entrada de este son las soluciones factibles obtenidas en la segunda fase. El

objetivo es optimizar un conjunto de cuatro objetivos. El problema original es un problema multiobjetivo que es transformado en un problema de optimización uniobjetivo con el uso de pesos proporcionados por los expertos de CRIDA. El primero de los objetivos se basa en optimizar una serie de condiciones deseables como el tiempo óptimo de trabajo y descanso continuado. El siguiente objetivo procura que las soluciones sean compactas y no estén completamente fragmentadas. El tercer objetivo está formado por dos sub-objetivos, uno procura que los controladores mantengan sus acreditaciones de trabajo y el otro reducir el número de cambios en sala. El cuarto y último objetivo intenta equilibrar la carga de trabajo entre todos los controladores del turno.

### 6.3.1. Fase 1: Creación de soluciones iniciales

La principal diferencia entre la primera aproximación al problema en la primera fase es que se permite el uso de soluciones infactibles, ya que una de las condiciones es que las soluciones tengan un número fijo de controladores.

En el resto de aspectos es similar al descrito en la primera fase de la primera aproximación al problema. La mayoría de restricciones del problema son distintas, por lo que también la estructura de las plantillas. Se continúa cumpliendo que el tiempo de descanso siempre es superior o igual al tiempo de descanso mínimo e inferior o igual a la mitad del tiempo de trabajo máximo, representado en la Expresión 6.1.

En este caso, el descanso mínimo es de tres slots (quince minutos) y el trabajo máximo es de veinticuatro slots (dos horas). Por tanto, el rango de descanso para formar las distintas plantillas está comprendido en el intervalo  $[3,12]$ , lo que indica que tenemos diez posibles longitudes distintas de descanso y diez soluciones iniciales.

Se sigue utilizando la misma plantilla que se mostró anteriormente en la Figura 6.2.

#### Introducción de plantillas

Utilizamos la misma plantilla ya que cumple con todas las restricciones y, en consecuencia, el algoritmo de introducción de plantillas es el mismo (Algoritmo 8).

### **6.3. VERSIÓN COMPLETA DEL PROBLEMA**

---

#### **Reparación de soluciones**

El algoritmo de reparación de soluciones es el mismo que el mostrado en el Algoritmo 9. No obstante, aquí encontramos la primera diferencia. Para realizar el traspaso o adquisición de trabajo comprobábamos que se cumplieran una serie de restricciones. El problema completo tiene unas restricciones más estrictas por lo que estas comprobaciones son distintas, y se necesita comprobar que se cumplan las siguientes restricciones:

1. La ventana de trabajo máximo (dos horas de trabajo como máximo y treinta minutos de descanso como mínimo).
2. La condición de trabajo máximo continuo (dos horas).
3. La condición de trabajo mínimo (quince minutos).
4. La condición de descanso mínimo (quince minutos).

Salvando estas salvedades, el resto del algoritmo funciona idénticamente.

#### **Asignación de los recursos disponibles**

En la aproximación al problema, omitimos todas las restricciones que influían sobre los controladores, por lo que se tiene que añadir una última parte a la creación de soluciones iniciales en la cual asignamos los controladores disponibles a los distintos turnos de la matriz de turnos.

Los controladores tienen que cubrir ambas posiciones de los sectores abiertos, pero dependiendo del sector a ocupar, se necesita cumplir una serie de requisitos. Existen varios aspectos a tener en cuenta:

- Acreditación de sectores: Un controlador puede tener acreditación PTD y, en ese caso puede ocupar sectores tanto de tipo Ruta como de aproximación (APP); o acreditación CON y, entonces, solo puede ocupar sectores de tipo Ruta.
- Núcleo de pertenencia: Un controlador puede trabajar únicamente en los sectores que pertenecen a su núcleo.

- Turno de pertenencia: Un controlador puede estar asignado a un turno largo o un turno corto y solo podrá trabajar en el turno que tenga asignado.

A partir de estos datos obtenemos información de los sectores que se encuentran en un turno de la matriz de trabajo y nos indica qué turnos puede cubrir un controlador. Por ejemplo, si en un turno tenemos el *sct1*, este es de tipo aproximación, pertenece al núcleo Este y se encuentra abierto durante el turno largo; solo podrá ser ocupado por un controlador con acreditación PTD, perteneciente al núcleo Este y asignado al turno largo.

El pseudocódigo del algoritmo de asignación se encuentra en el Algoritmo 10. Partimos de la matriz de turnos, la cual recorremos por filas (*Turnos*).

Una vez se tienen todos los controladores disponibles asignados, es posible que se queden turnos sin cubrir por algún controlador porque existan más turnos que controladores disponibles. En ese caso, durante el proceso de optimización pueden eliminarse estos turnos u otros, y el controlador será reasignado a uno de los turnos no cubiertos.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

**Algoritmo 10** Algoritmo de asignación de los recursos disponibles

---

```

1: para todo  $i \in Turnos$  hacer
2:   si ( $TipoSectores(i) == APP \&& TrabajoTurno(i) == Largo$ ) entonces
3:     si ( $ControladorDisponible(PTD, Largo) == true$ ) entonces
4:       ListaControladores = AsignarTurnoAControlador( $i, PTD, Largo$ )
5:     fin si
6:   fin si
7:   si ( $TipoSectores(i) == APP \&& TrabajoTurno(i) == Corto$ ) entonces
8:     si ( $ControladorDisponible(PTD, Corto) == true$ ) entonces
9:       ListaControladores = AsignarTurnoAControlador( $i, PTD, Corto$ )
10:    si no, si ( $ControladorDisponible(PTD, Largo) == true$ ) entonces
11:      ListaControladores = AsignarTurnoAControlador( $i, PTD, Largo$ )
12:    fin si
13:  fin si
14:  si ( $TipoSectores(i) == Ruta \&& TrabajoTurno(i) == Largo$ ) entonces
15:    si ( $ControladorDisponible(CON, Largo) == true$ ) entonces
16:      ListaControladores = AsignarTurnoAControlador( $i, CON, Largo$ )
17:    si no, si ( $ControladorDisponible(PTD, Largo) == true$ ) entonces
18:      ListaControladores = AsignarTurnoAControlador( $i, PTD, Largo$ )
19:    fin si
20:  fin si
21:  si ( $TipoSectores(i) == Ruta \&& TrabajoTurno(i) == Corto$ ) entonces
22:    si ( $ControladorDisponible(CON, Corto) == true$ ) entonces
23:      ListaControladores = AsignarTurnoAControlador( $i, CON, Corto$ )
24:    si no, si ( $ControladorDisponible(PTD, Corto) == true$ ) entonces
25:      ListaControladores = AsignarTurnoAControlador( $i, PTD, Corto$ )
26:    si no, si ( $ControladorDisponible(PTD, Largo) == true$ ) entonces
27:      ListaControladores = AsignarTurnoAControlador( $i, PTD, Largo$ )
28:    fin si
29:  fin si
30: fin para
31: si ( $ControladoresNoAsignados != 0$ ) entonces
32:   para todo  $c \in ControladoresNoAsignados$  hacer
33:     si ( $i = TipoControlador(c) == CON$ ) entonces
34:       si ( $i = SectoresEnTurno(Ruta) != -1$ ) entonces
35:         ListaControladores = AsignarTurnoAControlador( $i, c$ )
36:       si no
37:         ListaControladores = AsignarTurnoAControlador( $Random(TurnoNoAsignado), c$ )
38:     fin si
39:   si no, si ( $i = TipoControlador(c) == PTD$ ) entonces
40:     si ( $i = SectoresEnTurno(APP) != -1$ ) entonces
41:       ListaControladores = AsignarTurnoAControlador( $i, c$ )
42:     si no
43:       ListaControladores = AsignarTurnoAControlador( $Random(TurnoNoAsignado), c$ )
44:     fin si
45:   fin si
46: fin para
47: fin si
48: fin para
49: fin si
```

---

### 6.3.2. Fase 2: SA multicomienzo para obtención de soluciones factibles

El principal objetivo de esta fase es transformar las soluciones infactibles en factibles. Para ello, se reduce el número de controladores usados hasta el número de controladores disponibles y se eliminan progresivamente las restricciones incumplidas. Estos procesos se realizan en paralelo.

Para esta fase se utiliza un algoritmo basado en el *recocido simulado uniobjetivo*, el cual se encuentra explicado en la Sección 5.2.1. A continuación, explicaremos las decisiones de diseño tomadas para el diseño del algoritmo.

#### Algoritmo multicomienzo

Lo primero que se debe destacar es que el algoritmo es multicomienzo o *Multiple Independent Run (MIR)*. Esto quiere decir que se pueden ejecutar varias instancias del problema con distintas soluciones iniciales de forma independiente y paralela, lo cual proporciona una clara ventaja, principalmente por dos motivos. Primero, se puede ofrecer un conjunto de soluciones al usuario para su posterior elección, y segundo, si algunas de las ejecuciones no ofrece ninguna solución factible, no pierde tiempo ya que pueden ser ejecutados en paralelo.

#### Función objetivo

La función objetivo que se usa en esta segunda fase del algoritmo de solución tiene como objetivo conseguir una solución que sea factible con un número de controladores igual al disponible. La función constará de dos términos que serán normalizados para que puedan combinarse en un modelo aditivo ponderado. El primero de los términos intentará reducir el número de controladores en la solución, mientras que el segundo intentará reducir la infactibilidad de la misma.

- **Primer término:** La reducción del número de controladores se consigue maximizando

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

$$p = \sum_{k=0}^{nATC-1} \left( \frac{(k+1) \times h_k}{nATC^2} \right), \quad (6.3)$$

donde  $nATC$  es el número de controladores y  $h_k$  es el número de slots de trabajo del controlador  $k$  (en la fila  $k$ -ésima de la codificación de la solución).

Es importante tener en consideración que en las soluciones que vamos analizando, los controladores están ordenados en la codificación de menor o mayor carga de trabajo (en las primeras filas los de menor carga y en las últimas los de mayor carga).

En la expresión anterior, el producto del numerador hace que el valor de  $p$  sea mayor cuanto mayor carga de trabajo tengan los controladores de las últimas filas (es decir, los que tienen la mayor carga de trabajo). Por esta razón, en el proceso de búsqueda se tenderá a pasar carga de trabajo de los controladores de las primeras filas a los de las últimas, haciendo que los controladores de las primeras filas se queden sin carga y se puedan eliminar.

El denominador de la matriz se encuentra elevado al cuadrado para que siempre aumente el valor de  $p$  al eliminar un controlador. Si no estuviera elevado al cuadrado, podría ocurrir que al eliminar un controlador empeorase el valor objetivo.

A continuación, normalizamos el valor de  $p$  con la siguiente expresión:

$$f_1 = 1 - \frac{p_{max} - p}{p_{max} - p_{min}}, \quad (6.4)$$

siendo  $p_{max}$  y  $p_{min}$  el valor máximo y mínimo de  $p$  con el objetivo de normalizar la función. Para calcular el valor de  $p_{max}$  utilizamos la expresión:

$$p_{max} = \sum_{k=nATC-(nATC_{completos}+1)}^{nATC} \left( \frac{k \times nSlot}{nATC^2} \right), \quad (6.5)$$

donde asignamos la máxima carga de trabajo a los controladores que se encuentran en las últimas filas, mientras que a las primeras no les asignamos la carga de trabajo.

El valor de  $p_{min}$  se obtiene con

$$p_{min} = \sum_{k=1}^{nATC_{completos}-1} \left( \frac{k \times nSlot}{nATC^2} \right), \quad (6.6)$$

donde no asignamos carga de trabajo a los controladores que se encuentran en las últimas filas, pero a los controladores que se encuentran en las primeras filas se les asigna la máxima carga de trabajo.

Para calcular los valores de  $p_{max}$  y  $p_{min}$  necesitamos conocer el valor de  $nATC_{completos}$ . Para su cálculo, se necesita la carga de trabajo total que se obtiene a partir de la sectorización. Esto nos proporciona un número de slots de trabajo que se deben repartir entre todos los controladores, y al que denominamos  $C_{total}$ . La expresión utilizada para la obtención de  $nATC_{completos}$  es la siguiente:

$$nATC_{completos} = \frac{C_{total}}{nSlot}. \quad (6.7)$$

Este término hace referencia al número de controladores necesarios que tienen que trabajar todo el turno sin descanso para poder cubrir la carga de trabajo ( $C_{total}$ ).

- **Segundo término:** Para analizar la reducción de la infactibilidad, vamos a contabilizar el número de condiciones de entorno totales que están incumpliendo, el cual hay que minimizar, y que denotamos por  $R_i$ .

Para convertirlo a forma de maximización y normalizarlo usamos

$$f_2 = \frac{(MaxR_i - R_i)}{MaxR_i}, \quad (6.8)$$

siendo  $MaxR_i$  el valor máximo que se puede alcanzar incumpliendo todas las restricciones.

Tenemos que tener en cuenta dos situaciones:

- Turno de noche. En él aplican todas las restricciones, por lo que  $MaxR_i$  se calcula de la siguiente manera: existe un total de catorce restricciones, ocho (restricción 3, 5, 6, 9, 10, 11, 12 y 14) que pueden incumplirse  $nATC$  veces, dos restricciones (restricción 4 y 13) que se pueden incumplir  $2 \times nATC$  veces

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

y cuatro restricciones (restricción 1, 2, 7, 8) que se pueden incumplir  $nATC$  veces, pero en estas últimas cuatro se considera el grado de incumplimiento y se multiplica cada vez que no se cumple en cada slot del turno por  $1/nSlot$ . Esto da lugar la siguiente expresión:  $nSlot \times nATC \times (1/nSlot)$ .

En el caso del turno de noche la expresión a usar es la siguiente

$$MaxR_i = 8nATC + 2 \times 2nATC + (4(nATC + nSlot \times \frac{1}{nSlot} \times nATC)),$$

es decir,

$$MaxR_i = 20nATC. \quad (6.9)$$

- Turnos diurnos (mañana y tarde). No aplica la restricción 4, por lo que la expresión utilizada es la siguiente

$$MaxR_i = 8nATC + 1 \times 2nATC + (4(nATC + nSlot \times \frac{1}{nSlot} \times nATC)),$$

es decir,

$$MaxR_i = 18nATC. \quad (6.10)$$

Dicho lo anterior, la función objetivo completa queda de la siguiente forma:

$$f = \begin{cases} f_1 \times \mu_1 + f_2 \times \mu_2, & \text{si } nATC > nATC_{disp} \\ f_2, & \text{si } nATC \leq nATC_{disp} \end{cases}, \quad (6.11)$$

donde  $\mu_1$  y  $\mu_2$  son los pesos que expresan la importancia relativa de los dos términos de la función, cuyos valores son 0.2 y 0.8, respectivamente, habiendo realizado las pruebas pertinentes con otras combinaciones de valores.

Cuando el número de controladores disponibles ( $nATC_{disp}$ ) sea menor que el número de controladores usados en la solución, se utilizarán los dos términos en la función. En caso contrario, la función es  $f_2$ , se utiliza para simplemente buscar una solución con ese número de controladores, pero factible.

Tabla 6.2: Instancias más representativas del problema

	Centro de control	Núcleos	Turno	1º Núcleo		2º Núcleo	
				Sectorización	Nº ATCos	Sectorización	Nº ATCos
Caso 7	Barcelona	Ruta W	MC	3, 4, 5, 6, 5	16		
Caso 13	Barcelona	Ruta E	MC	3, 4, 5, 4	14		
Caso 23	Barcelona	Ruta W	ML	1, 4, 5, 7	18		
Caso 29	Barcelona	Ruta W/E	TC	5, 4, 2	14	4, 2	14
Caso 65	Madrid	Ruta 2	TL	7, 8, 7, 8, 4	22		

### Temperatura inicial, función de enfriamiento y número de iteraciones hasta el descenso de la temperatura

Existen multitud de métodos para calcular la temperatura inicial, pero no existe ninguno que ofrezca una clara ventaja sobre los otros, sino que depende del problema a resolver. Para esto se han realizado ciertas pruebas resolviendo un conjunto de instancias del problema. No obstante, el análisis de uno de los parámetros del SA unilateralmente no ofrece información fiable ya que puede estar condicionado a otros parámetros. Normalmente se realiza un estudio a pares de los parámetros con la intención de obtener mejor información para la elección de estos.

En este caso, hemos agrupado los parámetros principales que influyen en la variación de la temperatura durante la ejecución del algoritmo. Estos son la temperatura inicial, la función de enfriamiento y el número de iteraciones hasta el descenso de la temperatura. Primero se realiza un estudio para escoger el método más adecuado para el cálculo de la temperatura inicial. Posteriormente se realiza el estudio conjunto para la elección de la función de enfriamiento y el número de iteraciones hasta el descenso de la temperatura.

Para la realización de este estudio se ha tomado una muestra representativa de las instancias del problema, los cuales se encuentran descritos en la Tabla 6.2. Los resultados del cálculo de la temperatura inicial se muestran en la Tabla 6.3. Mientras que los resultados del segundo estudio son mostrados en la Tabla 6.4.

Previamente a la explicación del estudio realizado procedemos a exponer las instancias utilizadas y justificar la elección de las mismas. Estas instancias han sido escogidas por ser los casos más representativos y darse situaciones que puedan poner a prueba el correcto

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

Tabla 6.3: Resultados del ajuste de la temperatura inicial para la segunda fase

Parámetros	Tiempo de cómputo (min)					Temperatura escogida					
	C.7	C.13	C.23	C.29	C.65	C.7	C.13	C.23	C.29	C.65	
Temp. inicial	Ben-Ameur	0.095	0.069	0.112	0.137	0.103	0.0721	0.07129	0.08379	0.05532	0.07861
	Johnson	0.18	0.134	0.208	0.242	0.215	0.01277	0.01895	0.01131	0.01123	0.01124
	Kirkpatrick	<b>0.07</b>	<b>0.053</b>	<b>0.079</b>	<b>0.104</b>	<b>0.082</b>	0.0122	0.02452	0.01447	0.0327	0.01755

Tabla 6.4: Resultados del ajuste del número de iteraciones y función de enfriamiento para la segunda fase

		Tiempo de cómputo (min)					Nº soluciones factibles				
		C.7	C.13	C.23	C.29	C.65	C.7	C.13	C.23	C.29	C.65
Nº iteraciones	C. Adaptativas	7.96	6.72	<b>1.66</b>	10.52	12.98	3.5	<b>2</b>	<b>7.67</b>	<b>2</b>	1
	C. Estáticas	7.09	5.83	2.39	9.70	12.52	3.67	<b>2</b>	<b>7.67</b>	<b>2</b>	1
	CutOff	<b>6.32</b>	<b>5.40</b>	1.8	<b>8.04</b>	<b>12.14</b>	<b>3.83</b>	1.83	<b>7.67</b>	<b>2</b>	<b>1.33</b>
Descenso Temp	Kirkpatrick	<b>7.03</b>	<b>5.9</b>	2.48	11.74	<b>12.09</b>	<b>3.78</b>	1.89	<b>7.78</b>	<b>2</b>	0.89
	Adaptativo	7.21	6.06	<b>1.42</b>	<b>8.02</b>	13.47	3.56	<b>2</b>	7.56	<b>2</b>	<b>1.33</b>

funcionamiento del algoritmo. Las instancias son las siguientes:

1. Caso 7 (C.7): Esta instancia pertenece a un turno de mañana del centro de control de Barcelona. Para la creación del horario de trabajo disponemos únicamente de 16 controladores y debemos cubrir un período de aumento de tráfico aéreo en el que se encuentran abiertos 6 sectores. Debido a las restricciones del problema, sabemos que el número mínimo de controladores necesarios para cubrir 6 sectores son 16, por lo que durante este período los controladores únicamente tienen los descansos obligatorios. Esta situación junto con la apertura y cierre de sectores a lo largo del turno complica la resolución del problema.
2. Caso 13 (C.13): Esta instancia pertenece a un turno de mañana del centro de control de Barcelona. Para la creación del horario de trabajo disponemos de 14 controladores. Es similar a la instancia anterior, pero en este caso disponemos de dos controladores menos, con una carga de trabajo ligeramente menor.
3. Caso 23 (C.23): Esta instancia pertenece a un turno de mañana del centro de control de Barcelona. Para la creación del horario de trabajo disponemos únicamente de 18 controladores y debemos cubrir un período de aumento de tráfico aéreo en el que

se encuentran abiertos 7 sectores. Si se mantuvieran los 7 sectores abiertos durante un largo período de tiempo el problema no tendría solución factible, ya que necesitaríamos como mínimo 19 controladores. En este caso, los 7 sectores permanecen abiertos durante 5 horas, por lo que la resolución de esta instancia es cuanto menos complicada. Este es el principal reto al que nos enfrentamos a la hora de resolver el problema, pero también contamos con otro inconveniente, la apertura de tres sectores simultáneamente. Esto no es muy común ya que obliga a realizar numerosos cambios en sala, pero en algunos casos esto puede ser necesario.

4. Caso 29 (C.29): Esta instancia pertenece a un turno de tarde del centro de control de Barcelona. Para la creación del horario de trabajo disponemos únicamente de 14 controladores para el núcleo Este y otros 14 controladores para el núcleo Oeste. En muchas ocasiones los turnos de los distintos núcleos se resuelven por separado, pero en algunos centros de control los núcleos tienen sectores en común y en estos casos resolver los horarios juntos puede ofrecer algunas ventajas.

Resolver el horario para ambos núcleos es más complicado, primero porque doblamos las dimensiones del problema, y el segundo motivo es que los controladores de un núcleo solo pueden cubrir sectores pertenecientes a su núcleo. Esto genera muchas limitaciones a la hora de realizar los turnos. Por este motivo se ha escogido esta instancia para su resolución.

5. Caso 65 (C.65): Esta instancia pertenece a un turno de tarde del centro de control de Madrid. Para la creación del horario de trabajo disponemos de 22 controladores. El principal reto de este problema no solo se encuentra en sus dimensiones, debido a que debemos cubrir una sectorización con 7 y 8 sectores abiertos simultáneamente, sino al cierre y apertura realizado en varias ocasiones de los sectores. Esto provoca cambios en sala constantemente y dificulta cumplir con las condiciones laborales de los controladores.

Se han escogido varias opciones distintas para cada uno de los parámetros, por lo que cada instancia del problema es ejecutada un considerable número de veces con distintos parámetros. Todos los métodos estudiados han sido explicados previamente en la Sección 5.2.1.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

Para el cálculo de la temperatura inicial se han usado los métodos de *Ben-Ameur, Johnson* y *Kirkpatrick*.

Para conseguir un descenso adecuado de la temperatura, se han utilizado las funciones de enfriamiento de *Kirkpatrick* y una función de enfriamiento adaptativa mostrada en la Expresión 5.16.

Para el cálculo del número de iteraciones hasta el descenso de la temperatura ( $L$ ) se ha probado a utilizar *cadenas adaptativas*, *cadenas estáticas* y el método *cutoff*.

Como método para el cálculo de temperatura inicial, se ha escogido el propuesto por Kirkpatrick. El motivo se debe a que para las instancias resueltas es el método más rápido, como se muestra en la Tabla 6.3. También se observa una mejor adaptación a los casos concretos para el cálculo de la temperatura inicial comparándolo con los otros métodos.

Para la elección del conjunto de parámetros final se han tenido en cuenta varios factores. El primero, y más importante, la resolución del problema, el segundo es el tiempo de cómputo asociado, y el tercero es el número de soluciones factibles alcanzadas. Como cabría esperar, no existe un conjunto que sea el mejor en todos los factores por lo que la elección no es clara. Sin embargo, analizando los resultados mostrados en la Tabla 6.4, se decidió usar los siguientes métodos:

1. Para el cálculo de las iteraciones hasta el descenso de temperatura se decidió utilizar el método de *cutoff*. De acuerdo con los resultados obtenidos, es el método más rápido y además, obtiene unos resultados mejores que el resto en base al número de soluciones.
2. Como función de enfriamiento, elegimos la función propuesta por Kirkpatrick. Ambas ofrecen resultados muy similares. Pero esta función supera ligeramente al método adaptativo en tiempo y en número de soluciones factibles. La función requiere la estimación del parámetro  $\alpha$ , establecido en 0,95 según la referencia de Hajek [49].

#### Definición del entorno de una solución

La gran importancia de la correcta elección de los movimientos para el buen funcionamiento del algoritmo provoca que sea uno de los puntos más importantes. Una mala

elección de soluciones del entorno pueden provocar que el algoritmo no funcione adecuadamente.

Uno de los principales inconvenientes es la falta de una regla que garantice su correcto funcionamiento para cualquier problema. Por supuesto hay recomendaciones y consejos, que siguiéndolos, aumentan las probabilidades de su funcionamiento. Cumpliendo todas estas recomendaciones se han generado distintos movimientos y probado en distintas instancias del problema.

Procedemos a describir los movimientos más prometedores:

1. *Entorno 3*: Para la generación de soluciones dentro de este entorno se utiliza el siguiente movimiento. Este movimiento está ligado a la creación de soluciones iniciales. Esto se debe a que cada una de las soluciones generadas tienen un tamaño de intervalo inicial, y este va desde 3 a 12 slots, ambos incluidos. Por ello, el tamaño de los intercambios de trabajo que se realicen, serán con múltiplos del mínimo común divisor de estos números, siempre entre 3 y 12.

Por ejemplo, si la solución inicial tiene un intervalo inicial de 8, los posibles tamaños serán 4, 8 y 12. En el caso de que el intervalo inicial sea de 6, los tamaños serán 3, 6, 9 y 12. Comenzamos con la explicación del movimiento en los siguientes pasos:

- a) Se escoge un primer controlador aleatoriamente, el cual cederá un período de trabajo a un segundo controlador.
- b) Una vez tenemos el primer controlador, se obtienen sus períodos de trabajo continuados, y se escoge uno aleatoriamente. Después, se modifica la longitud del intervalo entre los posibles tamaños en función de la solución inicial. En el caso de que la longitud del período sea menor que la escogida, esta no varía. En caso contrario, se obtendrán únicamente los primeros slots del período de trabajo.
- c) En el siguiente paso se obtiene un segundo controlador de forma aleatoria. Al generar la solución se comprueba que en el turno de los controladores no se mezclen sectores pertenecientes a distintos núcleos. En caso de que esta condición no se cumpla, se está generando un individuo no válido, y repite el

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

proceso modificando el segundo controlador hasta obtener una solución que la cumpla, o probar con todos los controladores.

