Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema de Decisión Inteligente. Teoría de Juegos. Diseño, aplicación y evaluación.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Ferrer Sánchez, Sergio

Tutor: Barber Sanchís, Federico

Cotutor: Salido Gregorio, Miguel Ángel

2015-2016

Agradecer a Federico y Miguel, tutor y cotutor respectivamente del presente trabajo, su disponibilidad e implicación en el mismo siempre que la he necesitado.

Resumen

En este proyecto se plantea el diseño, desarrollo e implementación de un sistema capaz de resolver el ampliamente conocido *problema de asignación de turnos* de forma genérica, adaptable y en tiempos computacionalmente tratables con el fin de poder aplicarlo en distintos entornos y ajustándose a las necesidades concretas de cada posible escenario. El sistema desarrollado permite considerar diversos tipos de turnos, restricciones generales y específicas de asignación, y diversos criterios de optimización.

Dado el alto coste computacional y las complejidades espacial y temporal a las que nos enfrentamos en la resolución de este problema característico de entornos empresariales con amplias plantillas, se plantea una solución mediante técnicas de inteligencia artificial. El método computacional desarrollado, basado en técnicas heurísticas y metaheurísticas, permite converger y obtener soluciones razonablemente optimizadas en un tiempo aceptable sin necesidad de explorar todo el espacio de búsqueda.

Proporcionaremos una solución conceptual y teórica al problema, que se acompaña de la implementación del sistema final en Python-2.7.11 junto con la interfaz gráfica de usuario, creada con la tecnología JavaFX-2.0.2, ligada a su correspondiente base de datos mediante SQLite-3.13.0, tratando así de ofrecer una forma sencilla e intuitiva de parametrizar el problema, dotando de esta forma de una alta adaptabilidad al sistema resultante.

Por último, analizaremos los resultados obtenidos sobre casos reales de entornos que requieren una solución al problema de la asignación de turnos, lo que permite contrastar la viabilidad del sistema propuesto como solución a los problemas planteados.

Palabras clave: asignación de turnos, Python, JavaFX, SQLite, búsqueda guiada.

Abstract

In this project, we propose the design and implementation of a system able to solve the well known *staff rostering problem* in a generic and adaptable way and fitting computationally tractable times, in order to being able to adapt it in different environments and matching the different needs of any entity that use this system. The developed system is able to work with different types of turns, general and specific constraints and multiple optimization criteria.

Due to the high computational costs and to the spatial and time complexity that we have to deal with in the resolution of this distinctive problem of business environments with huge amount of staff, we expose a solution based on Artificial Intelligence techniques, specifically, guided searches and heuristics that converge in a reasonable time without exploring the complete search space.

We will offer a conceptual and theoretical solution, accompanied with the implementation of the final system in Python-2.7.11 and the graphic user interface created using JavaFX-2.0.2 linked with its appropriate data base through SQLite-3.13.0, trying to offer a simple and intuitive way of parameterize the problem, providing high adaptability to the final system.

Finally, we will analyze the results obtained on real cases of environments that require solutions to the staff rostering problem to study the feasibility of the proposed solution to these conflicts.

Keywords: staff rostering problem, Python, JavaFX, SQLite, guided search.

Tabla de contenidos

Índice de tablas

Índice de imágenes

1. Introducción

A lo largo de este documento se presenta el Trabajo Final de Grado desarrollado con la finalidad de alcanzar soluciones al ampliamente conocido problema de asignación de turnos para el personal de empresas en puestos de trabajo, mediante técnicas heurísticas y metaheurísticas de búsqueda guiada, tratando de resolver el problema de forma genérica con la finalidad de que el sistema implementado pueda adaptarse a una amplia gama de entornos distintos con las mismas tipologías de restricciones.

1.1 Interés empresarial

La resolución de este tipo de problemas se ha tornado de vital importancia en los últimos años debido al crecimiento masivo de demanda de algunos sectores específicos -entre los que destaca el sector tecnológico- que van acompañados de un aumento del número de integrantes de la plantilla de trabajadores con el fin de satisfacer esta demanda. Los gastos en salarios de estas grandes plantillas son el principal foco de inversión económica de las grandes empresas, siendo por lo tanto de crucial importancia para ellas la optimización de estos recursos, pudiendo ahorrase cantidades ingentes de capital si se logra una asignación óptima o cercana a ésta.