- d) Si no se obtiene ningún individuo válido, es decir, que cumpla la condición, se modifica el tamaño del intervalo escogiendo nuevamente una longitud aleatoria entre los posibles tamaños. El número escogido anteriormente es eliminado para evitar la repetición del mismo proceso, y nuevamente volvemos al paso c). En el improbable caso de que se hayan probado todos los posibles tamaños del intervalo, y no se haya obtenido una solución válida, modificamos el primer controlador junto con el período de trabajo.
  - e) Para esto, repetimos desde el paso a) (sin escoger nuevamente los controladores ya utilizados).
2. *Entorno 4*: Para la generación de soluciones dentro de este entorno se utiliza el *Entorno 3* y el movimiento que definiremos a continuación. Se escoge entre ambos con cierta probabilidad. Por cada iteración se escoge el *Entorno 3* con una probabilidad del 90 % y el nuevo movimiento con un 10 %. A continuación, explicamos el movimiento en los pasos siguientes:

- a) Se escoge un primer controlador aleatoriamente.
  - b) Se obtiene un segundo controlador de forma aleatoria.
  - c) Se dividen los turnos de ambos controladores en dos partes por el mismo punto de corte y se intercambian la segunda parte del turno entre los controladores. Al generar la solución se comprueba que, en el turno de los controladores, no se mezclen sectores pertenecientes a distintos núcleos. En caso de que esta condición no se cumpla, se está generando un individuo no válido, y repite de nuevo el proceso modificando ambos controladores y el punto de corte hasta obtener una solución que cumpla dicha condición.
3. *Entorno 6*: Para la generación de soluciones dentro de este entorno, se aplica el *Entorno 3* dos veces consecutivas.
4. *Entorno 12*: Para la generación de soluciones dentro de este entorno, se aplica un concepto que debemos explicar previamente, la *rejilla* o *malla*.

CAPÍTULO 6

Dada una solución inicial, se crea una malla que divide la solución en distintos conjuntos de slots. Esta malla establece los cortes en los distintos puntos donde existen cambios de trabajo a descanso o de un sector a otro. En la Figura 6.19 observamos un ejemplo de cómo se crearía una malla a partir de una solución inicial.

Figura 6.19: Solución inicial

La malla establece los puntos de inicio y fin de los distintos conjuntos de slots que permiten permutaciones a lo largo de las iteraciones del algoritmo, y el movimiento se define como el intercambio del conjunto de slots entre dos filas distintas. A continuación, explicamos el movimiento en detalle en los siguientes pasos:

- a) Se escoge un primer controlador aleatoriamente, el cual cederá un período de trabajo a un segundo controlador.
  - b) Una vez tenemos el primer controlador, se obtienen un conjunto de slot aleatorio dentro de los límites definidos por la malla generada a partir de la solución inicial.
  - c) En el siguiente paso se obtiene un segundo controlador de forma aleatoria. Al generar la solución se comprueba que en el turno de los controladores no se mezclen sectores pertenecientes a distintos núcleos. En caso de que esta condición no se cumpla, se está generando un individuo no válido, y repite el proceso modificando el segundo controlador y el intervalo hasta obtener una solución que la cumpla, o probar con todos los controladores.
  - d) En el improbable caso de que no se haya obtenido una solución válida, modificamos el primer controlador. Para esto, repetimos desde el paso a) (sin escoger nuevamente los controladores ya utilizados).

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

5. *Entorno 15*: Para la generación de soluciones dentro de este entorno, se aplica el *Entorno 12*, escogiendo de uno a tres intervalos consecutivos. El número de intervalos que se escoge es aleatorio.
6. *Entorno 17*: Para la generación de soluciones dentro de este entorno, se aplica el *Entorno 15*. La principal diferencia de este entorno con el *Entorno 15* se encuentra en que la rejilla se modifica cada cierto número de iteraciones reduciendo el tamaño de uno de los intervalos escogido aleatoriamente. Cada vez que dividimos un intervalo se comprueba que ninguno de los dos intervalos resultantes sean menores a tres slots; en caso contrario, se escoge otro intervalo. Si no existe ninguno, la rejilla no se modificará.
7. *Entorno 25*: Para la generación de soluciones dentro de este entorno, se aplica el *Entorno 17*. La diferencia de este entorno con el *Entorno 17* se encuentra en que si la rejilla se encuentra muy fragmentada (los intervalos son muy pequeños) permitimos aumentar el número de intervalos que se escogen hasta cuatro intervalos. Para observar la fragmentación de la rejilla utilizamos la media de la longitud de los intervalos.

En esta lista también se pueden añadir los *Entornos 1 y 2*, generados para la primera aproximación del problema y explicados en la Sección 6.2.2. No obstante, los resultados de estos entornos son inferiores en tiempo y eficacia a los entornos actuales.

El resultado de las pruebas realizadas con los distintos movimientos se exponen en la Tabla 6.5. Con el objetivo de mostrar con mayor claridad los datos, los *Entornos 1, 2 y 4* han sido descartados previamente debido a su inferior rendimiento en la resolución de los problemas.

En la Tabla 6.5, se observa que ninguno de los entornos consigue la resolución de todos los casos, los resultados no permiten tomar una decisión clara sobre el mejor entorno. El *Entorno 6* es el único que ofrece una solución factible al caso 29, mientras que solo los *Entornos 12, 15 y 17* ofrecen solución factible al caso 23.

Se observa que el *Entorno 15* obtiene mejores tiempos de cómputo, mientras que los *Entornos 17 y 25* obtienen un mayor número de soluciones factibles.

Tabla 6.5: Resultados del estudio de los distintos entornos para la segunda fase

Entornos	Tiempo de cómputo (min)					Nº sol. factibles				
	C.7	C.13	C.23	C.29	C.65	C.7	C.13	C.23	C.29	C.65
<b>Entorno 3</b>	4.21	3.68	-	-	7.02	3	2	0	0	2
<b>Entorno 6</b>	5.37	5.02	-	<b>8.17</b>	9.22	3	2	0	<b>2</b>	1
<b>Entorno 12</b>	3.63	2.65	3.98	-	4.99	3	4	1	0	4
<b>Entorno 15</b>	<b>2.86</b>	1.97	<b>3.58</b>	-	<b>3.64</b>	5	5	1	0	5
<b>Entorno 17</b>	4.09	2.11	6.87	-	4.71	<b>6</b>	<b>9</b>	<b>2</b>	0	<b>8</b>
<b>Entorno 25</b>	3.91	<b>1.95</b>	-	-	4.44	<b>6</b>	<b>9</b>	0	0	<b>8</b>

Analizando en profundidad los distintos entornos y el resto de instancias resueltas, comprobamos que el *Entorno 6* muestra unos resultados muy positivos en las instancias de mayores dimensiones, las que involucran un mayor número de controladores y sectores. También se observan buenos resultados en la resolución de instancias con más de un núcleo.

Por otro lado, el *Entorno 15* muestra resultados prometedores en instancias más reducidas, donde la carga de trabajo es elevada en comparación al número de controladores disponibles, es decir, el descanso se reduce prácticamente al mínimo necesario por las condiciones laborales. En cuanto a tiempo de cómputo también podemos observar que el *Entorno 15* es el más rápido.

Una vez realizado el estudio se ha tomado la decisión de comprobar las características de la instancia a resolver automáticamente y previamente a su resolución para utilizar en esta el *Entorno 6* o el *Entorno 15*.

### Función de aceptación

El problema que resolvemos es complejo, y el tiempo de cómputo de las distintas ejecuciones es elevado. Esto se debe a varios factores, entre ellos la función objetivo. Por esto, se ha decidido utilizar la distribución de Boltzman (Sección 5.2.1) debido a que es una de las que mejor resultados ofrece, y una de las más rápidas para la mayoría de problemas, además de ser la más estudiada y usada en la literatura para este tipo de problemas.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

#### Condición de parada

Dado que el objetivo del algoritmo en la segunda fase es obtener soluciones factibles, es decir, que no se incumpla ninguna restricción, la primera condición de parada es esta, lo que es equivalente a alcanzar el óptimo. En caso de que no se alcance el óptimo, existen dos condiciones que deben cumplirse para que el algoritmo detenga su ejecución:

1. La primera condición se basa en que tiene que existir un porcentaje de mejora del valor objetivo cada ciertas iteraciones para que el algoritmo continúe funcionando. En caso de que no se produzca esta mejora, se considera que el algoritmo se encuentra estancado.
2. La segunda condición evita que el algoritmo se detenga demasiado rápido, esto se realiza comprobando que el algoritmo genere un porcentaje de soluciones mejores a la solución actual durante un número determinado de iteraciones, si este porcentaje desciende de un umbral, se activa la condición de parada.

El hecho de utilizar dos funciones objetivo ponderadas puede provocar que una de estas necesite empeorar para que la otra mejore. En este caso, las soluciones iniciales cumplen casi todas las restricciones, pero se utiliza un número muy elevado de controladores. En el proceso de mejora se deben incumplir ciertas restricciones para disminuir el número de controladores. Esto provoca que el valor objetivo fluctúe durante las primeras iteraciones sin una mejora clara, y esta condición evita que la primera detenga el algoritmo antes de tiempo.

El cálculo de los parámetros de las condiciones de parada se ha realizado mediante la experimentación. Se han ejecutado múltiples instancias del problema un número determinado de ejecuciones, permitiendo avanzar la ejecución hasta que el valor objetivo de la solución quede estancado. Conociendo estos datos se ha realizado una media de los porcentajes de mejora durante las últimas iteraciones en cada una de las instancias. Esto nos proporciona el porcentaje de mejora medio que se obtiene antes de que el valor objetivo de una solución se estanke. Otro parámetro que también se calcula es el número de iteraciones. Este debe ser suficientemente elevado para que el algoritmo no se detenga antes de tiempo por un estancamiento temporal.

### 6.3.3. Fase 3: SA multicomienzo para obtención de soluciones óptimas

Una vez se ha obtenido una solución factible, comenzamos con la tercera fase del algoritmo de solución. Lo primero que cabe destacar en esta fase es que todas las soluciones visitadas durante su ejecución deben ser factibles, por lo que, a la hora de generar soluciones, es lo primero que se comprueba.

El número de soluciones iniciales dependerá del número de soluciones factibles obtenidas en la segunda fase y el objetivo en esta fase es la optimización de las soluciones en base a cuatro objetivos.

#### Función objetivo

La función objetivo está formada por cuatro objetivos, todos se encuentran en forma de maximización y normalizados con valores comprendidos entre 0 y 1. El primero de los objetivos optimiza un conjunto de condiciones deseables, como el tiempo óptimo de trabajo y descanso continuado, o el equilibrio entre el trabajo en las posiciones de planificador y ejecutivo. El segundo objetivo procura que las soluciones sean compactas y no estén completamente fragmentadas. El motivo es facilitar su comprensión y aceptación por parte del equipo de trabajo, además de evitar posibles confusiones en la lectura de los turnos de trabajo. El tercer objetivo está formado por dos sub-objetivos. El primero procura que los controladores mantengan sus acreditaciones de trabajo para poder cubrir ciertos sectores y el segundo incentiva reducir el número de cambios en sala unificando períodos de descanso. El cuarto y último objetivo incentiva el equilibrio de la carga de trabajo entre todos los controladores del turno.

1. El **primer objetivo** es cumplir una serie de restricciones deseables asociadas a las condiciones laborales de los controladores. La función objetivo que se utilizará para cumplir dicha meta es una suma ponderada a partes iguales del conjunto de restricciones incumplidas, dando a cada restricción el mismo peso.
  - a) Primera condición deseable: *Tiempo óptimo de trabajo en posición* (45 minutos). Es el tiempo que un controlador se encuentra sin cambio de sector y tipo

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

de control (ejecutivo/planificador).

Para comprobar si esta condición se cumple o no y su grado de incumplimiento se realiza lo siguiente: por cada fila de la matriz se calcula el número de minutos en los que el controlador se mantiene en el mismo sector cubriendo la misma posición de forma continuada, y se anota la diferencia entre este valor y el óptimo.

Posteriormente, sumamos todas las diferencias de todos los intervalos. Como ya hemos mencionado, esto se realiza por cada fila y se suma el total de diferencias de todas las filas.

Por último, se divide entre el número de controladores para obtener la media.

Para realizar la comprobación utilizamos

$$v_1 = \frac{\left( \sum_{k=0}^{nATC} \sum_{nIntervalos}^{i=0} |posOpt - l_i| \right)}{nATC}, \quad (6.12)$$

donde  $posOpt$  es el tiempo óptimo de trabajo (45 minutos) y  $l_i$  la longitud en minutos del intervalo  $i$ . Para normalizar el resultado se utiliza

$$vn_1 = \frac{v_{1max} - v_1}{v_{1max}}, \quad (6.13)$$

siendo  $v_{1max}$  el valor máximo de  $v_1$  y calculado con

$$v_{1max} = |posOpt - posMin| \times 8 \times \frac{nSlot}{30}, \quad (6.14)$$

donde  $posMin$  es el tiempo mínimo que debe permanecer en una posición un controlador para no incumplir ninguna restricción. Al dividir el valor de  $nSlot$  entre 30 slots, estamos calculando el número de veces que se puede dar un período de trabajo de dos horas (24 slots) continuado de un período de 30 minutos (6 slots) de descanso, y el motivo de multiplicar por 8 es que, en este período, se pueden encontrar 8 intervalos de trabajo en distintas posiciones sin incumplir ninguna restricción.

- b) Segunda condición deseable: *Tiempo óptimo de trabajo entre descansos* (90

minutos).

Para constatar si esta condición se cumple o no y conocer el grado de incumplimiento se realiza lo siguiente: por cada fila se calcula la longitud de cada intervalo de trabajo (en minutos) y se anota la diferencia entre esta y el óptimo de trabajo (*trabOpt*), sumando, al final, las diferencias entre todos los intervalos. Esto se realiza por cada una de las filas de la matriz, y después se divide entre el número de filas para obtener la media. La función previa normalización utilizada es

$$v_2 = \frac{\left( \sum_{k=0}^{nATC} \sum_{nIntervalos}^{i=0} |trabOpt - l_i| \right)}{nATC}, \quad (6.15)$$

y para normalizar este valor se necesita conocer el valor máximo de estas diferencias, el cual obtenemos con

$$v_{2max} = |trabOpt - trabMin| \times \frac{nSlot}{6}, \quad (6.16)$$

siendo, *trabMin* el trabajo mínimo posible sin incumplir ninguna restricción (15 minutos). En este caso dividimos entre 6 el número de slots debido a que el trabajo mínimo son 3 slots y el descanso mínimo también son 3, por lo que esta restricción se podrá incumplir por cada 6. Por último, para la normalización se utiliza la expresión

$$vn_2 = \frac{v_{2max} - v_1}{v_{2max}}. \quad (6.17)$$

- c) Tercera condición deseable: *El porcentaje de tiempo que un controlador trabaja en posiciones ejecutivas debe estar comprendido entre el 40% y el 60% del trabajo total realizado (sin tener en cuenta los descansos).*

Para verificar si esta condición se cumple o no y su grado de incumplimiento, se realiza lo siguiente: por cada fila de la matriz  $k$ , se divide el tiempo en que se encuentra un controlador trabajando en la posición de ejecutivo entre el tiempo de trabajo total, obteniendo el porcentaje de trabajo en posición ejecutiva en dicha fila ( $pEje_k$ ).

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

Si este es menor que 40 % o mayor que 60 %, se realiza la diferencia al valor más próximo y se divide por la diferencia máxima, es decir, 0.4 (40 %) para la normalización.

Después, se suman las diferencias entre todas las filas y se divide entre el número de filas para mantener normalizado el valor.

La Expresión 6.18 muestra la ecuación perteneciente a la tercera condición. Se ha estudiado la posibilidad de penalizar la función en base a la desviación típica entre los diferentes turnos de la matriz, para así tratar de evitar desequilibrios entre los controladores. No obstante, se han realizado pruebas que descartan esta posibilidad por aumentar el tiempo de cómputo sin mostrar alguna mejora.

$$vn_3 = \frac{1}{0.4} \times \sum_{k=0}^{nATC} \begin{cases} \frac{|pEje_k - 0.4|}{nATC} & \text{si } pEje_k < 0.4 \\ \frac{|pEje_k - 0.6|}{nATC} & \text{si } pEje_k > 0.6 \end{cases}. \quad (6.18)$$

Una vez tenemos los valores de cada una de las condiciones deseables, los ponderamos equitativamente y obtenemos un valor objetivo normalizado entre 0 y 1 mediante

$$f_1 = vn_1 \times \mu_1 + vn_2 \times \mu_2 + vn_3 \times \mu_3, \quad (6.19)$$

donde los pesos  $\mu_1, \mu_2$  y  $\mu_3$  tienen el mismo valor de 0.33.

2. El **segundo objetivo** es mantener una estructura/apariencia similar a la que se encuentra en los estadillos. A falta de una medida de similitud proporcionada por un experto, se entiende como estructura similar a un estadillo las soluciones más compactas y menos fragmentadas, que tengan más agrupados los sectores donde se trabaja, así como los descansos. Este es un proceso de compactación que tiene el objetivo de facilitar la aceptación y comprensión de las soluciones propuestas. Con el fin de cumplir este objetivo, se utiliza la expresión

$$v = \sum_{k=1}^{nATC-1} \sum_{j=1}^{nSlot-1} \begin{cases} 1, & \text{si } slot_{kj} = slot_{k+1,j} \\ 1, & \text{si } slot_{kj} = slot_{kj+1} \end{cases}. \quad (6.20)$$

En esta función se comprueba para cada  $slot_{kj}$ , si el slot de su derecha  $slot_{kj+1}$  tiene el mismo contenido y, si es así, suma uno. También comprueba si el slot de debajo  $slot_{k+1j}$  tiene el mismo contenido y, si es así, vuelve a sumar uno. En caso de que ambos sean distintos, no se añadirá ningún valor. Para obtener el valor máximo teórico utilizamos

$$v_{max} = (nSlot - 1) \times (nATC - 1) \times 2. \quad (6.21)$$

Este es un valor máximo teórico obtenido sin tener en cuenta las restricciones, por lo que el valor de la función normalizada nunca llegará a uno, siendo este el valor máximo teórico. Una vez tenemos este valor, ya podremos normalizar el objetivo con

$$f_2 = 1 - \frac{v_{max} - v}{v_{max}}. \quad (6.22)$$

3. El **tercer objetivo** está compuesto por dos sub-objetivos y ambos tienen el mismo peso sobre

$$f_3 = f_{3.1} \times \mu_{3.1} + f_{3.2} \times \mu_{3.2}. \quad (6.23)$$

- a) El primer sub-objetivo consiste en *minimizar el número de intervalos de descanso* (no hace referencia a su duración). Con esto se pretende conseguir minimizar el número de cambios en sala, ya que minimizando los intervalos de descanso, no se pretende reducir el descanso total, sino agruparlo para evitar un movimiento excesivo de los controladores. La función objetivo propuesta consiste en contabilizar el número de períodos de descanso para cada uno de los turnos de la matriz.

El valor mínimo de la función es el número de turnos  $v_{min} = nATC$ , ya que en todas debe existir al menos un descanso, mientras que el valor máximo se obtiene con

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

$$v_{max} = \frac{nSlot \times nATC}{6}, \quad (6.24)$$

siendo el motivo de dividir entre seis slots la suma de los intervalos de trabajo y descanso mínimos, por lo que esto nos daría el número máximo de intervalos por cada turno de la matriz.

Con los valores mínimos y máximos, normalizamos y obtenemos

$$f_{3.1} = \frac{v_{max} - v}{v_{max} - v_{min}}. \quad (6.25)$$

- b) El segundo tiene como objetivo que los controladores no pierdan la acreditación para cubrir cierto tipo de sectores. Esto puede suceder si durante un largo período de tiempo no cubren determinados sectores. Por tanto, el segundo sub-objetivo consiste en *maximizar el número de sectores elementales que cubren los controladores*.

Cada sector está formado por uno o más sectores elementales. El valor mínimo de la función es el número de turnos, suponiendo que cada controlador únicamente cubra un sector,  $v_{min} = nATC$ , y el valor máximo de la función es el número total de sectores elementales multiplicado por el número total de controladores, suponiendo que todos los controladores cubran todos los sectores elementales en algún momento del turno  $v_{max} = nATC \times nSectoresElementales$ .

Una vez tenemos los valores máximos y mínimos de la función calculamos el número de sectores elementales que cubren los controladores con

$$v = \sum_{k=1}^{nATC} \sum_{j=1}^{nSlot} 1 \quad \text{si } nuevoSectorElemental = \text{verdadero}, \quad (6.26)$$

siendo *nuevoSectorElemental* la comprobación de que el controlador  $k$  no ha cubierto en ningún otro slot  $j$  el sector elemental. Una vez calculado el valor  $v$ , procedemos a normalizarlo y obtenemos

$$f_{3.2} = \frac{v_{max} - v}{v_{max} - v_{min}}. \quad (6.27)$$

4. El **cuarto objetivo** es realizar una distribución de la carga de trabajo equilibrada entre todos los controladores. Para medir este objetivo se utilizará la desviación estándar de los minutos de trabajo de los turnos de la matriz.

El valor máximo de la desviación estándar (sin tener en consideración las restricciones del problema) es igual a la media. Por ejemplo, si tenemos una carga de trabajo de 200 slots, y contamos con 4 controladores que trabajan durante un máximo de 100 slots, la media será de 50 slots de trabajo. Para maximizar la varianza se repartirán 100 slots de trabajo a 2 controladores y 0 a los otros 2, lo cual resulta una varianza de  $50^2$  y al realizar la desviación típica un valor de 50, lo que es idéntico a la media. Por tanto, la función utilizada para el cuarto objetivo es

$$f_4 = \frac{ds_{max} - ds}{ds_{max}}. \quad (6.28)$$

Una vez se calculan los cuatro objetivos, se realiza una ponderación en base a una serie de pesos, utilizando

$$f_{total} = f_1 \times \mu_1 + f_2 \times \mu_2 + f_3 \times \mu_3 + f_4 \times \mu_4, \quad (6.29)$$

donde los pesos  $\mu_1, \mu_2, \mu_3$  y  $\mu_4$  son obtenidos mediante el uso del *método ROC* [14]. El método ROC (*rank-order centroid*) tiene una lógica teórica sólida y parece funcionar mejor que otros métodos basados en ordenación. Este método utiliza la siguiente expresión para asignar los pesos en función del orden de importancia de los objetivos:

$$\mu_i = \frac{\sum_{j=i}^n 1/j}{n}, i = 1, \dots, n, \quad (6.30)$$

por tanto,

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

$$\begin{aligned}\mu_1 &= \frac{1 + 1/2 + 1/3 + 1/4}{4} = 0.52, \\ \mu_2 &= \frac{1/2 + 1/3 + 1/4}{4} = 0.27, \\ \mu_3 &= \frac{1/3 + 1/4}{4} = 0.15, \\ \mu_4 &= \frac{1/4}{4} = 0.06.\end{aligned}$$

La información ordinal ha sido proporcionada por los expertos de CRIDA y también ha aprobado el uso del método de asignación de pesos. En caso de que surja algún cambio e interesaría modificar los pesos, esto sería completamente viable e instantáneo.

#### **Temperatura inicial, función de enfriamiento y número de iteraciones hasta el descenso de la temperatura**

Para el ajuste de estos tres parámetros, se ha realizado el mismo proceso que se realizó en la segunda fase con el mismo conjunto de instancias. Los resultados podemos observarlos en las Tablas 6.6 y 6.7.

Tabla 6.6: Resultados del ajuste de la temperatura inicial para la tercera fase

	Parámetros	Tiempo de cómputo (min)					Temperatura escogida				
		C.7	C.13	C.23	C.29	C.65	C.7	C.13	C.23	C.29	C.65
Temp. inicial	Ben-Ameur	1.72	0.528	1.593	2.606	3.67	0.0683	0.0493	0.0881	0.0575	0.0647
	Johnson	0.107	0.043	0.156	0.174	0.199	0.012	0.0155	0.0073	0.0096	0.0094
	Kirkpatrick	0.033	0.016	0.065	0.07	0.08	0.0039	0.0057	0.0042	0.0035	0.0055

Como método para el cálculo de temperatura inicial, se ha escogido el propuesto por Johnson. Pese a no ser el método más rápido, consideramos que la diferencia de tiempo no es significativa comparado con los tiempos de resolución se están utilizando. El motivo se debe a que la temperatura inicial ofrecida por Kirkpatrick se estima demasiado baja, mientras que el método Ben-Ameur oscila unos tiempos de cómputo más elevados.

En esta fase, para la elección de los parámetros, el primer factor a tener en cuenta es el valor objetivo obtenido, estando ponderado del 0 al 1. Además, el segundo factor a tener en consideración es el tiempo de ejecución.

Como ha sucedido con la segunda fase, ninguno de los métodos resulta claramente mejor, por lo que se ha optado por elegir un equilibrio entre tiempo de ejecución y cali-

Tabla 6.7: Resultados del ajuste del número de iteraciones y función de enfriamiento para la tercera fase

		Tiempo de cómputo (min)					Valor de la función objetivo				
		C.7	C.13	C.23	C.29	C.65	C.7	C.13	C.23	C.29	C.65
Nº iteraciones	C. Adaptativas	108.59	40.67	<b>73.33</b>	171.85	262.27	0.859	0.8714	0.9376	<b>0.8533</b>	0.8407
	C. Estáticas	105.53	43.43	113.7	152.11	243.19	<b>0.8603</b>	<b>0.8733</b>	<b>0.9384</b>	0.8516	0.8343
	CutOff	<b>72.87</b>	<b>35.33</b>	100.81	<b>122.56</b>	<b>204.41</b>	0.86	0.8729	0.9376	0.8518	<b>0.842</b>
Descenso Temp	Kirkpatrick	98.32	<b>41.44</b>	<b>123.92</b>	161.25	254.07	<b>0.8631</b>	<b>0.8786</b>	<b>0.9385</b>	<b>0.8573</b>	<b>0.8506</b>
	Adaptativo	<b>93.01</b>	42	127.76	<b>155.32</b>	<b>246.12</b>	0.8564	0.8669	0.9373	0.8482	0.8315

Tabla 6.8: Resultados del estudio de los distintos entornos para la tercera fase

Entornos	Tiempo de cómputo (min)					Valor de la función objetivo				
	C.7	C.13	C.23	C.29	C.65	C.7	C.13	C.23	C.29	C.65
Entorno 3	9.14	8.11	129.49	140.17	26.37	0.849	0.8425	0.8383	0.8479	0.7856
Entorno 6	59.97	16.59	272.9	292.68	151.01	0.8641	0.8471	0.8378	0.8499	<b>0.823</b>
Entorno 12	<b>6.52</b>	6.87	<b>11.89</b>	<b>11.71</b>	<b>11.26</b>	0.8524	0.8545	0.8346	0.8496	0.7608
Entorno 15	6.95	7.55	16.61	16.41	22.87	<b>0.8781</b>	<b>0.87</b>	<b>0.8452</b>	<b>0.8638</b>	0.8163
Entorno 17	13	6.94	16.66	15.59	21.15	0.8718	0.8625	0.8357	0.8618	0.8161
Entorno 25	7.42	<b>6.5</b>	17.43	16.29	21.99	0.8763	0.8603	0.8383	0.859	0.8106

dad de las soluciones obtenidas. Casualmente, esto se consigue con los mismos métodos escogidos para la segunda fase. Para el cálculo del descenso de la temperatura se escoge el método propuesto por Kirkpatrick. Y para el cálculo del número de iteraciones, se escoge el método *cutoff*.

### Definición del entorno de una solución

En esta tercera fase los entornos probados han sido los mismos que para la segunda fase, y los resultados se muestran en la Tabla 6.8. Para medir el rendimiento de los distintos entornos se han utilizado dos criterios, el primero es el valor de la función objetivo, el cual se encuentra normalizado entre 0 y 1, y el segundo es el tiempo de cómputo del algoritmo.

El entorno que ofrece mejores resultados es el *Entorno 15*. Pero en cuanto a tiempo de cómputo, se ve superado por el *Entorno 12*. Pese a dar más importancia al tiempo en la segunda fase de la metodología de resolución, en este caso, vemos que las diferencias de tiempo son menos significativas, por lo que optaremos por escoger el *Entorno 15*.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

#### 6.3.4. Algoritmo alternativo: VNS

Con el objetivo de mejorar y contrastar los resultados obtenidos por la metodología de resolución, se ha propuesto sustituir el algoritmo basado en el recocido simulado por un algoritmo basado en la búsqueda en entornos variables (VNS), el cual ya ha sido expuesto en la Sección 5.3.

El algoritmo VNS se utilizará durante la segunda y tercera fase de la metodología de resolución propuesta para la versión completa del problema, en sustitución al recocido simulado. Para poder comprobar el correcto funcionamiento del algoritmo y comparar la calidad de las soluciones, se utiliza la misma función objetivo que la usada en el recocido simulado. Al utilizar la misma función objetivo es necesario que la matriz de turnos se ordene de menor a mayor en función de la carga de trabajo, ya que si no se realiza este proceso la función objetivo dejaría de tener validez y no cumpliría su propósito.

El algoritmo está basado en el *VNS descendente*, que utiliza búsqueda local en los distintos entornos para mejorar la solución actual. Los algoritmos de búsqueda local son propensos al estancamiento en óptimos locales, por esto el VNS básico aplica un método llamado *shaking*, que consiste en la sustitución de la solución actual del algoritmo por una nueva solución generada aleatoriamente dentro del entorno de la misma. El algoritmo de búsqueda local diseñado es menos propenso al estancamiento, ya que no es determinista y realiza una búsqueda parcial en el espacio de búsqueda, por lo que en este caso, al igual que el VNS descendente no es necesario aplicar el método *shaking*.

Para facilitar la comprensión del algoritmo, en la Figura 6.20 se muestra un diagrama de flujos que resume el funcionamiento general del mismo. En esta figura se muestra como el algoritmo comienza en la primera iteración, donde realiza la búsqueda local con el primer entorno y el período uno. En el caso de que se encuentre una solución mejor que la mejor solución disponible hasta el momento, se reinicia la iteración, el período, y el entorno. En el caso contrario, se aumenta el período hasta un máximo de dieciocho slots. Si el período supera los dieciocho slots, se avanza al siguiente entorno y se reinicia a uno el período. Si finalizamos la lista de cuatro entornos, se debe avanzar a la siguiente iteración y se reinicia la lista de entornos. Cuando se alcance la cuarta iteración (límite establecido), el algoritmo finaliza su ejecución. En el caso que nos encontremos durante

la segunda fase del método de resolución, el algoritmo puede finalizar antes de realizar las cuatro iteraciones. Esto sucede en el momento en que se alcance una solución factible.

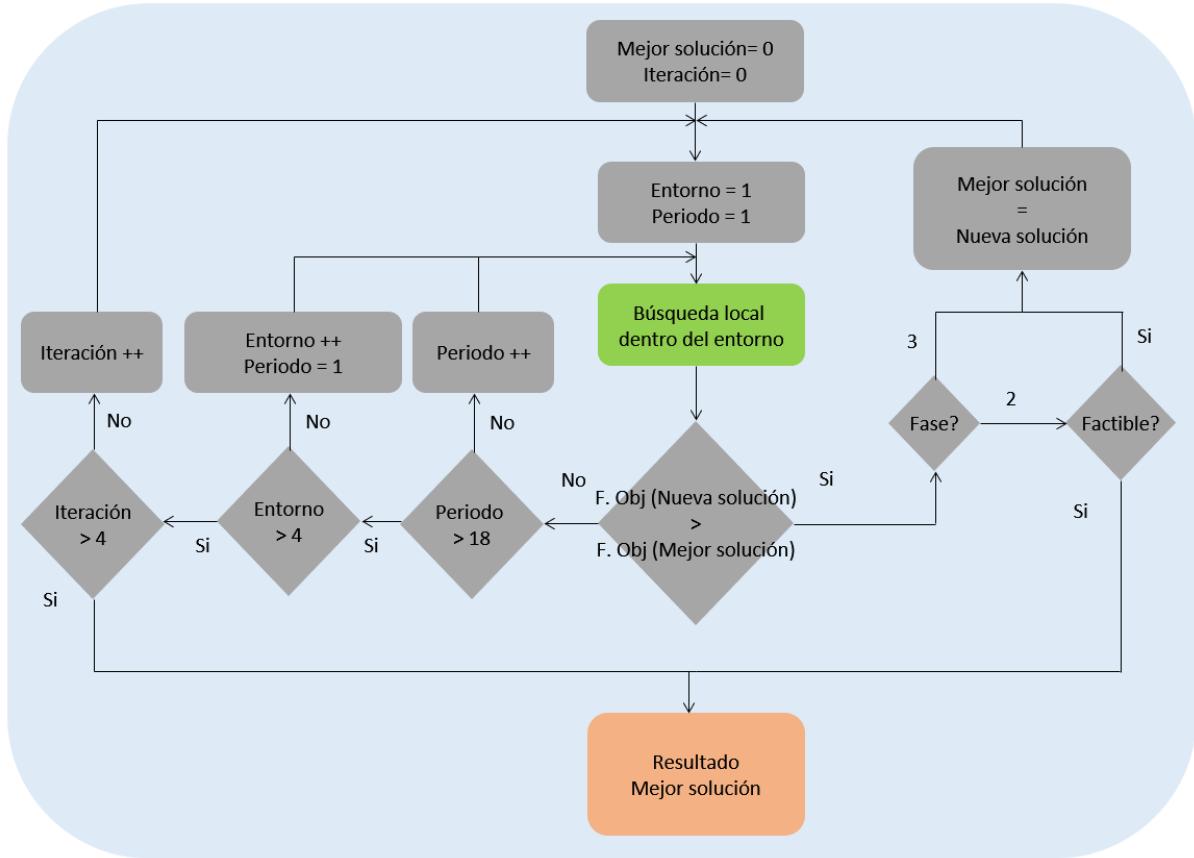


Figura 6.20: Diagrama de flujo del algoritmo VNS

### Definición del entorno de una solución

La definición de entornos de una solución para el algoritmo del VNS es uno de los puntos más importantes para garantizar su correcto funcionamiento. Pero no solo la elección de los entornos es importante, sino que la ordenación del uso de los entornos también lo es. Para la resolución del problema se han definido los siguientes cuatro entornos:

1. *Entorno 1*: Este entorno tiene como objetivo generar soluciones en las que se reduzca la carga de trabajo de las filas superiores de la matriz que representa la solución y se aumente en las filas inferiores. En la Figura 6.21 podemos observar un ejemplo de aplicación del mismo. Para que una solución esté dentro de este entorno debe cumplir las siguientes condiciones:

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

- a) El intervalo proveniente de la fila superior debe contener trabajo.
- b) El intervalo proveniente de la fila inferior debe contener descanso.
- c) El intervalo proveniente de la fila superior debe prolongar el trabajo que se encuentra en la fila inferior en el mismo sector, es decir, debe juntar dos slots o más de trabajo en el mismo sector.

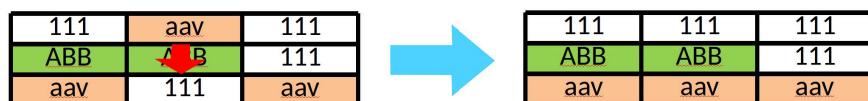


Figura 6.21: Ejemplo del *Entorno 1*

2. *Entorno 2*: Este entorno tiene como objetivo generar soluciones en las que se mueva la carga de trabajo entre filas. En la Figura 6.22 podemos observar un ejemplo de aplicación del mismo. Para que una solución esté dentro de este entorno debe cumplir las siguientes condiciones:

- a) El intervalo proveniente de la fila superior debe contener trabajo.
- b) El intervalo proveniente de la fila inferior debe contener trabajo.
- c) El intervalo proveniente de la fila superior debe prolongar el trabajo que se encuentra en la fila inferior en el mismo sector, es decir, debe juntar dos slots o más de trabajo en el mismo sector.

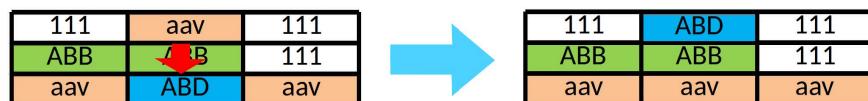
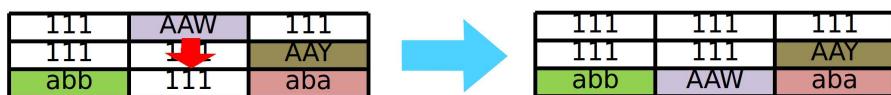


Figura 6.22: Ejemplo del *Entorno 2*

3. *Entorno 3*: Este entorno tiene como objetivo generar soluciones en las que se reduzca la carga de trabajo de las filas superiores y se aumente en las filas inferiores. Este entorno es muy similar al *Entorno 1*, pero negando la tercera condición. En la Figura 6.23 podemos observar un ejemplo de aplicación del mismo. Para que una solución esté dentro de este entorno debe cumplir las siguientes condiciones:

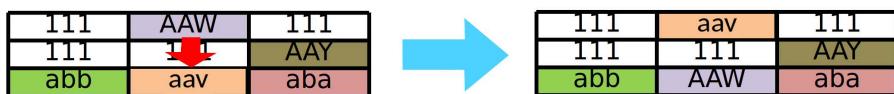
- a) El intervalo proveniente de la fila superior debe contener trabajo.

- b) El intervalo proveniente de la fila inferior debe contener descanso.
- c) El intervalo proveniente de la fila superior no debe prolongar el trabajo que se encuentra en la fila inferior en el mismo sector, es decir, no debe juntar dos slots o más de trabajo en el mismo sector.

Figura 6.23: Ejemplo del *Entorno 3*

4. *Entorno 4*: Este entorno tiene como objetivo generar soluciones en las que se mueva la carga de trabajo entre filas. Este entorno es muy similar al *Entorno 2*, pero negando la tercera condición. En la Figura 6.24 podemos observar un ejemplo de aplicación del mismo. Para que una solución esté dentro de este entorno debe cumplir las siguientes condiciones:

- a) El intervalo proveniente de la fila superior debe contener trabajo.
- b) El intervalo proveniente de la fila inferior debe contener trabajo.
- c) El intervalo proveniente de la fila superior no debe prolongar el trabajo que se encuentra en la fila inferior en el mismo sector, es decir, no debe juntar dos slots o más de trabajo en el mismo sector.

Figura 6.24: Ejemplo del *Entorno 4*

Nótese que cada uno de los entornos definidos puede utilizar un intervalo de tamaño mínimo tres y tamaño máximo doce slots, por lo que podemos considerar que el número real de entornos utilizados es de cuatro por diez (ya que por cada tamaño de intervalo podemos considerar que se tiene un entorno distinto). También hay que destacar, que la ordenación de filas por carga de trabajo solo se realiza durante la segunda fase del algoritmo de solución, mientras se reduce el número de turnos hasta que se alcance el número

### **6.3. VERSIÓN COMPLETA DEL PROBLEMA**

---

de controladores disponibles. Una vez se alcanza este punto, se detiene la ordenación, por lo que el orden de las filas carece de importancia y se deja de diferenciar entre la posición de las filas.

Por último, se debe establecer un orden de utilización de dichos entornos. Tras un análisis del comportamiento del algoritmo y de los resultados obtenidos se ha decidido lo siguiente: Para la segunda fase del método de resolución se utilizarán los entornos en el orden 2, 1, 3 y 4; mientras que el orden en la tercera fase será 2, 1, 3 y 4.

#### **Búsqueda local multicomienzo**

Como ya hemos comentando anteriormente, el algoritmo realiza un proceso de búsqueda local en cada uno de los entornos. Este proceso realiza una búsqueda iterativa en el que se analizan las soluciones en un orden determinista. El algoritmo no analiza todas las soluciones del entorno, sino que finaliza cuando no se encuentra una solución de mejora durante cierto número de iteraciones.

La búsqueda local es multicomienzo, por lo que se comienza dividiendo el entorno de la solución en  $n$  espacios de búsqueda y generando  $n$  semillas (cada semilla se genera dentro de un espacio distinto). El algoritmo de búsqueda ejecuta en paralelo múltiples búsquedas locales por cada una de las semillas.

El proceso de búsqueda parte de la semilla. El primer paso del proceso de búsqueda es seleccionar la siguiente solución disponible del espacio de búsqueda. En el caso de que la nueva solución sea la mejor encontrada, esta se sustituye por la actual. En el caso contrario, continúa la búsqueda en el espacio escogiendo la siguiente solución.

La búsqueda finaliza cuando el número de iteraciones realizadas sin encontrar ninguna solución mejor supera un umbral establecido. Este umbral se establece en relación al tamaño del entorno. Una vez finalicen las búsquedas en paralelo con las  $n$  semillas, se compara el valor objetivo de las distintas soluciones obtenidas con la mejor solución disponible hasta el momento. En el caso de que alguna de las nuevas soluciones sea mejor, esta es sustituida por la nueva solución.

## Condición de parada

La condición de parada utilizada es bastante sencilla, ya que contamos con un número de entornos reducido y la ejecución de los mismos no requiere elevados tiempos de cómputo, podemos permitirnos que el algoritmo ejecute los entornos en varias ocasiones.

Otro punto que tenemos en cuenta es que la búsqueda local no es una búsqueda exhaustiva ni determinista, por lo que pueden pasar desapercibidas ciertas soluciones más prometedoras que pueden alcanzarse en futuras ejecuciones. Por estos motivos se establece la siguiente condición de parada:

En el caso de que no se produzca ninguna mejora de la mejor solución encontrada hasta el momento en el período de cuatro iteraciones consecutivas por los cuatro entornos, la ejecución finaliza dando como solución resultante la mejor solución disponible.

### 6.3.5. Mejoras computacionales

En la primera aproximación al problema no se permite en ningún momento utilizar soluciones infactibles, y para comprobar la factibilidad de las soluciones se utilizaban expresiones regulares.

La comprobación de estas restricciones es un proceso costoso en tiempo que se debe realizar en cada iteración, por lo que realizamos un estudio computacional tanto en la segunda fase como en la tercera.

Muchas de las restricciones necesitan varias expresiones regulares para su comprobación, lo que disminuye las ventajas y reduce su eficiencia. Este estudio nos permitió observar como el tiempo de cómputo se incrementaba al utilizar expresiones regulares. Por lo cual, eliminamos el uso de las expresiones regulares en pro de la eficiencia computacional, perdiendo la modularidad y facilidades que nos ofrecen.

En la Figura 6.25 podemos ver cómo los tiempos de ejecución mejoran con la codificación de las restricciones a medida que se llevan a cabo más iteraciones.

La principal diferencia entre la segunda fase y la tercera, es la necesidad de comprobar todas las restricciones durante la segunda fase, ya que se admiten soluciones infactibles que, en función del grado de infactibilidad, tendrán mejor o peor valor objetivo. Mientras tanto, para la tercera fase no se admiten soluciones infactibles por lo que la comprobación

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

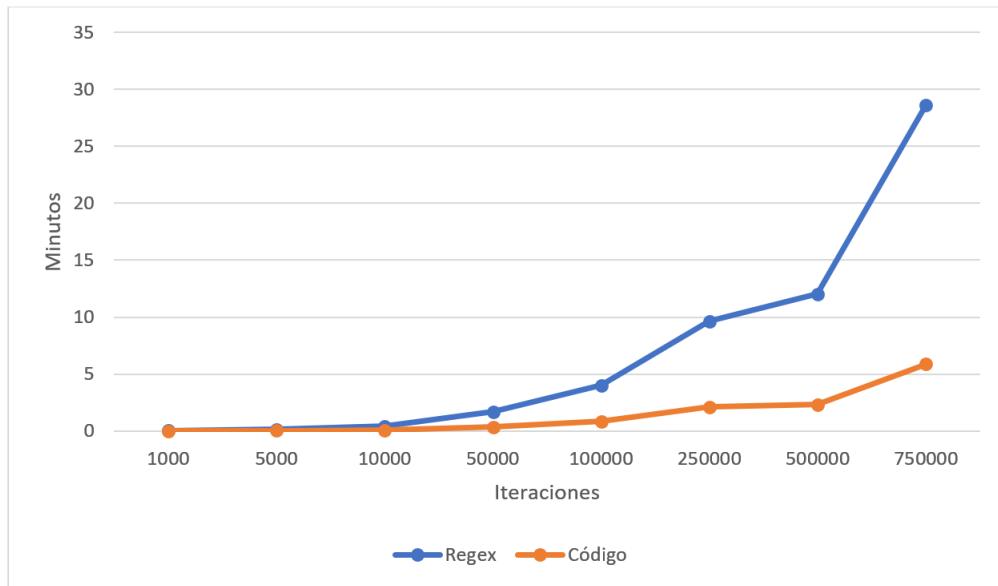


Figura 6.25: Comparativa de tiempos entre el uso de Regex y código

se detendrá al encontrar la primera restricción incumplida.

Visto lo costoso que resulta comprobar todas las restricciones, se decidió realizar otro estudio para determinar si la paralelización de este proceso podría aportar una reducción de los tiempos de ejecución significativa. La segunda fase se beneficiaría enormemente de esta mejora, mientras que para la tercera fase no es tan ventajosa.

Los resultados de este estudio se muestran en la Figura 6.26, y se aprecia una notable mejora de los tiempos de cómputo gracias a la paralelización. No obstante el grado de mejora es dependiente del número de núcleos que disponga el procesador del ordenador, en nuestro caso, las pruebas se han realizado en un ordenador con cuatro núcleos. Por este motivo hemos encontrado los mejores resultados con dos y cuatro hilos ejecutándose en paralelo. En el caso en el que aumentamos el número de hilos de ejecución hasta ocho, se puede ver como desciende significativamente el rendimiento incluso por debajo de la ejecución secuencial.

Como conclusión hay que destacar que se debe ajustar el número de hilos utilizados en función del ordenador que se utilice en la ejecución del código.

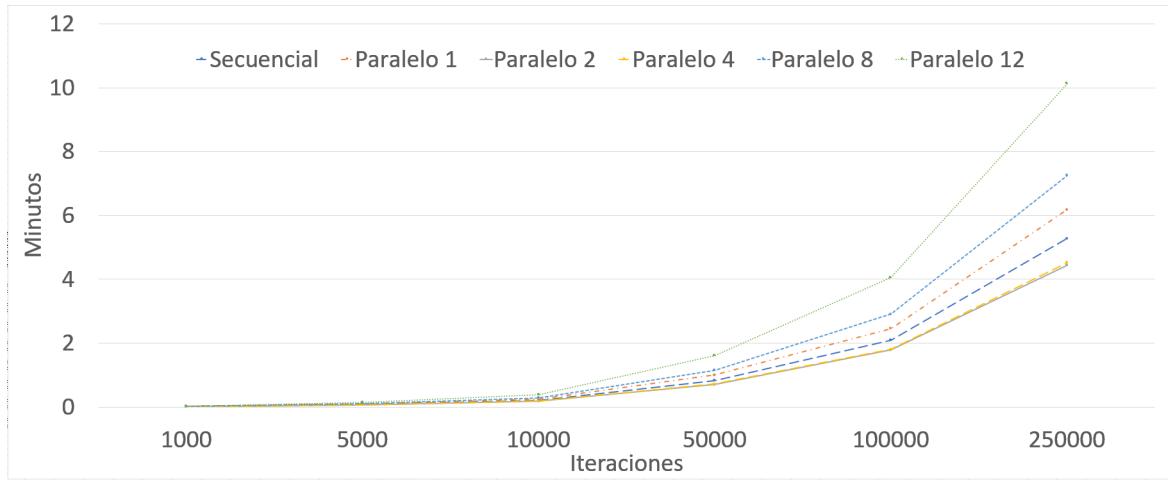


Figura 6.26: Comparativa de tiempos entre el uso de Regex y código

### 6.3.6. Ejemplo de aplicación: Recocido simulado

Una vez descrita la metodología de resolución para el segundo problema, procedemos a exponer mediante un ejemplo de la aplicación de la misma utilizando las instancias mostradas en la Tabla 6.2. Estas instancias son un conjunto de las instancias más complejas y representativas de las posibles situaciones que podemos encontrar.

La primera instancia escogida para explicar el funcionamiento de la metodología es el Caso 7. Esta instancia pertenece al turno de mañana del centro de control de Barcelona. Para la resolución del problema se dispone de 16 controladores que deben asignarse a 6 sectores abiertos. La complejidad de esta instancia reside en que únicamente disponemos de 16 controladores para 6 sectores, lo cual se sabe que es el número mínimo necesario de controladores para cumplir con todas las restricciones del problema. Durante parte del turno no se dispone más que del descanso que establecen las condiciones laborales como obligatorio. Esto junto con la apertura y cierre de sectores complica enormemente la creación de un horario de trabajo que cumpla todas las condiciones.

La sectorización del caso se encuentra expuesta en la Figura 6.27. En esta figura encontramos:



Figura 6.27: Sectorización del Caso 7

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

- 3 sectores abiertos (LECBGL, LECBLGU, LECBPPI) de 5:20 a 6:00,
- 4 sectores abiertos (LECBGL, LECBLGU, LECBGOI, LECBP2R) de 6:00 a 7:40,
- 5 sectores abiertos (LECBGOI, LECBP2R, LECBLVL, LECBLVS, LECBLVU) de 7:40 a 8:40,
- 6 sectores abiertos (LECBGOI, LECBG23, LECBLVL, LECBLVS, LECBLVU, LECBP2R) de 8:40 a 12:00,
- 5 sectores abiertos (LECBGOI, LECBP2R, LECBLVL, LECBLVS, LECBLVU) de 6:00 a 13:00.

Cada centro de control tiene sus propios núcleos y sectores por lo que se necesita la siguiente información de cada uno:

1. Relación entre todas las configuraciones y los sectores que las componen.
2. Relación entre todos los sectores y volúmenes que los componen.
3. Relación entre todos los núcleos y sectores que los componen.
4. Matriz de todos los sectores afines entre sí.

Este ejemplo pertenece al centro de control de Barcelona, y con toda esta información se procede con la primera fase de la metodología, la creación de soluciones iniciales. Durante la primera fase se crean diez soluciones iniciales con plantillas de 3x1, donde la longitud de los descansos abarca de tres slots (quince minutos) a doce slots (una hora). En este caso, las plantillas utilizan treinta controladores, ya que tenemos un total de diez sectores distintos abiertos a lo largo del turno.

A continuación, se asignan los distintos turnos a los controladores disponibles, pero algunos de los turnos permanecen sin asignar debido a que existen más turnos que controladores disponibles. Una vez realizado este paso, la primera fase queda finalizada, generando un conjunto de soluciones formado por soluciones como la mostrada en la Figura 6.28. Durante la siguiente fase, se reducirá el número de turnos y reasignarán los controladores, hasta que todos los turnos tengan un controlador asignado.

## CAPÍTULO 6

Figura 6.28: Solución inicial del Caso 7

A partir de las soluciones iniciales comienza la segunda fase, durante la cual, además de tener como objetivo reducir el número de controladores, también se eliminan todas las restricciones incumplidas. Esto se lleva a cabo mediante un algoritmo multicomienzo basado en el recocido simulado. La ejecución del algoritmo se realiza con los siguientes parámetros (los cuales han sido estudiados y ajustados previamente):

1. La temperatura inicial se calcula al inicio de la ejecución con el método propuesto por Johnson, en concreto esta instancia se resuelve con una temperatura inicial de 0.01049.
  2. La función de enfriamiento utilizada es la función de enfriamiento geométrica, con un parámetro  $\alpha$  igual a 0,95 según la referencia de Hajek.
  3. El número de iteraciones hasta el descenso de la temperatura ha sido establecido en  $L = 3000$ , utilizando el método de *cutoff* estableciendo el umbral en 0.5, es decir, se necesitan 1500 iteraciones en las que se acepte la solución encontrada para descender la temperatura antes de lo establecido.
  4. Las condiciones de parada son las siguientes, y ambas deben cumplirse simultáneamente para detener la ejecución del algoritmo:
    - a) Si la mejor solución encontrada no mejora en un 0.035 % durante 50000 iteraciones se cumple la primera condición.
    - b) Si no se genera un porcentaje de soluciones mejores a la solución actual mayor al 2 % durante 50000 iteraciones se cumple la segunda condición.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

5. Para la resolución de esta instancia el entorno utilizado ha sido el *Entorno 15*.

Una vez finaliza la ejecución del algoritmo, el cual finaliza cuando se encuentra la primera solución factible, mostrada en la Figura 6.29 (existe la posibilidad de esperar a resolver todo el conjunto de soluciones y ejecutar en la tercera fase de la metodología como un algoritmo multicomienzo).

Se lleva a cabo la ejecución de la tercera fase, en esta fase se ejecuta un algoritmo basado en *recocido simulado* para optimizar la solución factible procedente de la segunda fase. La ejecución del algoritmo se realiza con los siguientes parámetros:

1. La temperatura inicial se calcula al inicio de la ejecución con el método propuesto por Kirkpatrick, en concreto esta instancia se resuelve con una temperatura inicial de 0.00392.
2. La función de enfriamiento utilizada es la función de enfriamiento geométrica, con un parámetro  $\alpha$  igual a 0,95 según la referencia de Hajek.
3. El número de iteraciones hasta el descenso de la temperatura ha sido establecido en  $L = 3000$ , utilizando el método de *cutoff* estableciendo el umbral en 0.5, es decir, se necesitan 1500 iteraciones en las que se acepte la solución encontrada para descender la temperatura antes de lo establecido.
4. La condiciones de parada se activa cuando la mejor solución encontrada no se mejora en un 0.035 % durante 50000 iteraciones.
5. El entorno utilizado es el *Entorno 15*.

C11	111	lecbgo1	LECBG01	111	lecbgo1	111	lecbvlv	LECB023	111	lecbvlv	111	
C9	111	LECBGL	lecbgl	111	LECBGO1	lecbgo1	LECBG01	lecbvlv	111	LECBVLV	111	
C3	111	LECBGLU	lecbgl	111	LECBGO1	lecbgo1	LECBG01	lecbvlv	111	LECBVL	111	
C0	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C6	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBG01	111	
C7	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C8	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C13	LECBPP1	111	LECBGL	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111
C14	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C15	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C1	111	LECBGLU	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C2	lecbgl	111	LECBP2R	lecbgl	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111
C5	lecbgl	111	LECBP2R	LECBGLU	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111
C4	111	LECBP2R	LECBGLU	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R	111	
C12	LECBGL	lecbgl	111	LECBP2R	LECBGLU	111	LECBP2R	lecbgo1	LECBLVU	lecbvlv	111	LECBP2R

Figura 6.29: Solución factible del Caso 7

Una vez finaliza la ejecución de la tercera fase, obtenemos la solución óptima, la cual se muestra en la Figura 6.30.

CAPÍTULO 6

Figura 6.30: Solución óptima del Caso 7

En la Tabla 6.9 mostramos los resultados de la ejecución junto con el resto de casos propuestos. Como se puede apreciar, en la Tabla 6.9 encontramos los valores resultantes de la función objetivo, y el desglose en los distintos sub-objetivos que forman la función para cada uno de los casos.

Tabla 6.9: Resultados de la aplicación de la metodología de resolución sobre los Casos 7, 13, 23, 29 y 65.

	Valor objetivo global	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Tiempo de cómputo (mins)
Caso 7	0.8781	0.9445	0.7612	0.5539	0.9371	6.95
Caso 13	0.8700	0.9404	0.7493	0.5154	0.8989	7.55
Caso 23	0.8456	0.9349	0.6812	0.5198	0.9052	16.38
Caso 29	0.8662	0.9364	0.7162	0.6097	0.9136	27.09
Caso 65	0.8501	0.9302	0.6921	0.5098	0.9294	16.4

Cabe destacar que aunque los datos resulten poco intuitivos, ya que son los valores objetivos normalizados, hemos diseñado una tabla con un conjunto de datos que facilitan una evaluación de las soluciones de una forma mucho más sencilla. La Tabla 6.10 muestra más información el conjunto de casos analizados. Una vez analizados los datos de ambas tablas, los expertos de CRIDA han quedado satisfechos con la calidad de las soluciones obtenidas y con los tiempos en las que estas se obtienen.

Las tres primeras columnas de la tabla hacen referencia al primer objetivo. La primera columna contiene la información del porcentaje de intervalos de trabajo en una misma posición que se alejan del intervalo de trabajo óptimo en una misma posición por menos de 10, 15 y 25 minutos, respectivamente. La segunda columna contiene el porcentaje de intervalos de trabajo (independientemente de la posición y el sector) que se alejan del intervalo de trabajo óptimo por menos de 15, 20 y 25 minutos, respectivamente. La tercera columna muestra el número de controladores que tienen un porcentaje de trabajo

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

Tabla 6.10: Resultados explicativos de la aplicación de la metodología de resolución sobre los Casos 7, 13, 23, 29 y 65.

	Períodos de trabajo, descanso y posición			Cambios en el centro	Carga de trabajo		
	Sub-obj. 1	Sub-obj. 2	Sub-obj. 3		Nº de períodos de descansos	$\sigma$	Min
Caso 7	40,40,84 %	21,6,60,60 %	16,16,16	60	16.66	240	300
Caso 13	60.39,60.39,75.24 %	43.47,43.47,43.47 %	14,14,14	46	18.29	270	315
Caso 23	48.27,55.17,85.51 %	24.28,47.14,51.42 %	18,18,18	72	24.04	290	360
Caso 29	70,70,84.74 %	44.59,44.59,44.59 %	24,24,24	76	23.21	220	305
Caso 65	48.70,57.79,94.15 %	43.83,68.49,68.49 %	22,22,22	73	17,52	240	320

equilibrado entre sus posiciones (ejecutivo/planificador), muestra los controladores que se alejan del porcentaje menos de un 5, 10 y 15 %, respectivamente.

La cuarta columna de la Tabla 6.10 indica el número de intervalos de descanso que se encuentran en el horario, el cual representa el tercer objetivo. Mientras que la quinta es la desviación típica procedente de la carga de trabajo, la cual se utiliza para medir el equilibrio de trabajo entre los controladores del turno. Los valores mínimos y máximos de las columnas sexta y séptima indican el número de slots de trabajo mínimo y máximo de un controlador que se encuentra en ese horario.

Antes de mostrar las sectorizaciones y soluciones de todos los casos expuestos vamos a realizar una breve descripción para recordar las características de los casos analizados.

El Caso 13 pertenece a un turno de mañana del centro de control de Barcelona que abarca de 5:20 a 13:00. Para la creación del horario de trabajo disponemos de 14 controladores asignados al núcleo Ruta Este. Respecto a la dimensión del problema es similar a la instancia anterior, pero en este caso disponemos de dos controladores menos, con una carga de trabajo ligeramente menor. La sectorización de este caso se encuentra en la Figura 6.31 mientras que las soluciones inicial, factible y óptima se encuentran en las Figuras 6.32, 6.33 y 6.34, respectivamente.

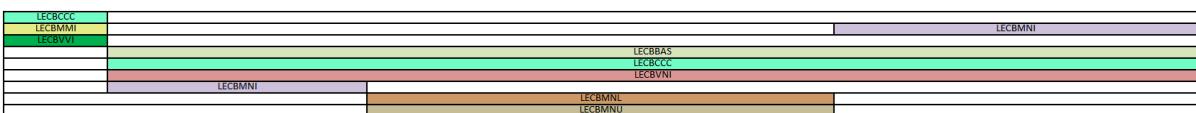


Figura 6.31: Sectorización del Caso 13

CAPÍTULO 6

Figura 6.32: Solución incial del Caso 13

C6	111	LECBVNI	111	LECBVNI	111	lecbmn	lecbmn	111	LECBAS	111	LECBMN	111	LECBMNI	111
C6	111	lecbbas	LECBAS	111	lecbmn	LECBMNU	111	lecbmn	111	lecbas	LECBCCC	111	lecbmn	LECBCCC
C0	111	LECBMNI	111	lecbmn	lecbmn	111	lecbbas	lecbccc	LECBCCC	111	lecbmn	111	lecbmn	LECBCCC
C1	111	LECBMN	111	LECBMN	111	LECBMN	LECBMN	111	LECBMN	111	LECBMN	111	LECBMN	111
C8	lecbccc	111	lecbccc	111	LECBAS	lecbvn	LECBVNI	111	lecbvn	LECBVN	111	lecbvn	LECBVN	111
C7	lecbbas	111	lecbccc	111	lecbmn	lecbmn	LECBCCC	111	lecbvn	LECBVN	111	lecbvn	LECBVN	111
C5	111	LECBCCC	lecbmn	111	lecbmn	lecbas	LECBAS	111	LECBMN	LECBAS	111	lecbmn	LECBAS	111
C9	lecbmni	LECBAS	111	lecbmn	lecbas	LECBAS	111	LECBMN	lecbmn	LECBMN	111	lecbmn	LECBMN	111
C13	111	lecbvn	LECBVN	111	LECBMN	lecbmn	111	lecbmn	111	lecbmn	111	lecbccc	lecbmn	111
C12	111	LECBCCC	lecbvn	111	LECBVN	lecbvn	LECBVN	111	LECBVN	LECBVN	111	lecbmn	LECBVN	111
C10	LECBCCC	111	lecbmn	111	LECBMN	lecbmn	111	LECBMN	111	lecbmn	LECBCCC	111	lecbmn	LECBCCC
C10	LECBMNI	lecbmni	111	LECBMN	111	LECBMN	lecbmn	111	LECBMN	lecbmn	LECBMNI	111	lecbas	LECBMNI
C3	111	lecbccc	111	LECBCCC	lecbccc	111	LECBMN	lecbas	LECBMN	111	LECBMN	111	lecbmn	LECBMN
C4	LECBVN	111	LECBVN	111	LECBCCC	lecbccc	111	LECBVN	lecbccc	LECBCCC	111	lecbmn	LECBCCC	111

Figura 6.33: Solución factible del Caso 13

C1	111	LECBMNI	lecbcc	LECBMNI	111	lecbmnl	lecbmnu	LECBMNU	lecbmnu	111	lecbmnu	111	lecbas	LECBAS
C8	111	lecbmni	lecbvn1	LECBVNI	lecbbas	111	lecbmnl	LECBMNL	lecbbas	111	lecbmni	111	lecbbas	LECBBAS
C2	111	LECBVNI	lecbbas	111	lecbmnl	lecbcc	LECBCC	lecbcc	LECBCC	111	lecbcc	111	lecbcc	LECBCCC
C4	111	LECBCCC	lecbcc	111	lecbmnl	lecbccc	LECBCCC	lecbccc	LECBCCC	111	lecbccc	111	lecbbas	LECBBCC
C7	111	LECBCCC	lecbccc	111	lecbmnl	lecbccc	LECBCCC	lecbccc	LECBCCC	111	lecbccc	111	lecbmni	LECBMNI
C3	111	LECBBCC	lecbccc	111	lecbmnl	lecbccc	LECBBCC	lecbccc	LECBBCC	111	lecbccc	111	lecbvni	LECBVNI
C5	111	LECBBCC	lecbmni	111	LECBMNU	lecbbas	LECBBAS	lecbbas	LECBBAS	111	lecbbas	111	LECBVNI	LECBVNI
C8	111	LECBMNI	lecbvn1	LECBAS	111	lecbmnu	LECBMNU	LECBMNU	lecbmnu	111	lecbvni	111	LECBVAS	LECBVAS
C9	111	LECBVNI	lecbbas	111	lecbmnu	LECBMNU	LECBMNU	LECBMNU	lecbmnu	111	lecbvni	111	LECBVAS	LECBVAS
C12	111	LECBMNI	lecbvn1	LECBVNI	111	lecbbas	LECBBAS	LECBBAS	LECBBAS	111	LECBMNI	111	LECBBAS	LECBBAS
C11	111	LECBVNI	lecbvn1	LECBVNI	111	lecbvni	LECBVNI	LECBVNI	lecbvni	111	LECBVNI	111	lecbvni	LECBVNI
C10	111	LECBCCC	lecbccc	111	LECBMNL	lecbvni	LECBMNL	LECBMNL	lecbvni	111	LECBMNL	111	lecbvni	LECBVNI

Figura 6.34: Solución óptima del Caso 13

El Caso 23 pertenece a un turno largo de mañana del centro de control de Barcelona que abarca de 4:40 a 13:00. Para la creación del horario de trabajo disponemos únicamente de 18 controladores y debemos cubrir un período de aumento de tráfico aéreo en el que se encuentran abiertos 7 sectores. En el caso que se mantuvieran los 7 sectores abiertos durante un período de tiempo más largo, el problema no tendría una solución factible, ya que se necesitarían 19 controladores como mínimo. Durante 5 horas tenemos una sectorización con 7 sectores abiertos. Este es el principal problema al que nos enfrentamos a la hora de resolver el problema. La sectorización de este caso se encuentra en la Figura 6.35 mientras que las soluciones inicial, factible y óptima se encuentran en las Figuras 6.36, 6.37 y 6.38, respectivamente.



Figura 6.35: Sectorización del Caso 23

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

Figura 6.36: Solución inicial del Caso 23

Figura 6.37: Solución factible del Caso 23

C5		III	lecbgo1	LECBG01	III	LECBP2R	lecbgo1	lecbgo1	III	lecbvi	LECBG01	III	lebg01			
C7		III	LECBG01	lecbgo1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III	lecbv1	lecbgo2			
C10		III	lecbgo1	LECBV1	III	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C12		III	LECBG01	LECBV1	III	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C13		III	lecbgo1	LECBV1	III	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C4		III	LECBG01	lecbv1	III	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C16		III	LECBP2R	lecbv1	III	lecbgo2	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C15		III	lecbgo1	LECBP2R	III	lecbgo2	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C14		III	lecbv1	LECBP2R	III	lecbgo2	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C11		III	LECBG01	lecbv1	III	lecbgo1	LECBG01	III	LECBV1	lecbv1	III	lecbv1	lecbv1	III		
C6		III	LECBG01	lecbv1	III	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III			
C17		III	LECBG01	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III		
C3		III	LECBG01	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III		
C.6		III	LECBP2R	lecbv1	III	lecbgo1	LECBG01	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III		
C.6		III	LECBP2R	lecbv1	III	lecbgo1	LECBG01	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III		
C.5		III	lecbkw	LECBG01	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III	LECBV1	lecbv1	III	
C.11		III	ECBBW1	lecbg1	III	lecbv1	III	LECBG01	lecbg1	III	LECBG01	lecbg1	III	LECBV1	lecbv1	III

Figura 6.38: Solución óptima del Caso 23

El Caso 29 pertenece a un turno de tarde del centro de control de Barcelona que abarca de 14:00 a 21:20. Para la creación del horario de trabajo se dispone de 14 controladores para el núcleo Este y otros 14 controladores para el núcleo Oeste. La resolución de este problema es más complicada por disponer de dos núcleos de trabajo, primero porque doblamos las dimensiones del problema, y el segundo motivo es que añadimos una restricción la cual no aplica en las instancias con un único núcleo. La sectorización de este caso se encuentra en la Figura 6.39 mientras que las soluciones inicial, factible y óptima se encuentran en las Figuras 6.40, 6.41 y 6.42, respectivamente.

CAPÍTULO 6



Figura 6.39: Sectorización del Caso 29

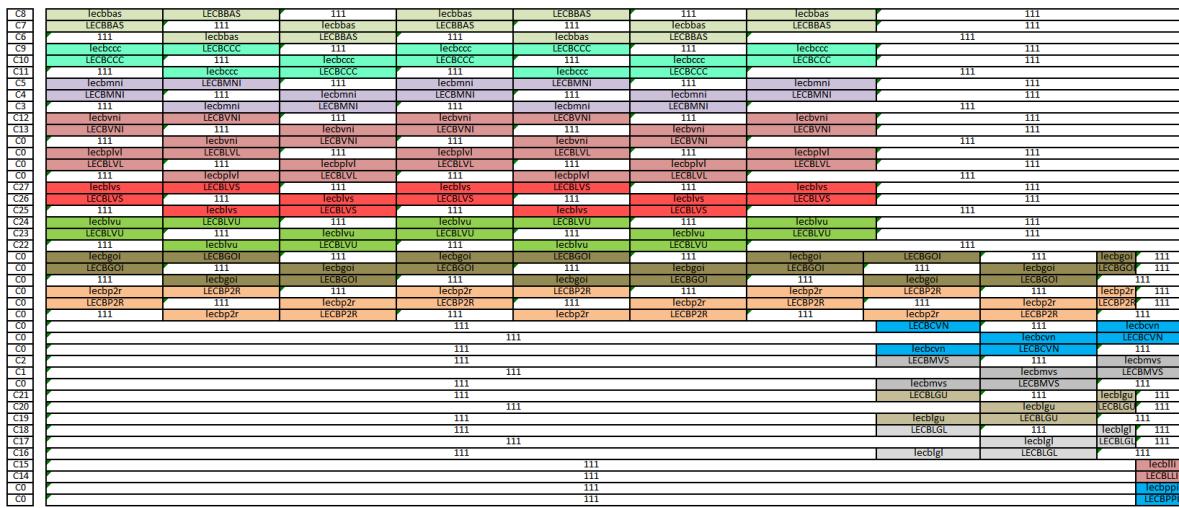


Figura 6.40: Solución inicial del Caso 29

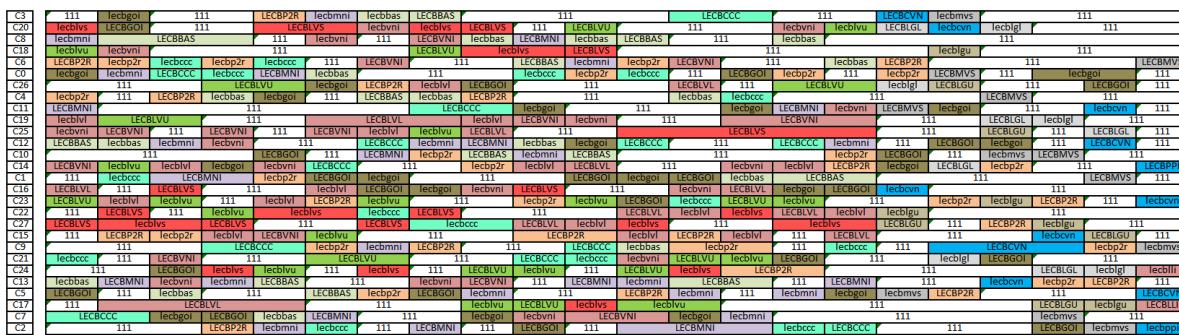


Figura 6.41: Solución factible del Caso 29



Figura 6.42: Solución óptima del Caso 29

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

El Caso 65 pertenece a un turno de tarde del centro de control de Madrid que abarca de 14:00 a 21:20. Para la creación del horario de trabajo disponemos de 22 controladores. El primer reto se encuentra en las dimensiones del problema, debemos cubrir una sectorización con 7 y 8 sectores abiertos. Pero además, se produce un cierre y apertura de sectores realizado en varias ocasiones. Esto provoca cambios en sala constantemente y dificulta cumplir las condiciones laborales. La sectorización de este caso se encuentra en la Figura 6.43 mientras que las soluciones inicial, factible y óptima se encuentran en las Figuras 6.44, 6.45 y 6.46, respectivamente.

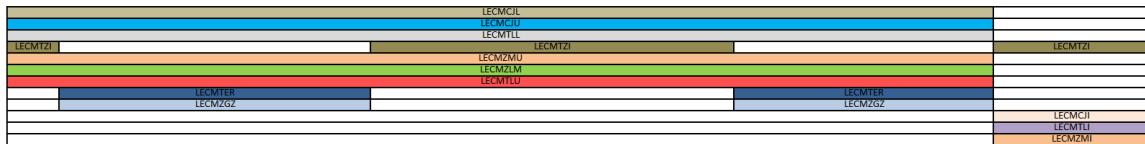


Figura 6.43: Sectorización del Caso 65

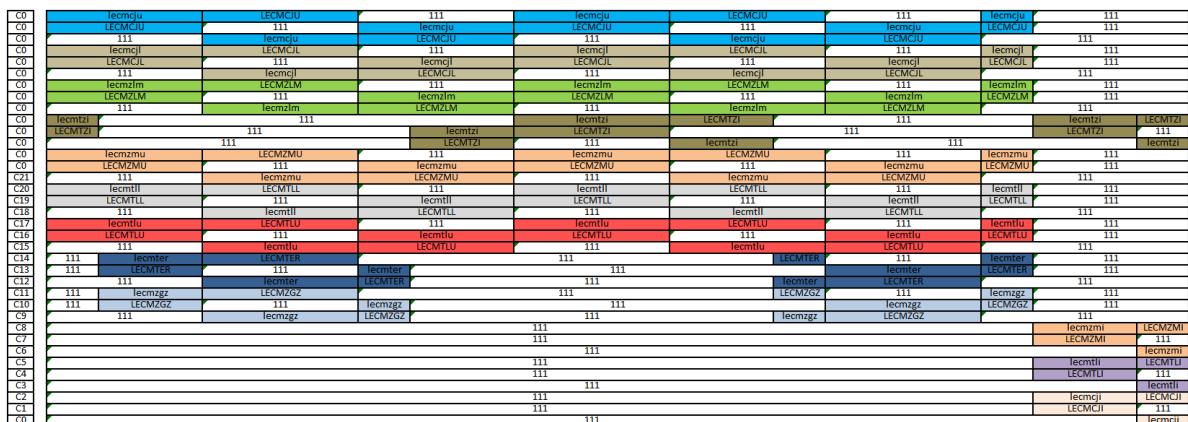


Figura 6.44: Solución inicial del Caso 65



Figura 6.45: Solución factible del Caso 65

C11	111	LECMTER	lecmtu	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C5	111	LECMJU	lecmter	111	LECMIL	lecmil	111	LECMZL	lecmzlu	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C1	111	LECMTER	lecmtu	111	LECMJU	lecmter	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C9	111	LECMTER	lecmtu	111	LECMJU	lecmter	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C18	lecmilu	111	LECMTER	lecmtu	111	LECMJU	lecmter	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111
C4	111	LECMTER	lecmtu	111	LECMJU	lecmter	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C6	111	LECMTER	lecmtu	111	LECMJU	lecmter	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C2	111	LECMZG	lecmil	111	LECMTER	lecmtu	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	
C14	LECMZL	111	LECMZG	lecmil	111	LECMIL	lecmil	111	LECMILU	lecmil	111	LECMIL	lecmil	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111	LECMZL	lecmzlu	111
C3	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111	
C8	lecmilm	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111
C0	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111	
C12	LECMIL	111	LECMZL	LECMZG	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111
C21	LECMIL	111	LECMZL	LECMZG	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMILU	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111	LECMIL	LECMZL	111
C5	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	
C17	LECMJU	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111
C16	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	
C7	LECMIL	111	LECMZM	LECMIL	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111
C13	lecmil	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111
C19	LECMZM	111	LECMZM	LECMIL	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111
C20	LECMIL	111	LECMZM	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111
C15	LECMILU	111	LECMZM	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMILU	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111	LECMIL	LECMZM	111

Figura 6.46: Solución óptima del Caso 65

### 6.3.7. Comparativa de los resultados de los algoritmos VNS y SA

En esta sección vamos a realizar una comparativa de los dos algoritmos utilizados para la resolución del problema en base al análisis de las soluciones obtenidas.

Vamos a utilizar las instancias expuestas y resueltas por el recocido simulado en la Sección 6.3.6. Para realizar la comparación nos centramos en la tercera fase del método de resolución y mostramos los resultados del VNS aplicado a las instancias previamente utilizadas. Estos resultados se encuentran en la Tabla 6.11, que nos muestra los cuatro objetivos que conforman la función objetivo utilizada en la última fase del método de resolución. Estos objetivos han sido previamente expuestos en la Sección 6.3.3, aun así vamos a realizar un breve recordatorio:

- Objetivo 1: Se debe cumplir un conjunto de tres de sub-objetivos deseables asociados a las condiciones laborales de los controladores. El primero es que el tiempo óptimo de trabajo en posición son 45 minutos, el segundo es que el tiempo óptimo de trabajo entre descansos son 90 minutos, y el tercero es que el porcentaje de tiempo que un controlador trabaja en posiciones ejecutivas debe estar comprendido entre el 40 % y el 60 % del trabajo total realizado.
- Objetivo 2: Se debe procurar mantener una estructura/apariencia similar a la que se encuentra en los estadios actualmente utilizados por los controladores.
- Objetivo 3: Este objetivo está compuesto por dos sub-objetivos. El primer sub-objetivo consiste en minimizar el número de intervalos de descanso, mientras que el segundo sub-objetivo consiste en maximizar el número de sectores elementales que cubren los controladores sin llegar a superar un umbral.

### 6.3. VERSIÓN COMPLETA DEL PROBLEMA

---

4. Objetivo 4: Se debe realizar una distribución de la carga de trabajo equilibrada entre todos los controladores.

Tabla 6.11: Comparativa de los resultados obtenidos aplicando los algoritmos VNS y SA.

		Valor objetivo global	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Tiempo de cómputo (mins)
VNS	Caso 7	0.8643	0.9421	<b>0.7631</b>	<b>0.5674</b>	0.9011	<b>4.25</b>
	Caso 13	0.8613	0.9397	<b>0.7526</b>	0.5023	0.8624	<b>6.83</b>
	Caso 23	0.8252	0.9283	0.6573	<b>0.5513</b>	<b>0.9181</b>	<b>11.93</b>
	Caso 29	0.8379	0.9333	0.6987	0.5678	<b>0.9187</b>	31.84
	Caso 65	0.8382	0.9277	0.6911	0.4846	<b>0.9321</b>	<b>15.39</b>
SA	Caso 7	<b>0.8781</b>	<b>0.9445</b>	0.7612	0.5539	<b>0.9371</b>	6.95
	Caso 13	<b>0.8700</b>	<b>0.9404</b>	0.7493	<b>0.5154</b>	<b>0.8989</b>	7.55
	Caso 23	<b>0.8456</b>	<b>0.9349</b>	<b>0.6812</b>	0.5198	0.9052	16.38
	Caso 29	<b>0.8662</b>	<b>0.9364</b>	<b>0.7162</b>	<b>0.6097</b>	0.9136	<b>27.09</b>
	Caso 65	<b>0.8501</b>	<b>0.9302</b>	<b>0.6921</b>	<b>0.5098</b>	0.9294	16.4

De acuerdo con los resultados expuestos en la Tabla 6.11 podemos obtener una serie de conclusiones tras analizar los datos. La primera es la obtención de mejores resultados (marcados en negrita) por parte del algoritmo SA frente al VNS, mientras que el algoritmo VNS obtiene resultados ligeramente inferiores en un tiempo de cómputo menor, excepto para el Caso 29.

El SA consigue una mejor optimización para el primer objetivo, y debido a que este tiene un mayor peso en la función objetivo obtiene un mejor resultado en conjunto. El VNS no es capaz de alcanzar este grado de optimización para el primer objetivo, pero lo compensa ligeramente superando los resultados obtenidos por el SA para algunos de los casos y en el resto de objetivos con un tiempo de cómputo menor.

Debido a que los datos de la función objetivo normalizada pueden resultar poco intuitivos, en la Tabla 6.12 se muestra un desglose de los objetivos con datos tangibles y que aportan una información legible, al igual que se ha realizado en la Sección 6.3.6.

Las tres primeras columnas de la tabla hacen referencia al primer objetivo. La primera columna contiene la información del porcentaje de intervalos de trabajo en una misma posición que se alejan del intervalo de trabajo óptimo en una misma posición por menos de 10, 15 y 25 minutos, respectivamente.

La segunda columna contiene el porcentaje de intervalos de trabajo (independientemente de la posición y el sector) que se alejan del intervalo de trabajo óptimo por menos de 15, 20 y 25 minutos, respectivamente.

La tercera columna muestra el número de controladores que tienen un porcentaje de trabajo equilibrado entre sus posiciones (ejecutivo/planificador). En concreto, muestra los controladores que se alejan del porcentaje menos de un 5, 10 y 15 %, respectivamente.

La cuarta columna de la tabla indica el número de intervalos de descanso que se encuentran en el horario, el cual representa el tercer objetivo. Finalmente, la quinta columna contiene la desviación típica de la carga de trabajo, la cual se utiliza para medir el equilibrio de trabajo entre los controladores del turno. Los valores mínimos y máximos, de las columnas sexta y séptima, indican el número de slots de trabajo mínimo y máximo de un controlador que se encuentra en ese horario.

Tabla 6.12: Comparación de los resultados explicativos

	Períodos de trabajo, descanso y posición			Cambios en el centro Nº de períodos de descansos	Carga de trabajo		
	Sub-obj. 1	Sub-obj. 2	Sub-obj. 3		$\sigma$	Min	Max
VNS	Caso 7 <b>42.32,42.32,60 %</b>	18.36,56.45,56.45 %	16,16,16	<b>59</b>	18.74	220	300
	Caso 13 58.23,58.23,73.7 %	<b>44.12,44.12,44.12 %</b>	14,14,14	48	20.65	260	315
	Caso 23 45.74,45.74,80.51 %	<b>28.93,49.42,57.2 %</b>	18,18,18	<b>67</b>	<b>23.45</b>	290	360
	Caso 29 66.39,66.39, <b>87.32 %</b>	42.85,42.85,42.85 %	24,24,24	79	<b>21.21</b>	220	<b>300</b>
	Caso 65 <b>50.67,50.67,85.79 %</b>	<b>47.37,64.87,64.87 %</b>	22,22,22	75	<b>15,78</b>	240	<b>310</b>
SA	Caso 7 40,40,84 %	<b>21.6,60,60 %</b>	16,16,16	60	<b>16.66</b>	<b>240</b>	300
	Caso 13 <b>60.39,60.39,75.24 %</b>	43.47,43.47,43.47 %	14,14,14	<b>46</b>	<b>18.29</b>	<b>270</b>	315
	Caso 23 <b>48.27,55.17,85.51 %</b>	24.28,47.14,51.42 %	18,18,18	72	24.04	290	360
	Caso 29 70,70,84.74 %	<b>44.59,44.59,44.59 %</b>	24,24,24	<b>76</b>	23.21	220	305
	Caso 65 48.70,57.79, <b>94.15 %</b>	43.83, <b>68.49,68.49 %</b>	22,22,22	<b>73</b>	17,52	240	320

En la Tabla 6.12 se observa que los resultados de ambas metaheurísticas no distan en exceso, existe una pequeña predominancia por parte del algoritmo SA.

Para el primer objetivo, el cual está compuesto por tres sub-objetivos, el recocido simulado obtiene mejores resultados en los dos primeros, y en el tercer objetivo tienen los mismos resultados. El reparto de pesos realizado por los expertos otorga a este objetivo un porcentaje significativamente mayor que al resto. Esto se observa reflejado en el valor objetivo.

En el objetivo tres (cambios en el centro) observamos resultados similares para ambos algoritmos, obteniendo en tres de los cinco casos utilizados como ejemplo mejores resul-

## 6.4. PROBLEMA TÁCTICO

tados el SA. En el objetivo cuatro, observamos que el VNS obtiene mejores resultados (aunque similares) en tres de los cinco casos.

Podemos concluir que el algoritmo basado en VNS, pese a otorgar resultados con una mayor velocidad, tienen una calidad ligeramente inferior a los resultados proporcionados por el recocido simulado.

### 6.4. Problema táctico

El problema consiste en la modificación de turnos de trabajo cuando estos se están ejecutando debido a que ha surgido un imprevisto, al cual debe proporcionarse una solución en el menor tiempo posible. La principal diferencia con respecto a los dos problemas anteriores se encuentra en que, este problema, es resuelto durante la fase táctica. En cambio, los otros dos problemas son resueltos durante la fase pre-táctica. Esto quiere decir que el tiempo disponible es reducido. Además, estamos trabajando sobre un horario que se estaba aplicando en ese mismo turno, mientras que en los otros dos problemas creamos un horario sin restricciones previas.

Para la resolución del problema actual, el método utilizado se compone de dos fases: Una primera fase en la que se utiliza una heurística para la modificación de los turnos de trabajo planificados y que se están aplicando en el momento que surge el imprevisto.

Como ya hemos mencionado, el tiempo se discretiza en slots, por lo que denominaremos este instante como *momento de cambio* o *slot de cambio*. Esto significa que el contenido de todos los slots anteriores al slot de cambio no pueden ser modificados. Ejemplificamos el concepto de slot de cambio en la Figura 6.47 con una linea roja vertical.

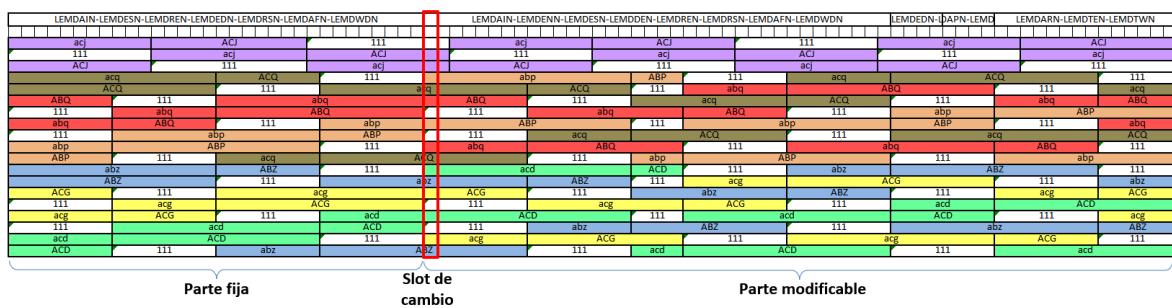


Figura 6.47: Ejemplo de slot de cambio

En función del imprevisto ocurrido, deberemos modificar la sectorización, la matriz de

turnos y la lista de controladores. Por ejemplo, la baja de un controlador (por enfermedad o cualquier urgencia) no siempre supone el cierre de sectores. En ocasiones, el resto de los controladores pueden cubrir su puesto de control hasta que pueda cubrirse la suplencia por un nuevo controlador.

Otro ejemplo es el cierre de un aeropuerto por las condiciones climáticas, en cuyo caso, se desviará el tráfico a los aeropuertos cercanos, lo que supondrá un aumento del tráfico aéreo y, por tanto, se deberán abrir nuevos sectores. Como consecuencia, se produciría un cambio en la sectorización y en la matriz de turnos y, posiblemente, deban incorporarse controladores nuevos para ocupar los nuevos sectores.

Los cambios necesarios en la sectorización serán establecidos por los responsables en sala, y deben ser introducidos al algoritmo de solución junto con los turnos de trabajo que se están utilizando.

Cabe destacar que las soluciones iniciales generadas en esta primera fase, tienden a incumplir algunas de las condiciones laborales y utilizar más controladores de los que se encuentran disponibles.

En la segunda fase, utilizamos un algoritmo basado en *recocido simulado* (SA). Como entrada del algoritmo se utiliza la solución que proviene de la primera fase para la generación de soluciones factibles. En esta fase, aplicamos una función objetivo que penaliza lo siguiente: el incumplimiento de las restricciones, la utilización de más controladores que los realmente disponibles para la resolución del problema, la generación de demasiados cambios de posición en la sala en el momento del imprevisto y la generación de soluciones desestructuradas.

#### 6.4.1. Fase 1: Modificación de la solución actual

El algoritmo de inicialización tiene como objetivo transformar la solución generada antes del imprevisto y aplicada durante parte del turno de trabajo, en una solución que contenga los cambios propuestos para solucionar el imprevisto surgido. Para esto, dividimos el algoritmo en cuatro pasos.

1. *Cierre de sectores*: En numerosas ocasiones un sector se cierra y al mismo tiempo se abren dos sectores afines al cerrado. El objetivo de este paso es detectar los sectores

#### 6.4. PROBLEMA TÁCTICO

---

cerrados y sus sectores afines con el fin de sustituir los slots de trabajo del sector cerrado por slots de trabajo de los nuevos sectores afines.

El algoritmo busca todos los sectores cerrados y comprueba si existen sectores afines para cada uno de ellos. En el caso de que no se encuentren nuevos sectores abiertos afines al sector cerrado, se sustituyen los slots de trabajo de ese sector por descansos en la matriz de turnos. En caso contrario, es decir, se encuentra un nuevo sector por abrir y este es afín al sector cerrado, se sustituyen los slots de trabajo del sector cerrado por slots de trabajo del sector abierto y afín.

2. *Apertura de sectores:* Cuando los nuevos sectores no son afines con ninguno de los sectores cerrados, o los sectores cerrados ya han sido asociados con otros sectores afines, se lleva a cabo este paso.

El algoritmo añade los slots de trabajo de los sectores abiertos en plantillas de trabajo, y añade estas plantillas a la matriz de turnos. Existen varias plantillas que pueden utilizarse, pero se ha optado por aplicar la plantilla mostrada en la Figura 2. El motivo de esta elección es la facilidad con la que se pueden realizar cambios en esta plantilla sin incumplir demasiadas restricciones debido a su estructura. Dicha plantilla utiliza tres controlares para cubrir un sector completo.

En este paso, se añaden más turnos de trabajo, por lo que es posible que existan más turnos de trabajo que controladores disponibles. No obstante, introducir los slots de trabajo entre los descansos de los controladores generaría matrices de turnos con un alto número de restricciones incumplidas y desestructuradas, lo cual dificultaría enormemente el trabajo del algoritmo en la segunda fase.

3. *Baja de controladores:* Otra posible causa es la baja de un controlador durante su turno de trabajo. En este paso, se modifica el turno de trabajo que tiene asociado el controlador que se encuentra de baja y en la matriz de turnos se indica que este controlador no se encuentra disponible para asumir ningún trabajo desde el momento en que se inicia la baja. Si el trabajo que tuviera asociado puede ser asumido e introducido al turno de otro controlador, se procederá a añadirlo y, en caso contrario, se generará un nuevo turno de trabajo y será añadido.

4. *Alta de controladores*: Un controlador se puede ausentar y es por esta razón que existen controladores de guardia para solucionar este tipo de situaciones. Los controladores de guardia tienen un plazo para presentarse en el centro de control desde que se produce el aviso. Por tanto, podemos contar con nuevos controladores a la hora de repartir la carga de trabajo.

En este paso, añadimos los nuevos controladores a la lista de controladores y añadimos un nuevo turno para cada uno, el cual admite trabajo desde el momento en que el controlador llega al centro de control. Por último, se asocia este turno al nuevo controlador.

Una vez se realizan los cuatro pasos del algoritmo de inicialización, obtenemos una solución inicial para comenzar con la segunda fase. Para ello, la solución debe cumplir con las siguientes premisas:

- Todos los sectores deben estar cubiertos en ambas posiciones de trabajo en todo momento.
- Todos los controladores deben tener un turno asociado en la lista de turnos, aunque pueden existir turnos de trabajo sin controladores asignados (designados como *controladores imaginarios*).
- Todos los controladores deben figurar en la lista de controladores.

#### 6.4.2. Fase 2: SA para obtención de soluciones factibles y óptimas

En esta fase contamos con cuatro objetivos, de los cuales el primero es de obligado cumplimiento.

1. Primer objetivo: La solución debe utilizar mismo número de turnos que de controladores disponibles.
2. Segundo objetivo: La solución debe ser factible. Se deben cumplir todas las restricciones laborales de los controladores y, además, se deben cubrir todos los sectores abiertos en ambas posiciones en todo momento.

## 6.4. PROBLEMA TÁCTICO

---

3. Tercero objetivo: Minimizar el número de cambios realizados en la estructura de la solución frente a la solución previa. Los controladores conocen su turno de antemano, por lo que un elevado número de modificaciones en mitad del turno puede provocar confusión, errores y posibles situaciones de peligro, de ahí la importancia de este objetivo.
4. Cuarto objetivo: Minimizar el número de cambios durante el slot de cambio, es decir, procurar favorecer que los controladores se mantengan en la misma posición en el momento del cambio. En ese momento, es posible que varios sectores se abran o cierren y algunos controladores cambien su puesto de trabajo de forma obligada, por lo que trataremos de minimizar el número de cambios para evitar producir una situación en la que todos los controladores se levanten de su puesto simultáneamente.

Para esta fase se utiliza un algoritmo basado en el *recocido simulado uniobjetivo*, el cual se encuentra explicado en la Sección 5.2.1. En esta sección explicaremos las decisiones de diseño tomadas en relación con el algoritmo.

### Función objetivo

Para cumplir con estos objetivos utilizaremos una función objetivo que estará compuesta por la suma de cuatro funciones ponderadas con distintos pesos.

$$f_{total} = f_1 \times \mu_1 + f_2 \times \mu_2 + f_3 \times \mu_3 + f_4 \times \mu_4. \quad (6.31)$$

1. **Primer término:** Este término de la función se encuentra explicado en la Sección 6.3.2. La reducción del número de controladores se consigue maximizando la Expresión 6.3. No obstante, la diferencia se encuentra a la hora de ordenar los turnos. En este caso la matriz se divide en dos secciones: los turnos que tienen un controlador asignado y los turnos que no tienen controlador asignado. De esta forma, los turnos sin controlador asignado ocuparán las primeras filas de la matriz ordenados por carga de trabajo, mientras que los turnos con un controlador asignado se ordenarán entre ellos posteriormente. El objetivo no es otro que favorecer que los turnos que van a ser eliminados sean los turnos sin controladores asignados.

2. **Segundo término:** Este término de la función también se encuentra explicado en la Sección 6.3.2. Para analizar la reducción de la infactibilidad contabilizamos el número de condiciones de entorno totales que se están incumpliendo, cuyo número hay que minimizar, y que denotamos por  $R_i$ . Para transformarlo en forma de maximización y normalizarlo, usamos la Expresión 6.8.
3. **Tercer término:** La función tiene como objetivo reducir el número de cambios inmediatamente después del *slot de cambio*. El hecho de que algunos de los controladores modifiquen su posición en este momento es inevitable. Sin embargo, otros pueden mantener la misma posición. Para evitar un cambio simultáneo de un gran número de controladores, utilizaremos la siguiente función:

$$f_3 = \frac{\sum_{k=1}^{nATC} \begin{cases} 1, & \text{si } slot_{sk} = slot_{(s-1)k} \\ 0, & \text{si } slot_{sk} \neq slot_{(s-1)k} \end{cases}}{nATC}, \quad (6.32)$$

donde  $slot_{sk}$  es el contenido del slot  $s$  en el slot de cambio y en el turno  $k$ , mientras que  $slot_{(s-1)k}$  es el contenido del slot anterior al cambio y en el turno  $k$ .

4. **Cuarto término:** El objetivo de este término es mantener una estructura y/o apariencia similar a la que se encuentra en los estadillos. A falta de una medida de similitud proporcionada por un experto, se entiende como estructura similar a un estadillo las soluciones que tengan los sectores más agrupados cuando estos sean iguales, así como los descansos. El término se encuentra explicado en la Sección 6.3.3 y utiliza la Expresión 6.20.

### Temperatura inicial, función de enfriamiento y número de iteraciones hasta el descenso de la temperatura

Para el ajuste de estos tres parámetros se ha realizado el mismo proceso que se realizó para el segundo problema resuelto.

En esta ocasión, para la realización de este estudio se ha tomado una muestra representativa de las instancias del problema, las cuales se encuentran descritas en la Tabla

## 6.4. PROBLEMA TÁCTICO

---

Tabla 6.13: Instancias más representativas del problema.

	Centro de control	Núcleos	Turno	1º Núcleo				2º Núcleo			
				Evolución de la sectorización		Nº controladores	Evolución de la sectorización		Nº controladores		
Caso 1	Barcelona	Ruta W	MC	3		7			5 → 6		17
Caso 4	Madrid	Ruta 2	TC	7 → 5 → 4 → 3		19					
Caso 5	Madrid	Ruta 2	TC	7 → 5 → 4 → 3		19					
Caso 8	Canarias	Ruta	MC	8		22					
Caso 9	Canarias	Ruta	MC	8		22					

6.13. Los resultados del cálculo de la temperatura inicial se muestran en la Tabla 6.14, mientras que los resultados del estudio sobre la función de enfriamiento y el número de iteraciones son mostrados en la Tabla 6.15.

Tabla 6.14: Resultados del ajuste de la temperatura inicial para el problema táctico

	Parámetros	Tiempo de cómputo (min)					Temperatura escogida				
		C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9
Temp. inicial	Ben-Ameur	0.223	<b>0.008</b>	0.009	<b>0.007</b>	<b>0.007</b>	0.1005	0.03871	0.0527	0.09882	0.09907
	Johnson	0.026	0.018	0.022	0.02	0.021	0.01578	0.08955	0.04277	0.03111	0.0279
	Kirkpatrick	<b>0.016</b>	<b>0.008</b>	<b>0.008</b>	<b>0.007</b>	<b>0.007</b>	0.00758	0.0161	0.06902	0.00885	0.04369

Tabla 6.15: Resultados del ajuste del número de iteraciones y función de enfriamiento para el problema táctico

	Nº de controladores excedido	Restricciones incumplidas								Valor objetivo					Tiempo de cómputo (min)						
		C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9
Nº iteraciones	C. Adaptativas	1.3	0	0	0	0	59	2	2.5	8	4.3	0.681	<b>0.91</b>	<b>0.966</b>	<b>0.962</b>	<b>0.965</b>	27.6	5.3	7.6	9.8	8.8
	C. Estáticas	71.1	0	0	0	0	65.2	2	2.5	10.8	4.8	<b>0.702</b>	<b>0.91</b>	<b>0.966</b>	<b>0.958</b>	<b>0.964</b>	19.9	4	6.5	7.1	7.2
	CutOff	2.1	0	0	0	0	74.1	2	2.3	12.8	3	0.6314	0.909	0.964	0.957	<b>0.965</b>	<b>19.6</b>	3	<b>4.9</b>	<b>6.6</b>	<b>5.8</b>
Descenso Temperatura	Kirkpatrick	1.2	0	0	0	0	67.1	2	2	5.7	1.7	<b>0.708</b>	<b>0.91</b>	<b>0.965</b>	<b>0.963</b>	<b>0.966</b>	23.7	5.7	8	8.8	9.4
	Adaptativo	1.8	0	0	0	0	<b>66.8</b>	2	2.8	15.3	6.3	0.634	0.909	<b>0.966</b>	0.955	0.963	<b>21</b>	<b>2.6</b>	<b>4.9</b>	<b>6.9</b>	<b>5.1</b>

Previamente a la explicación del estudio, realizado procedemos a exponer las instancias utilizadas y justificar la elección de las mismas. Estas instancias han sido escogidas por expertos de CRIDA por ser los casos más representativos y darse situaciones que puedan poner a prueba el correcto funcionamiento del algoritmo. Las instancias son las siguientes:

1. Caso 1: Esta instancia pertenece a un turno de mañana corto del centro de control de Barcelona. Para la creación del horario de trabajo disponemos de 24 controladores

(7 para el primer núcleo y 17 para el segundo) y debemos cubrir 9 sectores abiertos simultáneamente.

La distribución inicial, es decir, el horario previsto, contaba con el mismo número de controladores aunque con un sector abierto menos. En la sectorización planificada se debían cubrir 8 sectores, pero se produce un aumento del tráfico aéreo no esperado y, por tanto, se abre un nuevo sector a las 10:30 horas hasta el final del turno.

Esto provoca un caso extremo donde la carga de trabajo es muy elevada, se realiza el horario de trabajo de dos núcleos simultáneamente y algunos sectores deben ser cubiertos por controladores pertenecientes a distintos núcleos.

2. Caso 4: Esta instancia pertenece a un turno de tarde corto del centro de control de Madrid. Para la creación del horario de trabajo disponemos de 19 controladores. La distribución inicial, es decir, el horario previsto, cuenta con el mismo número de controladores y una sectorización de 7 sectores abiertos. Por un descenso no esperado en el tráfico aéreo, se reduce progresivamente durante el turno el número de sectores abiertos hasta mantenerse en 3.

Es un caso aparentemente sencillo pero existe una serie de restricciones que pueden generar ciertos problemas, como la duración mínima de un intervalo de trabajo. Es una de las instancias propuestas más sencillas, no obstante, es escogida por un caso particular que analizaremos en la Sección 6.4.3.

3. Caso 5: Esta instancia pertenece a un turno de tarde corto del centro de control de Madrid. Para la creación del horario de trabajo disponemos de 19 controladores. La distribución inicial, es decir, el horario previsto, cuenta con 19 controladores y una sectorización de 7 sectores abiertos. Aunque siendo muy similar a la instancia anterior, la principal diferencia es la baja de un controlador pasadas 02:30 horas tras el comienzo del turno y el alta de un controlador sustituto 2:50 horas más tarde (por lo que la solución se compone de un total de 20 controladores). Al igual que en el Caso 4, se reduce el número de sectores abiertos hasta 3 sectores.

En este caso, se debe solucionar la baja de un controlador repartiendo su carga de trabajo entre el resto de controladores hasta que se produzca la nueva incorporación.

## 6.4. PROBLEMA TÁCTICO

---

4. Caso 8: Esta instancia pertenece a un turno de mañana corto del centro de control de Canarias. Para la creación del horario de trabajo disponemos de 22 controladores. La sectorización del turno no se modifica en ningún momento y se mantienen abiertos 8 sectores durante todo el turno. Se produce un imprevisto que conlleva la baja de un controlador a las 03:10 horas tras el comienzo del turno y no se produce ninguna otra incorporación para sustituir al controlador, por lo que, se debe repartir su trabajo entre el resto de controladores.

Teniendo en consideración la alta carga de trabajo, perder un controlador dificulta enormemente la resolución del problema, ya que se debe repartir la carga de trabajo entre el resto de controladores.

5. Caso 9: Esta instancia pertenece a un turno de mañana corto del centro de control de Canarias. Muy similar al Caso 8, anteriormente expuesto, la única diferencia la encontramos en que, para esta instancia, se incorpora un controlador sustituto 02:30 horas después de sufrir la baja del controlador. Al igual que en el caso anterior, repartir la carga de trabajo no es tarea fácil, pero la incorporación de un controlador facilita ligeramente este trabajo.

Se han escogido los mismos métodos que se escogieron en el problema anterior para el estudio paramétrico, todos ellos explicados en la Sección 5.2.1.

Como método para el cálculo de la temperatura inicial, se ha escogido el método propuesto por Ben-Ameur. Es el método más rápido junto con el de Kirkpatrick, como se puede observar en la Tabla 6.14.

La diferencia entre estos dos métodos es la efectividad del método de Ben-Ameur, con temperaturas mejor adaptadas a cada uno de los casos y una variación mucho menor en el cálculo de la temperatura inicial durante múltiples ejecuciones.

Para la elección de los parámetros del número de iteraciones hasta el descenso de la temperatura y de la función de enfriamiento hemos tenido en cuenta varios factores. El primero es alcanzar el número de controladores disponibles, segundo es el número de restricciones incumplidas, tercero es el valor objetivo y cuarto el tiempo de cómputo. En base a estos factores se han elegido los siguientes métodos:

1. Para el cálculo de las iteraciones hasta el descenso de temperatura decidió utilizarse

el método de cadenas estáticas. Esto se debe a que según los resultados mostrados en la Tabla 6.15, ofrece muy buenos resultados para todos los aspectos en un tiempo razonable comparado con el resto de métodos.

2. Para la elección de la función de enfriamiento escogemos la función propuesta por Kirkpatrick sin lugar a dudas. En nuestro caso ofrece mejores resultados en todos los aspectos y con un tiempo de cómputo inferior al método adaptativo.

### Definición del entorno de una solución

En esta tercera fase los entornos probados han sido los mismos que para la segunda fase, a excepción del *Entorno 6*, debido a su inferior rendimiento para la resolución de este problema. Los resultados del estudio se muestran en la Tabla 6.16.

Tabla 6.16: Resultados del estudio de los distintos entornos en el problema táctico

Entornos	Nº de controladores excedido					Restricciones incumplidas				Valor objetivo				Tiempo de cómputo (min)						
	C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9	C.1	C.4	C.5	C.8	C.9
Entorno 12	0	0	0	0	0	53	2	1	8	3	0.8783	0.9105	0.965	0.9601	0.9638	17.17	9	14.57	11.14	12.11
Entorno 15	2	0	0	0	0	78	2	2	4	1	0.6464	0.9106	0.9674	0.9652	0.9679	27.91	8.9	13.39	13.15	11.65
Entorno 17	0	0	0	0	0	33	2	0	3	0	0.8876	0.9114	0.9508	0.9654	0.9677	24.03	9.8	12.36	14.85	12.56
Entorno 25	0	0	0	0	0	35	2	0	5	0	0.872	0.9084	0.9448	0.9626	0.9667	27.58	7.2	10.32	13.16	12.27

Para medir el rendimiento de los distintos entornos se han utilizado varios criterios. El primero, la diferencia entre el número de controladores disponibles y el número de controladores necesarios, ya que, en los casos más extremos, es posible no alcanzar el número de controladores disponibles. El segundo es el número de restricciones incumplidas, mientras que el tercer criterio es el valor resultante de la función objetivo, el cual se encuentra normalizado entre 0 y 1. Finalmente, el cuarto es el tiempo de cómputo del algoritmo.

El entorno que ofrece mejores resultados es el *Entorno 17* seguido del *Entorno 25*. Podemos observar en la Tabla 6.16 un menor número de restricciones incumplidas por parte del *Entorno 17*. Además, ofrece soluciones completamente factibles en los Casos 5 y 9. También observamos que el valor objetivo de las soluciones obtenidas es ligeramente superior en la mayoría de casos al del resto de entornos y, aunque los tiempos de cómputo son peores, la diferencia no es significativa.

## 6.4. PROBLEMA TÁCTICO

---

### 6.4.3. Ejemplos de aplicación

Una vez descrita la metodología de resolución para el problema táctico, procedemos a exponer varios ejemplos de aplicación de la misma, utilizando las instancias mostradas en la Tabla 6.13, al igual que se ha realizado en el resto de problemas.

La primera instancia escogida para explicar el funcionamiento de la metodología es el Caso 1. Esta instancia pertenece al turno de mañana del centro de control de Barcelona. Para la resolución del problema se dispone de 24 controladores, 17 pertenecientes al núcleo Ruta Oeste y 7 pertenecientes al núcleo Ruta Este. El núcleo Ruta Este se mantiene durante todo el turno con una sectorización de tres sectores abiertos, mientras que el núcleo Ruta Oeste se incrementa de cinco a seis sectores a las 10:30 horas. El gran volumen de trabajo junto con la escasez de controladores, y el hecho de resolver dos núcleos simultáneamente, convierte a esta instancia en un problema difícil de resolver.

La sectorización del caso se encuentra expuesta en la Figura 6.48.

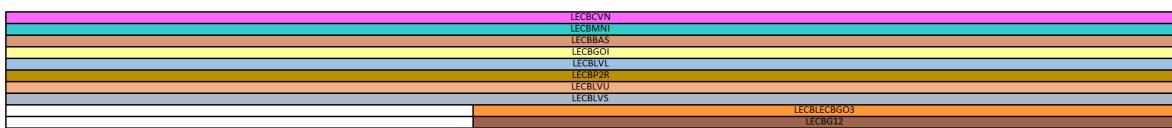


Figura 6.48: Sectorización del Caso 1

Con toda la información del caso se procede con la primera fase de la metodología, la creación de soluciones iniciales. Durante la primera fase se modifica la distribución inicial para adaptarla al imprevisto en cuestión. Durante este proceso, se pueden añadir o eliminar slots de trabajo en función de la apertura o cierre de sectores, y se pueden cerrar o abrir nuevos turnos por la baja o alta de los controladores.

A continuación, puede suceder que algunos de los turnos permanezcan sin asignar debido a que se utilizarán más turnos que controladores disponibles para construir algunas soluciones. Una vez realizado este paso, la primera fase queda finalizada, obteniendo la solución mostrada en la Figura 6.49. En la solución se puede observar como los tres primeros turnos no tienen controlador asignado (C0) y a partir del slot de cambio se ha abierto un nuevo sector y se ha sustituido uno de los sectores abiertos por otro.

Durante la siguiente fase, se reducirá el número de turnos utilizado y reasignarán los controladores, hasta que todos los turnos tengan un controlador asignado, en caso de ser

necesario.

C0	000	111	lecbg03	LECBG03	111	lecbg03	LECBG03	111	lecbg03	LECBG03	111
C1	000	111	lecbv03	LECBV03	111	lecbv03	LECBV03	111	lecbv03	LECBV03	111
C2	111	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111
C3	111	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111
C4	111	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111
C5	111	111	LECBAS	lecbas	111	LECBAS	lecbas	111	LECBAS	lecbas	111
C6	111	111	LECBVL	lecbvl	111	LECBVL	lecbvl	111	LECBVL	lecbvl	111
C7	111	111	LECBLVU	lecblvu	111	LECBLVU	lecblvu	111	LECBLVU	lecblvu	111
C8	111	111	LECBLVU	lecblvu	111	LECBLVU	lecblvu	111	LECBLVU	lecblvu	111
C9	111	111	lecbvu	LECBVL	111	lecbvu	LECBVL	111	lecbvu	LECBVL	111
C10	111	111	LECBG01	lecbg01	111	LECBG01	lecbg01	111	LECBG01	lecbg01	111
C11	111	111	LECBG01	lecbg01	111	LECBG01	lecbg01	111	LECBG01	lecbg01	111
C12	111	111	LECBG01	lecbg01	111	LECBG01	lecbg01	111	LECBG01	lecbg01	111
C13	111	111	LECBLV	lecblv	111	LECBLV	lecblv	111	LECBLV	lecblv	111
C14	111	111	LECBLV	lecblv	111	LECBLV	lecblv	111	LECBLV	lecblv	111
C15	111	111	lecbvs	LECBVS	111	lecbvs	LECBVS	111	lecbvs	LECBVS	111
C16	111	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111
C17	111	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111
C18	111	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111	LECBVN	lecbvn	111
C19	111	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111
C20	111	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111	LECBMN	lecbmn	111
C21	111	111	LECBAS	lecbas	111	LECBAS	lecbas	111	LECBAS	lecbas	111
C22	111	111	LECBAS	lecbas	111	LECBAS	lecbas	111	LECBAS	lecbas	111
C23	111	111	LECBAS	lecbas	111	LECBAS	lecbas	111	LECBAS	lecbas	111
C24	111	111	LECBAS	lecbas	111	LECBAS	lecbas	111	LECBAS	lecbas	111

Figura 6.49: Solución inicial del Caso 1

A partir de la solución inicial comienza la segunda fase, durante la cual, además de tener como objetivo reducir el número de controladores, también se eliminan todas las restricciones incumplidas y se optimizan el resto de objetivos. Esto se lleva a cabo mediante un algoritmo basado en el recocido simulado, cuya ejecución se realiza en base a un conjunto de parámetros, los cuales han sido estudiados y ajustados previamente.

Una vez finaliza la ejecución de la segunda fase, obtenemos la solución óptima, la cual se muestra en la Figura 6.50.

C1	111	LECBLV	lecbvl	111	LECB03	111	lecbvl	111	LECBLV	lecbvl	111
C2	111	lecbvl	111	LECBLV	lecbvl	111	lecbvl	111	LECBLV	lecbvl	111
C3	111	lecbvl	111	LECB03	lecb03	111	LECB03	111	LECB03	lecb03	111
C4	111	lecb03	111	LECB03	lecb03	111	LECB03	111	LECB03	lecb03	111
C5	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C6	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C7	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C8	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C9	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C10	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C11	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C12	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C13	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C14	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C15	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C16	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C17	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C18	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C19	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C20	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C21	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C22	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C23	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111
C24	111	111	LECB03	lecb03	111	LECB03	lecb03	111	LECB03	lecb03	111

Figura 6.50: Solución óptima del Caso 1

En la Tabla 6.17 mostramos los resultados de la ejecución junto con el resto de casos propuestos. Como se puede apreciar en la tabla, cada columna contiene los valores referentes a los distintos criterios por los cuales se evalúan las soluciones. Podemos encontrar los valores resultantes de la función objetivo, el número de controladores necesarios para dar solución al problema, el número de restricciones incumplidas y el tiempo de resolución utilizado.

Podemos observar que los casos en los que obtiene soluciones factible son los casos en los que la ejecución finaliza más rápido. Mientras que el caso en que peor funciona

## 6.4. PROBLEMA TÁCTICO

---

Tabla 6.17: Resultados explicativos de la aplicación de la metodología de resolución sobre los casos 1,4,5,8,9.

	Nº de controladores requeridos	Restricciones incumplidas	Cambios en sala	Valor Objetivo	Tiempo de resolución (min.)
<b>Caso 1</b>	24	28	11	0.8814	27.38
<b>Caso 4</b>	19	2	10	0.9111	9.59
<b>Caso 5</b>	20	0	4	0.9472	11.3
<b>Caso 8</b>	22	3	2	0.9653	14.05
<b>Caso 9</b>	23	0	4	0.9517	12.2

es el Caso 1, donde encontramos dos núcleos de trabajo. En conjunto, los resultados del algoritmo son muy prometedores, y los tiempos de ejecución permiten obtener resultados en el tiempo que se dispone.

Antes de mostrar las sectorizaciones y soluciones de todos los casos expuestos, realizaremos una breve descripción para recordar sus características.

El **Caso 4** pertenece a un turno de tarde del centro de control de Madrid que abarca desde las 15:00h hasta las 22:30h. Para la creación del horario de trabajo disponemos de 19 controladores asignados al núcleo TMA. Se estima una sectorización de 7 sectores abiertos durante todo el turno, pero por un descenso del tráfico esperado se reduce progresivamente el número de sectores abiertos a lo largo del turno.

Este caso no tiene una solución completamente factible por un motivo sencillo, cuando se produce el cierre de un sector, es posible que los controladores que trabajen en el mismo no cumplan una restricción laboral que establece un trabajo mínimo obligatorio. Por esta razón, si los controladores se encuentran trabajando en el sector que se cierra es posible que no exista ninguna solución que cumpla con esta condición. La manera en que debe resolverse este problema será retrasar unos minutos el cierre del sector siempre que esto sea posible.

La sectorización de este caso se encuentra en la Figura 6.51 mientras que las soluciones inicial y óptima se encuentran en las Figuras 6.52 y 6.53, respectivamente.

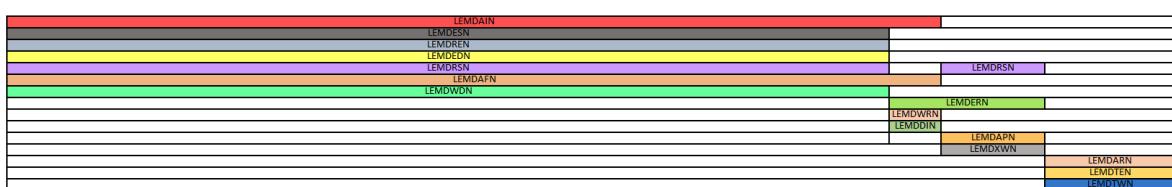


Figura 6.51: Sectorización del Caso 4 y 5

## CAPÍTULO 6

C1	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdwtn	lemdrsn	MDRS	LEMDARN	
C2	111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdwtn	lemdrsn	MDRS	lemdarn
C3	LEMDRSN		111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdwtn	lemdrsn	MDRS	lemdarn	111
C4	lemdesn	LEMDESN	111	lemdesn	LEMDESN	111	lemdesn	LEMDESN	111	lemdesn	lemdesn	LEMDERN	lemdern	
C5	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	LEMDXWN	111
C6	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	LEMDXWN	111
C7	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdain	LEMDAIN	111	lemdten
C8	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdain	LEMDAIN	111	lemdten
C9	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdain	LEMDAIN	111	lemdten
C10	lemdafn	LEMDAFN	111	lemdafn	111									
C11	LEMDAFN	111	lemdafn	LEMDAFN	111									
C12	lemdren	LEMDREN	111	lemdren	111									
C13	LEMDREN	111	lemdren	LEMDREN	111									
C14	LEMDEDN	111	lemdedn	LEMDTWN										
C15	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDERN	LEMDERN	111
C16	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDERN	LEMDERN	111
C17	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDERN	LEMDERN	111
C18	lemdwtn	LEMDWTON	111	lemdwtn	LEMDARN									
C19	LEMDWTON	111	lemdwtn	LEMDARN	lemdarn									

Figura 6.52: Solución inicial del Caso 4

C1	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdwtn	lemdrsn	LEMDERN	lemdern	111	
C2	111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdrsn	LEMDRSN	111	lemdwtn	lemdrsn	LEMDERN	111	
C3	LEMDRSN		111	lemdesn	LEMDESN	111	lemdesn	LEMDESN	111	lemdesn	LEMDAIN	111	lemdwtn	LEMDRSN	111
C4	lemdesn	LEMDESN	111	lemdesn	LEMDESN	111	lemdesn	LEMDESN	111	lemdesn	LEMDAIN	111	lemdwtn	LEMDRSN	111
C5	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	LEMDRSN	111	
C6	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	LEMDRSN	111	
C7	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	
C8	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	
C9	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdain	LEMDAIN	111	lemdesn	LEMDAIN	111	lemdwtn	
C10	lemdafn	LEMDAFN	111	lemdafn	LEMDAFN	111									
C11	LEMDAFN	111	lemdafn	LEMDAFN	111										
C12	lemdren	LEMDREN	111	lemdren	LEMDAFN										
C13	LEMDREN	111	lemdren	LEMDAFN	111										
C14	LEMDEN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDTWN		
C15	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDERN	LEMDERN	111	
C16	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDERN	LEMDERN	111	
C17	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDEDN	111	lemdedn	LEMDERN	LEMDERN	111	
C18	lemdwtn	LEMDWTON	111	lemdwtn	LEMDARN										
C19	LEMDWTON	111	lemdwtn	LEMDARN	lemdarn										

Figura 6.53: Solución óptima del Caso 4

El **Caso 5** es un caso muy similar al Caso 4, el cual pertenece a un turno de tarde del centro de control de Madrid que abarca de 15:00h a 22:30h. Para la creación del horario de trabajo disponemos de 19 controladores asignados al núcleo TMA. Se estima una sectorización de 7 sectores abiertos durante todo el turno, pero por un descenso del tráfico esperado se reduce progresivamente el número de sectores abiertos a lo largo del turno.

La diferencia con el Caso 4 es la siguiente: antes de que se reduzca la carga de trabajo, uno de los controladores es dado de baja a las 17:30h, y el controlador sustituto no puede presentarse hasta las 20:20, por lo que debe repartirse el trabajo de este controlador entre el resto durante ese período.

En este caso, se encuentra una solución completamente factible que cumple todas las restricciones. La sectorización coincide con la del Caso 4 y se encuentra en la Figura 6.51, mientras que las soluciones inicial y óptima se encuentran en las Figuras 6.54 y 6.55, respectivamente.

El **Caso 8** pertenece a un turno de mañana del centro de control de Canarias que abarca de 7:00h a 15:00h. Para la creación del horario de trabajo disponemos de 22 controladores asignados al núcleo Palma APP. Se estima una sectorización de 8 sectores abiertos durante todo el turno. Se produce la baja inesperada de un controlador a mitad de turno (10:10h), y no se dispone de ningún controlador sustituto, por lo que el resto del

## 6.4. PROBLEMA TÁCTICO

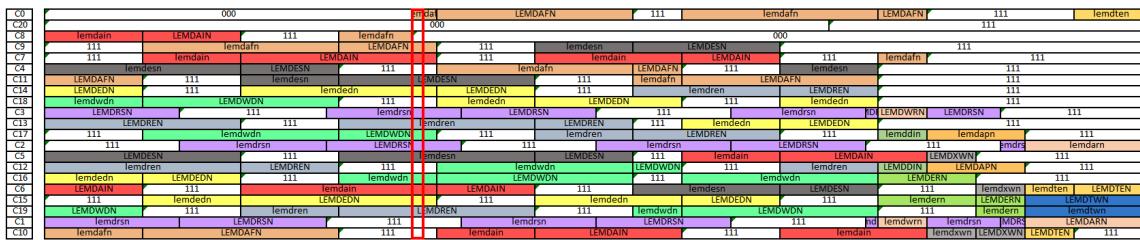


Figura 6.54: Solución inicial del Caso 5

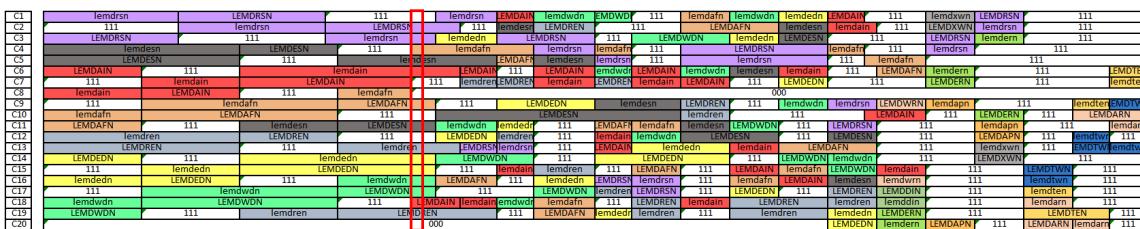


Figura 6.55: Solución óptima del Caso 5

turno, los controladores restantes deberán asumir su carga de trabajo.

Se dispone de un número muy reducido de controladores en comparación a la carga de trabajo, perder un controlador dificulta enormemente y posiblemente elimine la opción de crear un horario de trabajo donde se cumplan todas las restricciones.

La sectorización del Caso 8 se encuentra en la Figura 6.56 mientras que las soluciones inicial y óptima se encuentran en las Figuras 6.57 y 6.58, respectivamente.



Figura 6.56: Sectorización del Caso 8 y 9

El **Caso 9**, es un caso muy similar al Caso 8, pertenece a un turno de mañana del centro de control de Canarias que abarca de 7:00h a 15:00h. Para la creación del horario de trabajo disponemos de 22 controladores asignados al núcleo Palma APP. Comparte sectorización con el Caso 8, y esta no es modificada en ningún momento. Se produce la baja inesperada de un controlador a mitad de turno (10:10h), pero en este caso, sí disponemos de un controlador sustituto capaz de incorporarse a las 12:40h.

Los Casos 8 y 9 son casos muy similares, en el que se dispone de un número reducido de controladores en comparación a la carga de trabajo, pero en el Caso 9 al disponer de un controlador sustituto, se concede un ligero margen para equilibrar mejor la carga de

CAPÍTULO 6

Figura 6.57: Solución inicial del Caso 8

C1	aae	AAE	111	aa	AAE													
C2	111	aae	AAE	111	aa	AAE	111	aa	AAE	aa	111	aa	AAE	aa	111	aa	AAE	
C3	AAE	111	aae	AAE	111	aa	ABE	111	aa	AAE	aa	111	aa	AAE	aa	111	aa	AAE
C4	aa	AAX	111	aa	AAX	111	aa	AAX	111	aa	abe	aa	AAX	aa	abe	aa	AAX	
C5	AAX	111	aa	AAX	111	aa	AAX	111	aa	ABE	aa	AAX	aa	ABE	aa	AAX		
C6	ABC	111	abc	AAZ	111	aaZ	ABC	ABE										
C7	abc	111	ABC	aaZ	111	aaZ	abc	ABC										
C8	AAU	aaU	111	aa	AAU	aaU	aa	AAU	aaU	aa	ABE	aaU	AAU	aaZ	111	aaZ	aa	
C9	ABE	aaU	111	aa	ABE	aaU	aa	ABE	aaU	aa	ABE	aaU	AAU	aaZ	111	aaZ	aa	
C10	ABE	aaU	111	aa	ABE	aaU	aa	ABE	aaU	aa	ABE	aaU	AAU	aaZ	111	aaZ	aa	
C11	ABE	aaJ	111	aa	ABE	aaJ	aa	ABE	aaJ	aa	ABE	aaJ	AAU	aaZ	111	aaZ	aa	
C12	abe	AAJ	111	aa	ABE	aaJ	aa	ABE	aaJ	aa	ABE	aaJ	AAU	aaZ	111	aaZ	aa	
C13	111	ABE	aae	111	ABE	aae	aa	ABE	aae	aa	abf	aaJ	AAE	111	abc	aaJ	aa	
C14	111	ABE	aae	111	ABE	aae	aa	ABE	aae	aa	abf	aaJ	AAE	111	abc	aaJ	aa	
C15	AAE	111	aa	AAE	111	aa	AAE	111	aa	AAU	aa	AAU	aa	AAU	aa	AAE	111	AAU
C16	aaZ	111	AAZ	aaZ	111	AAZ	aaZ	111	AAZ	aaZ	aa	AAU	aa	AAU	aa	AAU	aa	AAU
C17	AAU	aaZ	111	aa	AAU	aaZ	aa	AAU	aaZ	aa	AAU	aaZ	AAU	aa	AAU	aa	AAU	
C18	aaU	AAZ	111	aa	AAU	aaU	aa	AAU	aaU	aa	AAU	aaU	AAU	aa	AAU	aa	AAU	
C19	ABF	aaU	111	aa	ABF	aaU	aa	ABF	aaU	aa	abf	aaU	ABF	aa	ABF	aa	ABF	
C20	abf	AAU	111	aa	ABF	aaU	aa	ABF	aaU	aa	abf	aaU	ABF	aa	ABF	aa	ABF	
C21	111	ABF	abf	111	ABF	abf	aa	ABF	abf	aa	abf	ABF	aa	ABF	111	abf	aa	ABF
C22	111	ABF	abf	111	ABF	abf	aa	ABF	abf	aa	abf	ABF	aa	ABF	111	abf	aa	ABF

Figura 6.58: Solución óptima del Caso 8

trabajo. Por esto, todavía es viable la opción de crear un horario de trabajo donde se cumplan todas las restricciones.

La sectorización coincide con la del Caso 8 y se encuentra en la Figura 6.56 mientras que las soluciones inicial y óptima se encuentran en las Figuras 6.59 y 6.60, respectivamente.

Figura 6.59: Solución inicial del Caso 9

## 6.4. PROBLEMA TÁCTICO

---

C1	aee	AAE	AAE	aee	AAE	AAX	aax	AAU	111	111	AAE	aax	111	AAE	aee
C2	111	aee	AAE	111	aee	AAE	aax	AAU	111	111	AAE	aax	AAU	111	AAE
C3	AAE	111	111	aee	AAE	111	aax	AAU	111	111	AAE	aax	AAU	111	AAE
C4	aax	AAE	AAE	111	aax	AAE	AAE	AAU	111	111	AAE	aax	AAU	111	AAE
C5	111	aax	AAE	AAE	111	aax	AAE	AAU	111	111	AAE	aax	AAU	111	AAE
C6	AAX	111	111	aax	AAE	AAE	AAE	AAU	111	111	AAE	aax	AAU	111	AAE
C7	ABe	111	111	abc	ABe	ABe	ABe	ABe	111	111	AAE	ABe	ABe	111	AAE
C8	abc	111	111	ABC	abc	abc	abc	abc	111	111	aee	abc	abc	111	aee
C9	AAJ	abc	111	aaJ	AAJ	AAJ	AAJ	AAJ	111	111	AAE	abc	abc	111	AAJ
C10	aaJ	ABC	111	aaJ	AAJ	AAJ	AAJ	AAJ	111	111	AAE	abc	abc	111	aaJ
C11	ABE	aaJ	111	abe	ABE	ABE	ABE	ABE	111	111	AAU	abc	abc	111	aaJ
C12	abe	AAJ	111	ABE	111	ABE	ABE	ABE	111	111	AAU	abc	abc	111	AAJ
C13	111	ABE	abe	abe	111	abc	abc	ABC	111	111	AAU	abc	ABC	111	ABE
C14	111	abe	ABE	ABE	111	aaZ	aaZ	AAZ	111	111	AAZ	abc	ABC	111	AAZ
C15	aaZ	aaZ	111	aaZ	AAZ	AAZ	AAZ	AAZ	111	111	AAZ	aaZ	AAZ	111	aaZ
C16	aaZ	111	111	aaZ	AAZ	AAZ	AAZ	AAZ	111	111	AAZ	aaZ	AAZ	111	aaZ
C17	AAU	aaZ	111	aaZ	AAU	AAU	AAU	AAU	111	111	AAZ	aaZ	AAZ	111	aaZ
C18	aaU	AAZ	111	aaU	AAU	AAU	AAU	AAU	111	111	AAZ	aaU	AAU	111	aaU
C19	ABF	aaU	111	abf	ABF	ABF	ABF	ABF	111	111	AAU	aaU	AAU	111	AAU
C20	abf	ABF	111	abf	ABF	ABF	ABF	ABF	111	111	ABF	abf	ABF	111	ABF
C21	111	ABF	abf	abf	111	ABF	abf	ABF	111	111	ABF	abf	ABF	111	ABF
C22	111	abf	ABF	ABF	111	ABF	abc	abf	111	111	ABF	abf	ABF	111	abf
C23					000				000						

Figura 6.60: Solución óptima del Caso 9

---

## CAPÍTULO 6

---

## Capítulo 7

# Desarrollo de la herramienta software

---

En este capítulo explicaremos el proceso que se ha seguido durante toda la etapa de creación del sistema en base a las etapas de la Ingeniería del Software. El proceso de ingeniería del software es un proceso que en el que se aplican principios y metodologías en el desarrollo y mantenimiento de sistemas software.

La metodología de Ingeniería del Software consta de varias etapas, en las cuales se realizan distintos documentos plasmando la información necesaria para llevar a cabo el proyecto. Consideramos que el presente documento no tiene el propósito de reflejar todo el procedimiento completo realizado mediante ingeniería del software, por lo que se resumirán y abreviarán las diferentes etapas del proceso y toda la información obtenida de las mismas.

Otro punto a tener en consideración es que tenemos tres problemas relacionados pero de resolución independiente entre sí. Por ello, podría considerarse la necesidad de llevar a cabo tres procesos de ingeniería del software para cada uno de los sistemas implementados. No obstante, los tres problemas comparten características y objetivos comunes, por lo que se engloban todos en el mismo proyecto.

Las distintas etapas que se pueden encontrar en la ingeniería del software son:

- *Análisis de requisitos*: En este proceso se genera un documento que recoge todos los requisitos que debe cumplir el sistema. Existen multitud de tipos de requisitos, pero en este caso nos centraremos en los requisitos funcionales y no funcionales.
- *Diseño y arquitectura*: En este proceso se elabora un documento en el que se plasma el diseño de distintos sistemas que se utilizarán en el desarrollo del proyecto y cómo estos interactúan entre sí para cumplir con los objetivos. Dicho documento suele

ir acompañado de diferentes tipos de diagramas que faciliten el entendimiento del mismo. Algunos de los más utilizados son los diseños arquitectónicos, diagramas de clases, diseños de bases de datos, etc.

- *Desarrollo*: En esta etapa se realiza la implementación y desarrollo de la aplicación. Es frecuente el uso de distintas herramientas como los entornos de programación, repositorios de código, sistemas de control de versiones, etc. Estas herramientas permiten que un equipo de desarrollo trabaje de forma eficiente y rápida.
- *Pruebas de software*: Este proceso consiste en el diseño de un conjunto de pruebas que se ejecutan para comprobar que el software realiza correctamente las tareas indicadas. Uno de los procesos más comunes es el uso de pruebas unitarias en paralelo a la etapa de desarrollo y posteriormente, el uso de pruebas de integración aplicadas al sistema completo.
- *Documentación y mantenimiento*: En la mayoría de procedimientos de ingeniería del software esta etapa se encuentra dividida en dos. La primera, es la etapa de documentación, donde se realizan todos los documentos necesarios para el desarrollo y la gestión del software, incluyendo manuales, diagramas de casos de uso, descripción del código, etc.

La otra etapa, la etapa de mantenimiento, consiste en la mejora y mantenimiento del software, ya sea corrección de errores o incorporación de nuevas funcionalidades, lo cual debe ir siempre acompañado de su respectiva documentación.

## 7.1. Análisis de requisitos

Para comenzar con el análisis de requisitos, debemos comprender cuáles son los objetivos del sistema, es decir, qué se espera que el sistema haga.

El objetivo descrito de forma general sería, dada la entrada del problema (sectorización, lista de controladores, centro de control, detalles del turno...), construir un horario de trabajo para los controladores aéreos que satisfaga todas las condiciones laborales y, al mismo tiempo, que todos los sectores se encuentren cubiertos en todo momento.

## 7.1. ANÁLISIS DE REQUISITOS

---

Dado que esta descripción es muy genérica, procedemos a exponer el proceso que se ha seguido en el caso concreto del análisis de requisitos realizado en la versión completa del problema.

El objetivo del sistema es la creación de un horario de trabajo en el cual se asignen un conjunto de controladores aéreos disponibles a los distintos sectores abiertos a lo largo del turno. Además, el horario de trabajo debe cumplir las restricciones expuestas en la Sección 3.2.1.

Una vez se conoce el objetivo del sistema, procedemos a determinar los requisitos funcionales del mismo.

### 7.1.1. Análisis de requisitos funcionales

Los requisitos funcionales se definen como requisitos que indican qué debe hacer el sistema, es decir, la funcionalidad del mismo. Para facilitar el entendimiento de los requisitos realizaremos una agrupación de algunos requisitos en función de las distintas fases que forman la metodología de resolución.

1. La entrada con la información de la instancia a resolver debe estar compuesta por tres ficheros referentes a la instancia y otros cuatro ficheros referentes al centro de control.
2. El sistema debe comprobar la integridad de la entrada al mismo. Esto se realizará comprobando que no existan discrepancias entre la información que contienen los ficheros de entrada.
3. El sistema permitirá modificar los tiempos asignados a cada una de las restricciones sin necesidad de modificar la implementación.
4. El sistema realizará un conjunto de comprobaciones previas al proceso de creación del horario de trabajo con el objetivo de comprobar si con los datos de entrada, se puede facilitar un horario que cumpla con todas las restricciones del problema o no existe ningún horario que cumpla con las restricciones para dichos datos.
5. La primera fase debe proporcionar un horario de trabajo que cumpla las siguientes características:

- a) El horario de trabajo debe estar formado por una matriz de turnos y una lista de controladores.
  - b) La matriz de turno debe estar formada por un número de columnas igual a la longitud del turno (cada celda representa un período de cinco minutos) y un número de filas igual o superior al número de controladores. El contenido de cada celda será el identificador de un sector, el cual es una cadena de texto formada por tres caracteres alfabéticos.
  - c) La lista de controladores debe contener la siguiente información para cada uno de los controladores: Identificador del controlador, turno de trabajo, núcleo, acreditación, filas asignada.
  - d) Todos los sectores abiertos en la sectorización deben estar cubiertos en la matriz de turnos en todo momento y en ambas posiciones por únicamente un controlador por posición.
  - e) Todos los controladores deben tener un único turno asignado.
6. La segunda fase debe proporcionar un horario que cumpla, además de las características anteriormente mencionadas, las siguientes:
- a) El horario de trabajo debe cumplir todas las restricciones descritas en la Sección 3.2.1.
  - b) El horario de trabajo debe tener el mismo número de turnos que controladores disponibles.
7. En caso de que no se encuentre un horario que cumpla las características anteriormente mencionadas, el sistema no deberá ejecutar la tercera fase de la metodología de resolución.
8. La tercera fase debe proporcionar un horario que cumpla, con todas las características anteriores, así como las siguientes:
- a) Cada turno del horario de trabajo debe aproximarse lo más posible a los tiempos de trabajo óptimo continuado, trabajo óptimo en una posición y descanso óptimo continuado.

## 7.1. ANÁLISIS DE REQUISITOS

---

- b) La estructura del horario debe asemejarse en la medida de lo posible a la estructura de un horario creado a partir de plantillas.
  - c) El horario debe maximizar el número de sectores elementales por los que pasan los controladores a lo largo del turno.
  - d) El horario debe reducir el número de cambios en sala.
  - e) La carga de trabajo de los controladores debe encontrarse lo más balanceada posible entre todos los turnos.
9. El sistema debe proporcionar una salida legible con la información necesaria, dando a conocer a cada uno de los controladores el trabajo por realizar durante el turno.
10. En el fichero que contenga la salida, cada sector debe tener asignado un color para facilitar su entendimiento y debe diferenciarse claramente la posición de control que se asigne a cada controlador.
11. Si no se obtiene un horario que cumpla todas las restricciones en el fichero de salida debe notificarse, junto con el número de restricciones incumplidas y qué tipo de restricciones se incumplen.

### 7.1.2. Análisis de requisitos no funcionales

Los requisitos no funcionales se definen como requisitos que describen cómo debe hacer la tarea el sistema. Estos requisitos son los siguientes:

1. El sistema debe proporcionar un horario de trabajo en menos de dos horas.
2. El sistema debe proporcionar una respuesta rápida en caso de que las condiciones definidas en la entrada no permitan crear un horario factible, entendiendo por respuesta rápida un mensaje en un tiempo inferior a cinco minutos informando de la situación.
3. El sistema debe permitir la integración con otros sistemas y plataformas.
4. El sistema debe contar con un proceso de control de errores que permita su detección rápidamente.

5. El sistema debe contar con manuales de uso y la documentación completa de la codificación.
6. El sistema debe tener integradas todas las librerías y aplicaciones necesarias para su ejecución.

## 7.2. Diseño y arquitectura

En este proyecto podemos prescindir de la arquitectura, ya que el sistema no necesita de ningún tipo de interacción con otros sistemas para su funcionamiento. Para la etapa de diseño, nos centraremos en el uso de modelos de procesos de negocio (BPM, *business process management*) y el modelo especificativo del sistema.

En la Figura 7.1 observamos el modelo especificativo del sistema representado mediante UML (*Unified Modeling Language*), del cual se han excluido todas las partes prescindibles para el funcionamiento del sistema. Algunos ejemplos son los ficheros con las pruebas realizadas, el código necesario para el análisis de parámetros o la generación de las trazas del algoritmo. Todo lo anterior es necesario para asegurar el correcto funcionamiento del sistema en un primer momento, pero una vez cumplida su función, puede ser eliminado con el objetivo de simplificar el código y reducir los tiempos de cómputo.

## 7.2. DISEÑO Y ARQUITECTURA

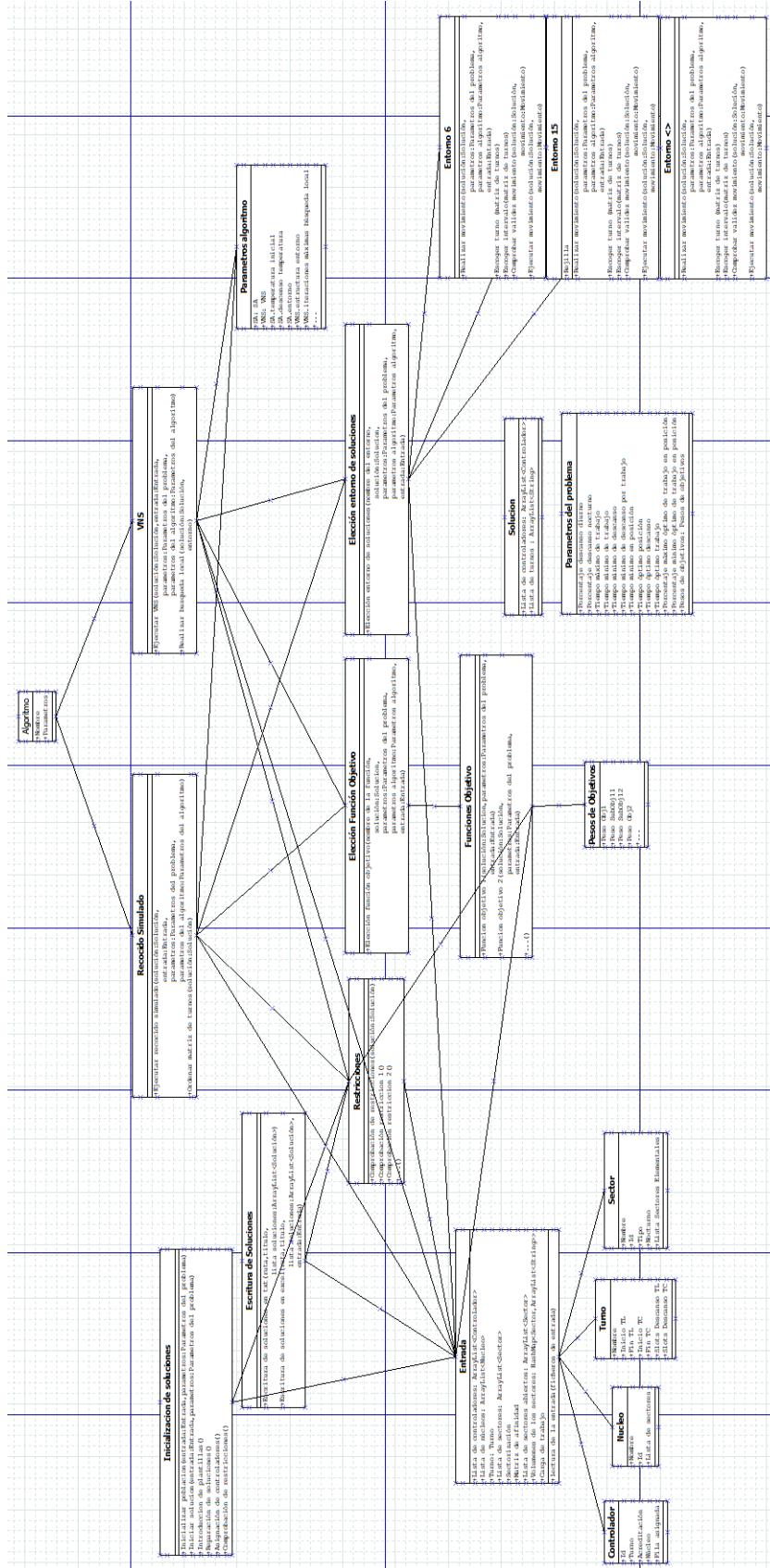


Figura 7.1: Modelo específico del sistema

El modelo UML expuesto en la Figura 7.1, muestra la estructura del proyecto, las distintas clases y la interacción entre estas. Incluye un mayor nivel de detalle mostrando alguno de los métodos y atributos de cada una de las clases. Se puede ver como se dispone de una clase principal que deriva en los dos algoritmos utilizados para la resolución del problema, el SA y el VNS. Ademas tenemos distintas clases agrupadas para la comprobación de restricciones, el cálculo de la función objetivo, o la lectura y tratamiento de los datos de entrada; entre otras. Estas clases son comunes para ambas metaheurísticas. No obstante, tenemos clases propias para cada uno de los algoritmos como pueden ser las clases que contienen los movimientos para generar las soluciones del entorno de una solución.

Como ya hemos mencionado anteriormente se están resolviendo tres problemas distintos, los cuales cada uno de ellos cuenta con un proyecto específico.

En la Figura 7.2 encontramos el BPM correspondiente al proceso de creación de un horario de trabajo, empezando por la lectura de los ficheros de entrada y finalizando por la escritura del horario en un fichero.

El sistema no necesita un hardware específico ni interacciona con más de un equipo, tampoco requiere interfaz ni base de datos, por lo que consideramos que los modelos aportados contienen información suficiente para el diseño del sistema.

### 7.3. Implementación

En esta sección veremos todas las herramientas relacionadas con la programación utilizadas durante todo el proceso. La implementación del proyecto completo se ha realizado en el lenguaje de programación *Java* con el entorno de programación *Eclipse*. Además, se han utilizado otro conjunto de herramientas y librerías las cuales mencionaremos a continuación.

Se ha hecho uso de la herramienta *GitHub* como repositorio online del proyecto y gestión del control de versiones. Como cliente para el repositorio *Git* se ha utilizado *SourceTree*, que provee una interfaz gráfica para la gestión del repositorio.

También se ha utilizado *Javadoc* para generar la documentación pertinente en formato *HTML* del código implementado.

## 7.4. PRUEBAS DE SOFTWARE

---

Para la representación de soluciones y trazas de las ejecuciones, se ha hecho uso de una librería de *Apache* llamada *POI*, para lectura y escritura de archivos de *Microsoft Office*. En este caso, restringimos su uso a los ficheros *.xls* o *.xlsx*, los cuales se trabajan con el programa *Microsoft Excel*. Cabe destacar que la librería no permite crear gráficas, pero permite modificar los datos de estas, por lo que una vez creada una plantilla, se pueden crear informes de las trazas incluyendo gráficos a partir de la copia y modificación de la misma. Esta funcionalidad ha sido de gran utilidad a la hora de realizar el análisis del algoritmo y poder optimizarlo.

Igualmente, se ha hecho uso de otras librerías de *Apache* como: *Commons Math*, *Commons Collections* y *XMLBeans*.

## 7.4. Pruebas de software

En esta sección explicaremos el proceso pruebas de software realizado y cómo se ha diseñado. Al igual que en las secciones anteriores, en esta sección se ha tenido que definir un conjunto de pruebas distinto para cada uno de los distintos problemas resueltos.

No obstante, algunas de las pruebas realizadas en algunos problemas se han podido reutilizar en otros. A continuación, expondremos el proceso de pruebas de software realizado para la versión completa del problema.

Durante el diseño de las pruebas unitarias, se han diseñado distintos casos de uso para comprobar el cumplimiento de los requisitos funcionales previamente definidos. El objetivo es diseñar un conjunto de pruebas que comprueben el correcto funcionamiento del código dados todos los posibles casos de uso, así como las distintas entradas al sistema.

También se han realizado distintas pruebas unitarias independientes a los casos de uso para comprobar que una actualización del sistema, o una modificación del mismo, no pueda generar errores que pasen desapercibidos. Todas las pruebas diseñadas se encuentran en la Tabla 7.1.

Además del conjunto de pruebas expuesto, destinado a comprobar el correcto funcionamiento de la herramienta y detectar posibles fallos en la codificación, se ha comprobado que la herramienta otorgue soluciones válidas para instancias reales. Este paso es fundamental ya que muchos prototipos o primeras versiones de algunos software funcionan

## CAPÍTULO 7

---

Tabla 7.1: Resultados de las pruebas unitarias

Categoría	Relación R.F.	Nombre	Datos de prueba	Respuesta esperada	Respuesta obtenida
Entrada	1	Ficheros de entrada correctos	<i>AperturaSectorizaciones_IdIm-06-03-2017.csv</i> <i>RecursosDisponibles_IdIm-06-03-2017.csv</i> <i>Turno_IdIm-06-03-2017.csv</i>	Ninguna	Ninguna
Entrada	1	Ficheros de entrada incorrectos 1	<i>AperturaSectorizaciones_IdIm-06-03-2017.csv</i> <i>RecursosDisponibles_IdIm-06-03-2017.csv</i> <i>AperturaSectorizaciones_IdNull.csv</i>	Falta fichero	Falta fichero
Entrada	1	Ficheros de entrada incorrectos 2	<i>AperturaSectorizaciones_IdIm-06-03-2017.csv</i> <i>RecursosDisponibles_IdIm-06-03-2017.csv</i> <i>Turno_IdNull.csv</i>	Fichero <nombre> no encontrado	Fichero <nombre> no encontrado
Entrada	2	Duración turno correcta	Fichero AperturaSectorizaciones: 6:20 a 14:40	Ninguna	Ninguna
Entrada	2	Duración turno incorrecta	Fichero Turno: 6:20 a 14:40	Comprobar inicio y fin del turno	Comprobar inicio y fin del turno
Entrada	2	Sectores abiertos correctos	Fichero SectoresBarcelona: LECBBAS	Ninguna	Ninguna
Entrada	2	Sectores abiertos incorrectos	Fichero SectoresBarcelona: LECBBAS Fichero AperturaSectorizaciones: LECBBAX	Comprobar sectores abiertos	Comprobar sectores abiertos
Entrada	4	Carga trabajo correcta	Fichero Recursos disponibles: 22	Ninguna	Ninguna
Entrada	4	Carga trabajo incorrecta	Fichero AperturaSectorizaciones: 8 Sectores Fichero Recursos disponibles: 20	Carga de trabajo excesiva, no existe solución factible	Carga de trabajo excesiva, no existe solución factible
1º Fase	5	Características horario correctas	Matriz de turnos Lista controladores	Ninguna	Ninguna
1º Fase	5	Características horario incorrectas	Matriz de turnos incorrecta Lista controladores	Error de código	Error de código
2º Fase	6.a, 7	Características horario correctas	Sin restricciones incumplidas	Ninguna	Ninguna
2º Fase	6.a, 7	Características horario incorrectas	Restricciones incumplidas	La restricción <n> es incumplida La restricción <n> es incumplida Ejecución detenida	La restricción <n> es incumplida La restricción <n> es incumplida Ejecución detenida
2º Fase	6.b, 7	Características horario correctas	Controladores suficientes	Ninguna	Ninguna
2º Fase	6.b, 7	Características horario incorrectas	Controladores insuficientes	Se requieren más controladores Ejecución detenida	Se requieren más controladores Ejecución detenida
3º Fase	8	Características horario correctas	Sin restricciones incumplidas	Ninguna	Ninguna
3º Fase	8	Características horario incorrectas	Restricciones incumplidas	Error de código	Error de código
Salida	9, 10, 11	Características de la salida correctas	Fichero salida correcto	Ninguna	Ninguna
Salida	9, 10, 11	Características de la salida incorrectas	Fichero salida incorrecto	Error de código	Error de código

## **7.5. DOCUMENTACIÓN Y MANTENIMIENTO**

---

correctamente con instancias artificiales, pero a la hora de resolver las instancias reales para las que se ha implementado no son capaces. Por lo que se ha probado el funcionamiento con instancias reales, y se han validado los resultados de la herramienta contando con el visto bueno de los expertos pertenecientes a CRIDA.

### **7.5. Documentación y mantenimiento**

El proceso de documentación del código no solo incorpora la propia documentación existente en el código y la documentación generada mediante *Javadoc*. En este mismo documento podemos encontrar información sobre el dominio del problema o todos los procesos algorítmicos implementados entre las distintas clases del proyecto.

En la documentación generada podemos encontrar la estructura del proyecto, como se puede apreciar en la Figura 7.3, y, además encontraremos comentarios explicativos sobre las distintas clases y objetos existentes (Figura 7.4). Igualmente, se muestran los distintos constructores, así como los métodos de cada una de las clases justo con los parámetros utilizados, todo ello descrito y explicado como se muestra en las Figuras 7.5 y 7.6.

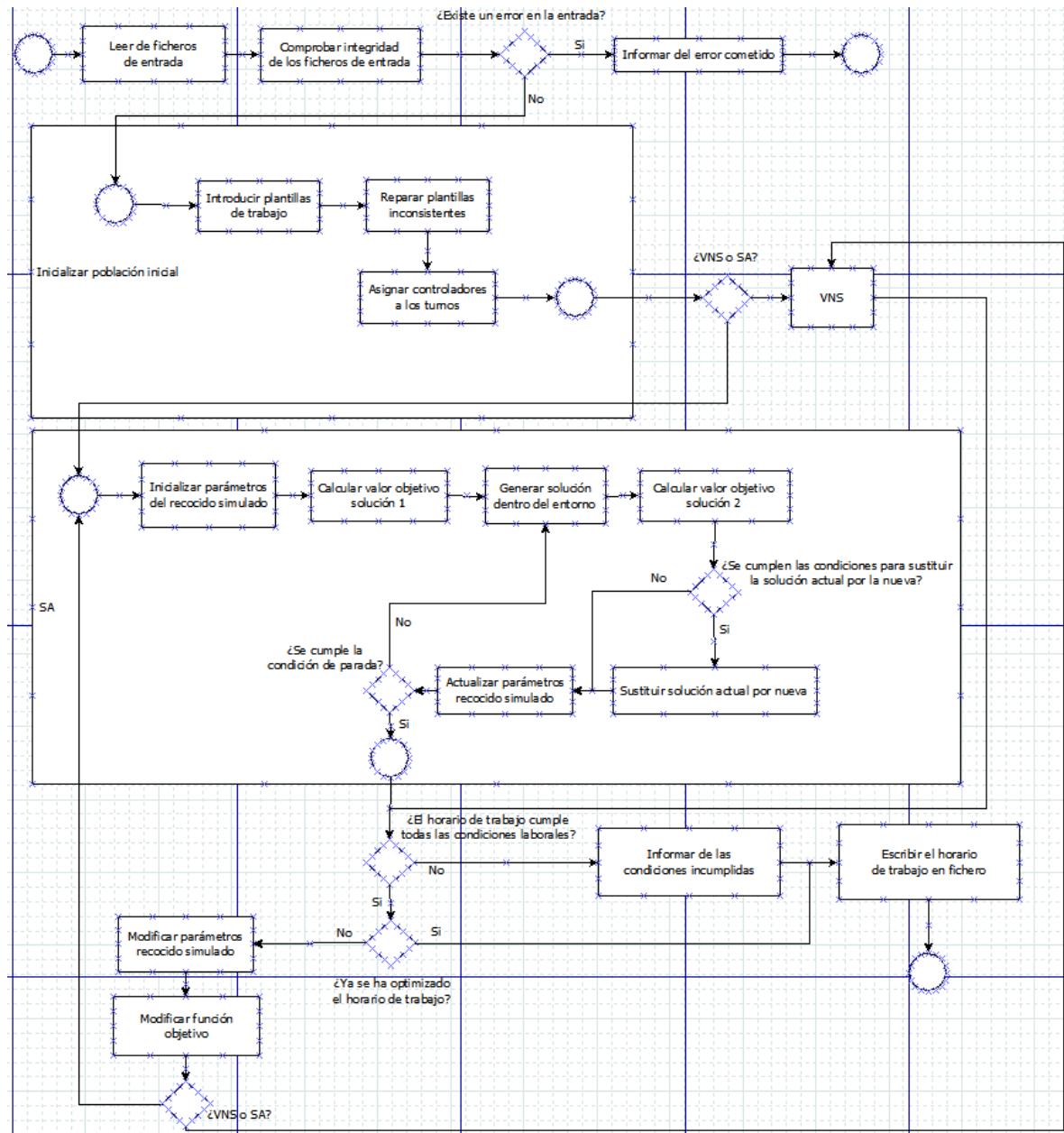


Figura 7.2: Modelo de proceso de negocio

## 7.5. DOCUMENTACIÓN Y MANTENIMIENTO

---

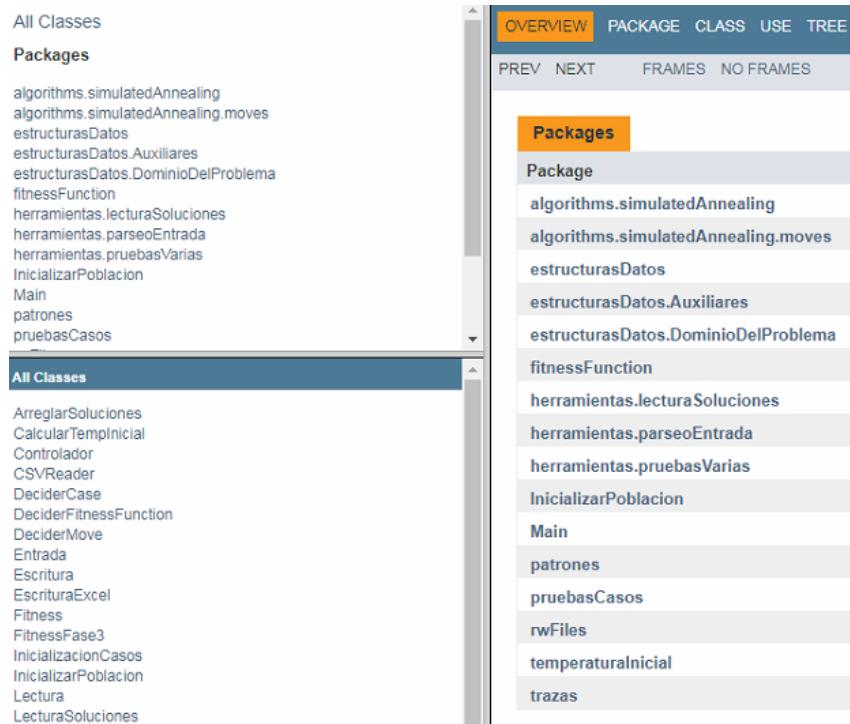


Figura 7.3: Estructura del proyecto

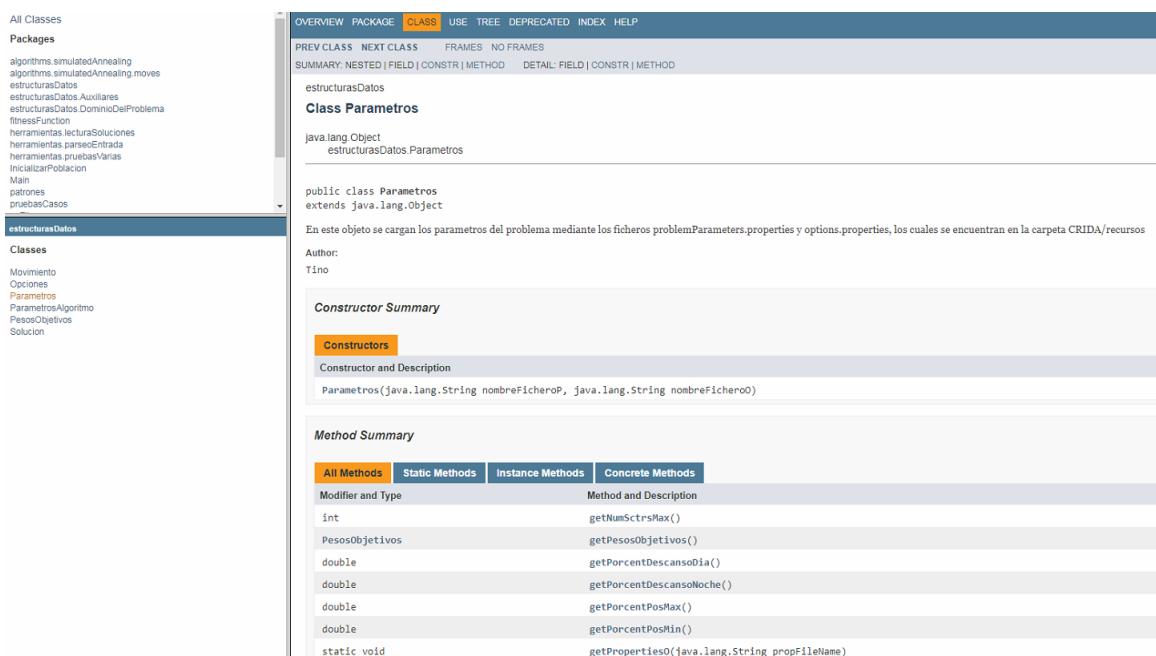


Figura 7.4: Ejemplo de la documentación generada sobre una clase

**Constructor Detail**

**Parametros**

```
public Parametros(java.lang.String nombreFicheroP,
                 java.lang.String nombreFicheroO)
```

**Parameters:**

nombreFicheroP - Ruta del fichero con las propiedades del problema.

nombreFicheroO - Ruta del fichero con las opciones para la resolucion del problema.

Figura 7.5: Ejemplo de la documentación generada para un constructor

**Method Detail**

**switchMoves**

```
public static Solucion switchMoves(Solucion individuo,
                                    int gMax,
                                    int gMin,
                                    double desviacionTipica,
                                    ParametrosAlgoritmo parametrosAlg,
                                    Patrones patrones,
                                    Entrada entrada,
                                    Parametros parametros,
                                    java.util.ArrayList<java.lang.String> iteracion)
```

Metodo utilizado para la eleccion del movimiento.

**Parameters:**

individuo - Solucion

gMax - Slots maximos que puede contener el intervalo.

gMin - Slots minimos que puede conterner el intervalo.

desviacionTipica - Parametro para la elección de controlador (cuando se realiza de forma PseudoAleatoria).

parametrosAlg - Parametros del algoritmo.

patrones - Patrones utilizados para comprobar las restricciones.

entrada - Entrada del problema.

parametros - Parametros del problema.

iteracion - Array con informacion de las trazas del algoritmo.

**Returns:**

Nueva solucion generada por el movimiento a partir de la anterior.

Figura 7.6: Ejemplo de la documentación generada para un método

---

## Capítulo 8

# Conclusiones y líneas futuras de investigación

---

La presente Tesis Doctoral se centra en la utilización de metaheurísticas para la resolución de un problema de optimización multiobjetivo complejo, la asignación de controladores aéreos a puestos de control en sus turnos de trabajo cumpliendo con una serie de restricciones técnicas y laborales y tomando como referencia la sectorización establecida en función del tráfico aéreo.

En un primer acercamiento al problema, se ha considerado una versión del mismo uniobjetivo en la que se intenta minimizar el número de controladores necesarios para cubrir cierta sectorización de un día completo (formado por tres turnos) y con ciertas simplificaciones del problema (se considera un único núcleo y no se tienen en cuenta todas las restricciones laborales de los controladores). Una vez alcanzado el número mínimo de controladores necesario, se intenta buscar soluciones que equilibren la carga de trabajo de los mismos.

Para la resolución de este problema se ha diseñado y desarrollado una metodología de solución que ha supuesto la base para la solución de los posteriores problemas más complejos considerados en esta Tesis Doctoral. En ella, en una primera fase se generan distintas soluciones iniciales en base a una heurística que utiliza plantillas optimizadas. En una segunda fase, se lanza una metaheurística (recocido simulado) multicomienzo, utilizando como soluciones iniciales las soluciones alcanzadas en la primera fase que sean factibles. Para verificar la factibilidad de las soluciones se utilizan expresiones regulares.

Expertos de CRIDA han proporcionado instancias reales con distintas sectorizaciones

establecidas y distintos números de controladores que han servido para validar la metodología desarrollada, alcanzando soluciones con el número mínimo de controladores en tiempos razonables para la fase estratégica, en torno a seis días antes del día de operación.

A continuación, se ha procedido a resolver el problema en forma más compleja y cercana a la realidad. En él, el número de controladores disponible está previamente fijado y se ha adoptado una perspectiva multiobjetivo del problema, en la que se tienen en cuenta condiciones deseables sobre tiempos de trabajo y de descanso y posiciones de los controladores, la homogeneidad en la carga de trabajo de estos, el parecido de la solución a las plantillas utilizadas previamente (para su más fácil interpretación) y el número de cambios en la sala de control.

Adicionalmente, se tiene en cuenta la posibilidad de que haya dos núcleos en el centro de control, se optimizan turnos de trabajo y no días completos, y se tiene en cuenta de forma completa el conjunto de condiciones laborales de los trabajadores.

Para la resolución de este problema se ha desarrollado una metodología de solución basada en tres fases. La primera fase es la misma que en la primera aproximación al problema, consistente en una heurística que utiliza plantillas optimizadas para la creación de distintas soluciones iniciales.

Dado que estas soluciones iniciales pueden ser infactibles, tanto por tener más controladores que los que hay disponibles como por violar ciertas condiciones laborales de los controladores, en la segunda fase se utiliza una metaheurística multicomienzo con el objetivo de alcanzar soluciones factibles. Para ello, la función objetivo tendrá en cuenta el número de controladores utilizados (que se debe reducir hasta el número disponible) y el número de restricciones violadas (que deberá reducirse a 0).

En el momento en que se encuentre una primera solución factible en la fase 2, vamos a la fase 3, en la que asumimos una perspectiva multiobjetivo utilizando de nuevo una metaheurística que se mueve por soluciones factibles hasta alcanzar el óptimo. Tomamos como objetivos las condiciones deseables sobre el trabajo de los controladores, la homogeneidad en la carga de trabajo de estos, el parecido de la solución a las plantillas utilizadas previamente y el número de cambios en la sala de control. Se utiliza la función centroide (con la que se encuentran de acuerdo los expertos de CRIDA) para ponderar los objetivos considerados y transformar el problema en uno uniobjetivo.

---

Tanto en la fase 2 como en la fase 3 se han utilizado y comparado dos metaheurísticas, el *recocido simulado* y la *búsqueda en entornos variables*. En ambos casos, se alcanzan soluciones de muy buena calidad, en opinión de los expertos de CRIDA. Sin embargo, los tiempos de respuesta de la búsqueda en entornos variables es mejor que en el recocido simulado en instancias pequeñas o medianas del problema, mientras que ocurre lo contrario en el caso de instancias de gran dimensión (sectorizaciones con muchos sectores abiertos y que, por tanto, requieren muchos controladores).

Se han realizado estudios con el objetivo de reducir al máximo los tiempos de respuesta, llegándose a la conclusión de que es mejor, en este sentido, la implementación en código de la comprobación de las restricciones del problema que la utilización de expresiones regulares (que se venían utilizando hasta este estudio), y se ha procedido a la paralelización de este proceso, lo que ha supuesto mejoras notables en los tiempos de respuesta.

Expertos de CRIDA nos han proporcionado numerosas instancias del problema en las que se han probado todas las casuísticas a las que nos podríamos enfrentar en situaciones reales (sectorizaciones con poco o mucho tráfico, con pocas o muchas aperturas y cierre de sectores, con más o menos holgura en el número de controladores disponibles...), habiendo alcanzado muy buenas soluciones en tiempos razonables (en opinión de los expertos de CRIDA), teniendo en cuenta que este tipo de problemas se resuelve en la fase estratégica.

Finalmente, se ha considerado una nueva variante del problema asociada a la fase táctica, es decir, que debe ser resuelto el propio día de operación y, por tanto, de forma muy rápida. Se intenta dar respuesta a posibles imprevistos que surjan durante el día de operación y que suponen que se deba recalcular de forma rápida la solución desde el instante en que se produce el incidente hasta el final de turno de trabajo.

El imprevisto puede consistir en que uno o varios controladores se sientan indispuestos y deban abandonar la sala de control, pudiendo o no venir a sustituirles otro/s controlador/es de guardia en instantes posteriores, o que nos llegue tráfico que se ha redirigido desde otro aeropuerto por temas meteorológicos, huelgas o cualquier otro motivo.

En estas situaciones puede ocurrir que la mejor solución no llegue a ser factible, violándose ciertas condiciones laborales de los controladores.

Se ha propuesto una metodología de solución basada en dos fases. En la primera fase se ha utilizado en una nueva heurística que de forma rápida obtiene una solución

inicial para el resto del turno que haga frente al imprevisto. En la segunda fase se utiliza una metaheurística (*recocido simulado o búsqueda en entornos variables*) que tiene como primer objetivo fundamental alcanzar una solución factible (no siempre se tiene por qué conseguir), pero también intenta reducir el número de cambios en sala en el instante justo posterior a la ocurrencia del imprevisto (para evitar el caos que conllevaría un número grande de cambios) y que la estructura de la solución se parezca a una plantilla. Se ha utilizado de nuevo la función centroide para resolver un problema uniobjetivo.

También en esta ocasión, los expertos de CRIDA han proporcionado una batería de instancias con ejemplos reales de imprevistos sucedidos en salas de control, representando una casuística variada (bajas de controladores, llegada de tráfico adicional y combinación de ambas con la posibilidad o no de que se incorporen controladores de guardia).

Los resultados alcanzados en este problema son bastante mejores tanto en lo relativo a los valores alcanzados en las funciones objetivo como en tiempos de ejecución para la búsqueda en entornos variables (5 minutos en el peor de los casos). Los expertos de CRIDA consideran las soluciones alcanzadas como de muy buena calidad y los tiempos de respuesta razonables.

Las herramientas desarrolladas para resolver los problemas considerados han sido integradas en los sistemas informáticos de CRIDA para automatizar la carga de los datos asociados a los escenarios considerados (sectorización, controladores y sus acreditaciones...) y para presentar los resultados alcanzados.

Por todo lo anterior, se puede afirmar que el objetivo fundamental de esta Tesis Doctoral se ha alcanzado. Los resultados de la misma se han difundidos en congresos internacionales y revistas indexadas dentro del JCR, dando lugar a las siguientes publicaciones:

- F. Tello, A. Jiménez-Martín, A. Mateos, P. Lozano. A Comparative Analysis of Simulated Annealing and Variable Neighborhood Search in the ATCo Work-Shift Scheduling Problem, *Mathematics*, 7(7):636, 2019.
- F. Tello, A. Mateos, A. Jiménez-Martín, A. Suárez. The Air Traffic Controller Work-Shift Scheduling Problem in Spain from a Multiobjective Perspective: A Metaheuristic and Regular Expression-Based Approach, *Mathematical Problems in Engineering*, 2018.

- A. Mateos, F. Tello, A. Jiménez-Martín, JA. Fernández. The ATC Work Shift Scheduling Problem Based on Multistart Simulated Annealing and Regular Expressions, *5th International Conference on Control, Decision and Information Technologies (CoDIT)*, 152-157, 2018.
- F. Tello, A. Mateos, A. Jiménez-Martín, JA. Fernández. ATC work shift scheduling using multistart simulated annealing and regular expressions, *Proceedings of the 2017 International Conference on Decision Support Systems Technology*, 169-175, 2017.

En relación con las líneas futuras de investigación asociadas a esta Tesis Doctoral, en primer lugar, todavía se puede intentar mejorar más los tiempos de respuesta de las soluciones propuesta incorporando un mayor nivel de paralelización en algunas de las tareas realizadas, optimizando el código de la implementación o probando con variantes de las metaheurísticas utilizadas (probar con nuevas definiciones de entornos...).

En la versión del problema en fase táctica, en la que se da respuesta a la ocurrencia de un imprevisto, dado que es posible en casos extremos no alcanzar una solución factible, sería de gran utilidad poder obtener las preferencias de los expertos de CRIDA sobre la violación de las distintas restricciones, para que de esta forma se violen aquellas que los expertos consideren menos relevantes.

En la presente Tesis Doctoral se ha trabajado con dos metaheurísticas trayectoriales, el *recocido simulado* y la *búsqueda en entornos variables*. Se propone también como línea futura de investigación la utilización de otras metaheurísticas trayectoriales o poblaciones e hibridaciones de las mismas.

Finalmente, se propone como trabajo futuro continuar con la evaluación de nuevas instancias del problema en sus distintas versiones para asegurar la robustez de las soluciones propuestas antes de su puesta en producción.

---

## CAPÍTULO

---

## Anexo A

# Anexo 1: Expresiones Regulares

---

En este anexo se muestran todas las expresiones regulares utilizadas para comprobar el cumplimiento de todas las restricciones del problema en cada una de las soluciones generadas.

Lista de condiciones del entorno:

1. Una determinada posición podrá ser asignada a un controlador si el controlador está habilitado en el núcleo al que pertenece el sector que le corresponde, o bien en uno de los núcleos a los que pertenece el sector, si este es un sector común, independientemente de la sectorización por la que el sector se encuentre abierto.
  - No utiliza expresiones regulares para su comprobación.
2. A un controlador tipo CON se le podrá asignar una posición cuyo sector sea Ruta.
  - No utiliza expresiones regulares para su comprobación.
3. Porcentaje de tiempo de descanso mínimo en turno diurno (mañana y tarde), incluyendo turnos largos es de 25 %. Mientras que el porcentaje de tiempo de descanso mínimo en el turno de noche es de 33 %.
  - `Pattern.compile("(111)")`
4. Los sectores o agrupaciones de dos sectores que se indique en la entrada del problema, se deberán cubrir con 4 controladores en el turno de noche. Nota: puede existir más de un sector o agrupación de sectores que se deban cubrir con 4 controladores. Esta restricción únicamente se aplica sobre los turnos de noche.

- `Pattern.compile("^(" + sctNc + "111){" + numSlots + "}" + "$", Pattern.CASE_INSENSITIVE)`

donde *sctNc* es una cadena de texto que contiene la lista de sectores nocturnos de la misma agrupación (ej. aaa|abb|aad) y *numSlots* contiene el número de slots totales del turno.

5. En los turnos de mañana, tarde y noche, no es posible un período de trabajo continuo mayor de 2 horas en los que el controlador no realice ningún período de descanso.

- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`
- `Pattern.compile("^((111)*[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)+[a-zA-Z]{3," + tMax + "})(111)*$")`

Solo es necesario que cumpla 1 de las 12 utilizadas, con eso ya sabremos que cumple la restricción. El valor de *tMax* es de 72 ( $3 \times 120/5 = 72$ ).

6. Un controlador solo puede operar en su turno correspondiente, si pertenece al turno largo, en el turno largo y si pertenece al turno corto en el turno corto.

- `Pattern.compile("^.*(111){" + resto + "}" + "$")`
- `Pattern.compile("^((111){" + resto + "}).*$")`

---

donde *resto* es la diferencia entre el turno largo y corto, es decir, lo que obligatoriamente los controladores pertenecientes al turno corto tienen que descansar.

7. En los turnos de mañana, tarde y noche, no puede existir ningún período de dos horas y media en los que un controlador realice un período total de descanso menor de media hora. Es decir, dentro de una ventana de tiempo de dos horas y media un controlador debe tener mínimo 30 minutos de descanso, sin ser necesario que estos se realicen de forma continua.

- No utiliza expresiones regulares para su comprobación.

8. Un controlador no puede cambiar de posición de control de una posición ejecutiva de un determinado sector a una posición ejecutiva de otro sector diferente, sin que exista un descanso entre medias, a no ser que ambos sectores sean afines entre sí.

- No utiliza expresiones regulares para su comprobación.

9. El tiempo mínimo de trabajo continuado. Es decir, el tiempo de trabajo de un controlador entre dos descansos es de 15 minutos.

- `Pattern.compile("^((111)*[a-zA-Z]{\"+tMin+\"},)(111)+[a-zA-Z]{\"+tMin+\"},(111)*$")`

`(111)+[a-zA-Z]{\"+tMin+\"},(111)*$")`

Se utilizan 12 expresiones regulares con la estructura mostrada. Solo es necesario que cumpla 1 de las 12 utilizadas, con eso ya sabremos que cumple la restricción. El valor de *tMin* es de 9 ( $3 * 15/5 = 9$ ).

10. El tiempo mínimo de descanso de un controlador es de 15 minutos.

- `Pattern.compile("^((111)*[a-zA-Z]{3,}(111){\"+dMin+\"},)[a-zA-Z]{3,}((111)*$)")`

Se utilizan 12 expresiones regulares con la estructura mostrada. Solo es necesario que cumpla 1 de las 12 utilizadas, con eso ya sabremos que cumple la restricción. El valor de *dMin* es de 9 ( $3 * 15/5 = 9$ ).

11. El tiempo mínimo en posición de un controlador es de 15 minutos.

- `Pattern.compile("^((111)*(([A-Z]{\"+pMin+",})|([A-Z]{\"+pMin+",})[a-z]{\"+pMin+",})|([A-Z]{\"+pMin+",})|([A-Z]{\"+pMin+",})[a-z]{\"+pMin+",}[A-Z]{\"+pMin+",})|(([a-z]{\"+pMin+",})|([a-z]{\"+pMin+",})[A-Z]{\"+pMin+",})|(([a-z]{\"+pMin+",})[A-Z]{\"+pMin+",}[a-z]{\"+pMin+",})|(([a-z]{\"+pMin+",})[A-Z]{\"+pMin+",}[a-z]{\"+pMin+",}[A-Z]{\"+pMin+",}))|(111)*$")`

Se utilizan 12 expresiones regulares con la estructura mostrada. Solo es necesario que cumpla 1 de las 12 utilizadas, con eso ya sabremos que cumple la restricción. El valor de  $pMin$  es de 9 ( $3 * 15/5 = 9$ ).

12. Número máximo de sectores por los que rota un controlador, teniendo en cuenta las posibles afinidades: 3

- No utiliza expresiones regulares para su comprobación.

13. Todos los controladores disponibles deben tener una línea de la matriz asignada y a su vez todas las líneas de la matriz deben estar asociadas a un controlador.

- No utiliza expresiones regulares para su comprobación.

14. Todos los controladores deberán tener al menos 3 slots de trabajo, no podrán permanecer descansando toda la jornada.

- `Pattern.compile("^((111){\"+nSlot+\"}$$")`

donde,  $nSlot$  es el número total de slots que tiene el turno.

# Bibliografía

---

- [1] E. Aarts, E. H. Aarts, y J. K. Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 1997.
- [2] E. H. Aarts y J. H. Korst. Boltzmann machines for travelling salesman problems. *European Journal of Operational Research*, 39(1): 79–95, 1989.
- [3] D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37(1): 98–113, 1991.
- [4] D. Abramson, M. Krishnamoorthy, y H. Dang. Simulated annealing cooling schedules for the school timetabling problem. *Asia Pacific Journal of Operational Research*, 16: 1–22, 1999.
- [5] R. Álvarez-Valdés, G. Martín, y J. Tamarit. Constructing good solutions for the spanish school timetabling problem. *Journal of the Operational Research Society*, 47(10): 1203–1215, 1996.
- [6] M. Arnvig, B. Beermann, B. Köper, M. Maziul, U. Mellett, C. Niesing, y J. Vogt. Managing shiftwork in european ATM: Literature review. Technical report, Brussels: EUROCONTROL, 2006.
- [7] T. Atan y N. Secomandi. A rollout-based application of the scatter search/path relinking template to the multi-vehicle routing problem with stochastic demands and restocking. *PROS Revenue Management, Inc.*, Houston, TX, 1999.
- [8] S. Bandyopadhyay, S. Saha, U. Maulik, y K. Deb. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE transactions on Evolutionary Computation*, 12(3): 269–283, 2008.

- [9] M. Bellmore y G. L. Nemhauser. The traveling salesman problem: a survey. *Operations Research*, 16(3): 538–558, 1968.
- [10] W. Ben-Ameur. Computing the initial temperature of simulated annealing. *Computational Optimization and Applications*, 29(3): 369–385, 2004.
- [11] I. Boussaïd, J. Lepagnot, y P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237: 82–117, 2013.
- [12] E. K. Burke, G. Kendall, M. Misir, y E. Özcan. Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, 196(1): 73–90, 2012.
- [13] E. K. Burke, G. Kendall, y E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6): 451–470, 2003.
- [14] J. Butler y D. L. Olson. Comparison of centroid and simulation approaches for selection sensitivity analysis. *Journal of Multi-Criteria Decision Analysis*, 8(3): 146–161, 1999.
- [15] M. W. Carter y G. Laporte. Recent developments in practical course timetabling. En: *International Conference on the Practice and Theory of Automated Timetabling*, 3–19. Springer, 1997.
- [16] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1): 41–51, 1985.
- [17] I. A. Chaudhry y A. A. Khan. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3): 551–591, 2016.
- [18] B. Cheang, H. Li, A. Lim, y B. Rodrigues. Nurse rostering problems - a bibliographic survey. *European Journal of Operational Research*, 151(3): 447–460, 2003.
- [19] H. Cohn y M. Fielding. Simulated annealing: searching for an optimal temperature schedule. *SIAM Journal on Optimization*, 9(3): 779–802, 1999.

- [20] A. Colorni, M. Dorigo, y V. Maniezzo. Genetic algorithms: A new approach to the timetable problem. *Combinatorial Optimization*, 85: 235–239, 1992.
- [21] A. Colorni, M. Dorigo, V. Maniezzo, y M. Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research*, 34: 39–53, 1994.
- [22] R. Conniss, T. Curtis, y S. Petrovic. Scheduling air traffic controllers. En: *10th International Conference of the Practice and Theory of Automated Timetabling*, 2014.
- [23] D. Costa. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76(1): 98–110, 1994.
- [24] M. Creutz. Microcanonical monte carlo simulation. *Physical Review Letters*, 50(19): 1411, 1983.
- [25] M. A. Cruz-Chávez, M. G. Martínez-Rangel, y M. H. Cruz-Rosales. Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. *International Transactions in Operational Research*, 24(5): 1119–1137, 2017.
- [26] I. Das y J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1): 63–69, 1997.
- [27] L. Davis. Job shop scheduling with genetic algorithms. En: *Proceedings of an International Conference on Genetic Algorithms and their Applications*, 140, 136–140, 1985.
- [28] M. Dorigo, V. Maniezzo, y A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Management, and Cybernetics (Part B)*, 26(1): 29–41, 1996.
- [29] M. Dorigo y T. Stützle. Ant colony optimization: overview and recent advances. En: *Handbook of metaheuristics*, 311–351. Springer: Cham, 2019.

- [30] G. Dueck y T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1): 161–175, 1990.
- [31] R. W. Eglese. Simulated annealing: a tool for operational research. *European Journal of Operational Research*, 46(3): 271–281, 1990.
- [32] M. Eley. Ant algorithms for the exam timetabling problem. En: *International Conference on the Practice and Theory of Automated Timetabling*, 364–382. Springer, 2006.
- [33] Enaire. Enaire servicios. <https://www.enaire.es/servicios>.
- [34] W. Erben y J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. En: *International Conference on the Practice and Theory of Automated Timetabling*, 198–211. Springer, 1995.
- [35] G. Erétéo, F. Gandon, O. Corby, y M. Buffa. Semantic social network analysis. *arXiv preprint arXiv:0904.3701*, 2009.
- [36] A. T. Ernst, H. Jiang, M. Krishnamoorthy, y D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1): 3–27, 2004.
- [37] T. A. Feo y M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operation Research Letters*, 8(2): 67–71, 1989.
- [38] G. H. Fonseca y H. G. Santos. Variable neighborhood search based algorithms for high school timetabling. *Computers & Operations Research*, 52: 203–208, 2014.
- [39] G. H. Fonseca, H. G. Santos, y E. G. Carrano. Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, 74: 108–117, 2016.
- [40] J. E. Friedl. *Mastering regular expressions*. O'Reilly Media, Inc., 2006.
- [41] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1): 156–166, 1977.

- [42] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5): 533–549, 1986.
- [43] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3): 190–206, 1989.
- [44] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2(1): 4–32, 1990.
- [45] F. Glover. A template for scatter search and path relinking. En: *European Conference on Artificial Evolution*, 1–51. Springer, 1997.
- [46] F. Glover y M. Laguna. Tabu search. En: *Handbook of combinatorial optimization*, 2093–2229. Springer, 1998.
- [47] F. Glover, M. Laguna, y R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3): 653–684, 2000.
- [48] A. Gunawan, K. M. Ng, y K.-L. Poh. A mathematical programming model for a timetabling problem. En: *International Conference on Scientific Computing*, 42–47, 2006.
- [49] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2): 311–329, 1988.
- [50] P. Hansen y N. Mladenović. An introduction to variable neighborhood search. En: *Meta-heuristics*, 433–458. Springer, 1999.
- [51] P. Hansen y N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3): 449–467, 2001.
- [52] P. Hansen, N. Mladenovic, y J. A. M. Pérez. Variable neighbourhood search. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 19: 77–92, 2003.
- [53] P. Hansen, N. Mladenović, y D. Pérez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4): 335–350, 2001.
- [54] J. Heimdal, S. Strand, G. Skraaning, U. Mellet, y J. Vogt. Shiftwork practices study - atm and related industries. Technical report, Brussels: EUROCONTROL, 2006.

- [55] Y. T. Herer, M. Tzur, y E. Yücesan. The multilocation transshipment problem. *Institute of Industrial and Engineers Transactions*, 38(3): 185–200, 2006.
- [56] A. Hertz. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54(1): 39–47, 1991.
- [57] A. Hertz y M. Mittaz. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35(4): 425–434, 2001.
- [58] J. H. Holland. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [59] B. Hoppe y É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1): 36–62, 2000.
- [60] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11): 29–57, 1993.
- [61] D. S. Johnson, C. R. Aragon, L. A. McGeoch, y C. Schevon. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations Research*, 37(6): 865–892, 1989.
- [62] D. S. Johnson, C. R. Aragon, L. A. McGeoch, y C. Schevon. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3): 378–406, 1991.
- [63] I. Kacem, S. Hammadi, y P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Management, and Cybernetics (Part C)*, 32(1): 1–13, 2002.
- [64] S. Karakatič y V. Podgorelec. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27: 519–532, 2015.
- [65] J. P. Kelly, B. Rangaswamy, y J. Xu. A scatter-search-based learning algorithm for neural network training. *Journal of Heuristics*, 2(2): 129–146, 1996.

- [66] J. Kennedy y R. Eberhart. Particle swarm optimization. En: *Proceedings of International Conference on Neural Networks*, 4, 1942–1948, 1995.
- [67] S. Kirkpatrick, C. D. Gelatt, y M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.
- [68] E. L. Lawler y D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4): 699–719, 1966.
- [69] R. Linn y W. Zhang. Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering*, 37(1-2): 57–61, 1999.
- [70] M. Locatelli. Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, 104(1): 121–133, 2000.
- [71] W. Mahdi, S. A. Medjahed, y M. Ouali. Performance analysis of simulated annealing cooling schedules in the context of dense image matching. *Computación y Sistemas*, 21(3): 493–501, 2017.
- [72] F. Marass y C. Upton. Sequence searcher: A java tool to perform regular expression and fuzzy searches of multiple dna and protein sequences. *BMC Research Notes*, 2(1): 14, 2009.
- [73] J. F. D. Martín y J. M. R. Sierra. A comparison of cooling schedules for simulated annealing. En: *Encyclopedia of Artificial Intelligence*, 344–352. IGI Global, 2009.
- [74] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, y E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6): 1087–1092, 1953.
- [75] N. Mladenović y P. Hansen. Variable neighborhood search. *Computers & operations research*, 24(11): 1097–1100, 1997.
- [76] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826, 1989.

- [77] T. Murata, H. Ishibuchi, y H. Tanaka. Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering*, 30(4): 1061–1071, 1996.
- [78] D. Nam y C. H. Park. Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2): 87–97, 2000.
- [79] F. Neri y C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2: 1–14, 2012.
- [80] S. Niarchakou y M. Cech. Atfcm operations manual: Network manager. Technical report, Brussels: EUROCONTROL, 2019.
- [81] Y. Nourani y B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41): 8373–8385, 1998.
- [82] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, y T. J. Chua. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, 181(12): 2455–2468, 2011.
- [83] M. Pennacchiotti y A.-M. Popescu. A machine learning approach to twitter user classification. En: *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.
- [84] N. Pérez-Díaz, D. Ruano-Ordas, F. Fdez-Riverola, y J. R. Méndez. Wirebrush4spam: a novel framework for improving efficiency on spam filtering services. *Software: Practice and Experience*, 43(11): 1299–1318, 2013.
- [85] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, y S. Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1): 55–89, 2009.
- [86] M. G. Resende. Greedy randomized adaptive search procedures: Greedy randomized adaptive search procedures gpasp. *Encyclopedia of Optimization*, 1460–1469, 2009.
- [87] I. Rivin, I. Vardi, y P. Zimmermann. The n-queens problem. *The American Mathematical Monthly*, 101(7): 629–639, 1994.

- [88] R. Ruiz y J. A. Vázquez-Rodríguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1): 1–18, 2010.
- [89] A. Schaefer. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2): 87–127, 1999.
- [90] P. Serafini. Simulated annealing for multi objective optimization problems. En: *Multiple criteria decision making*, 283–292. Springer, 1994.
- [91] P. Siarry, G. Berthiau, F. Durbin, y J. Haussy. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. *ACM Transactions on Mathematical Software*, 23(2): 209–228, 1997.
- [92] C. Siefkes, F. Assis, S. Chhabra, y W. S. Yerazunis. Combining winnow and orthogonal sparse bigrams for incremental spam filtering. En: *European Conference on Principles of Data Mining and Knowledge Discovery*, 410–421. Springer, 2004.
- [93] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3): 233–272, 1999.
- [94] M. J. F. Souza, N. Maculan, y L. S. Ochi. A grasp-tabu search algorithm for solving school timetabling problems. En: *Metaheuristics: Computer decision-making*, 659–672. Springer, 2003.
- [95] M. Stojadinović. Air traffic controller shift scheduling by reduction to csp, sat and sat-related problems. En: *International Conference on Principles and Practice of Constraint Programming*, 886–902. Springer, 2014.
- [96] M. Stojadinović. Hybrid of hill climbing and sat solving for air traffic controller shift scheduling. *Journal of Information Technology and Applications*, 10(2), 2016.
- [97] E. H. Team. Atm manpower planning in practice: Introduction to a qualitative and quantitative staffing methodology. Technical report, Brussels: EUROCONTROL, 1998.

- [98] F. Tello, A. Jiménez-Martín, A. Mateos, y P. Lozano. A comparative analysis of simulated annealing and variable neighborhood search in the atco work-shift scheduling problem. *Mathematics*, 7(7): 636, 2019.
- [99] F. Tello, A. Mateos, A. Jiménez-Martín, y J. F. de Pozo. Atc work shift scheduling using multistart simulated annealing and regular expressions. En: *Proceedings of the 2017 International Conference on Decision Support Systems Technology*, 169–175, 2017.
- [100] F. Tello, A. Mateos, A. Jiménez-Martín, y A. Suárez. The air traffic controller work-shift scheduling problem in spain from a multiobjective perspective: A metaheuristic and regular expression-based approach. *Mathematical Problems in Engineering*, vol. 2018: 15, 2018.
- [101] P. J. Van Laarhoven, E. H. Aarts, y J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1): 113–125, 1992.
- [102] S. R. White. Concepts of scale in simulated annealing. En: *AIP Conference Proceedings*, 122, 261–270. AIP, 1984.
- [103] A. Wren. Scheduling, timetabling and rostering - a special relationship? En: *International Conference on the Practice and Theory of Automated Timetabling*, 46–75. Springer, 1995.
- [104] W. Xia y Z. Wu. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 48(2): 409–425, 2005.
- [105] Y. Zhang, S. Wang, y G. Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, vol. 2015, 2015.