A pesar de ello, en el mercado actual existe un vacío tecnológico con respecto a la resolución de este problema, no existiendo ningún software aceptado de forma generalizada para este fin, debido a que en cada entorno donde se encuentra este problema presenta unas distintas restricciones específicas y no generalizables al resto de entornos, exigiendo de esta forma una solución a medida. Con este trabajo se logra extraer las mínimas características comunes a una gran parte de los entornos y generar un sistema capaz de adaptarse a todas ellas e implementando varias restricciones específicas de algunos entornos en particular.

El problema se enmarca dentro de la optimización combinatoria con un altísimo espacio de búsqueda y con una complejidad temporal que aumenta de forma exponencial con el número de trabajadores y el horizonte de asignación. Debido a las enormes plantillas que presentan los entornos en los que este problema es aplicable y al aumento temporal que esto supone, se hace muy difícil encontrar soluciones al problema y mucho más encontrar soluciones óptimas.

Por lo general, las empresas dan a este problema una solución manual, siendo el personal encargado de la planificación el responsable de hacer cuadrar las necesidades de demanda de recursos humanos con los trabajadores disponibles. Este método manual, aunque eficaz, no es eficiente ni garantiza soluciones óptimas al problema, ni tan siquiera soluciones cercanas a ella.

1.2 Interés científico y práctico del problema

Además del interés empresarial y económico que existe en torno a este problema, también se ha desarrollado un interés científico y práctico al respecto debido a la alta complejidad temporal que exige un método computacional que lo resuelva, animando al estudio, investigación e implementación de técnicas que permitan la exploración de espacios combinatorios de tamaños tan descomunalmente grandes como el que presenta este problema.

Tanto es así, que incluso se realizan competiciones que consisten en buscar soluciones a problemas de estas mismas características. Un ejemplo concreto es *The Nurse Problem,* un problema de asignación de turnos que trata de dar solución a la planificación de un hipotético entorno de enfermeras que deben cumplir una serie de restricciones pero cubriendo una demanda de turnos requeridos. En este contexto, nacen diversas competiciones que consisten en buscar la mejor solución al problema dado. En concreto, la *Nurse Rostering Competition,* celebrada entre el uno de mayo y el veinte de junio de 2010, en la que el departamento de electrónica e ingeniería informática de la *University of Patras,* Grecia, consiguió los mejores resultados hasta la fecha, es un buen ejemplo del interés científico que suscita este problema en particular.

Esta competición ha realizado su segunda edición recientemente desde octubre del 2014 hasta agosto del 2015, donde se consiguieron mejorar los resultados obtenidos en 2010 por parte de la *Martin Luther University Halle-Wittenber*, Alemania.

El interés práctico de lograr formas eficientes de explorar grandes espacios de búsqueda también ha incentivado el nacimiento de varias herramientas capaces de realizar búsquedas eficaces en este tipo de problemas que presentan una combinatoria tan extensa, como por ejemplo:

- **BOLD WorkPlanner:** Es una herramienta software para la planificación de plantillas de trabajo cuyo motor ha sido desarrollado por la empresa española GPS. Capaz de dar soluciones en segundos a problemas de asignaciones basadas en la relación demanda/recursos disponibles.

- **OptaPlanner:** Es un motor de planificación que ofrece facilidades de integración en otros sistemas. Aunque trae casos de usos prediseñados, permite adaptarse a algunos entornos que tratan de resolver problemas similares al nuestro con respecto a gestión de recursos para satisfacer demanda por parte de los clientes.

- **OptQuest**: Aplicación desarrollada por la empresa Optek Systems Inc., de Estados Unidos. Este sistema implementa técnicas muy avanzadas de búsqueda y está orientado a la resolución de problemas que presentan una alta complejidad. Además, es adaptable a distintas plataformas y lenguajes de programación.

- **Workforce Schedule:** Herramienta software desarrollada en exclusiva para la planificación de la platilla de trabajo de la empresa estadounidense Kronos. Es capaz de realizar la asignación a la plantilla de trabajadores respetando la normativa interna de la empresa así como la legislación vigente.

En resumen, como se puede apreciar, el problema de la asignación de turnos presenta un gran interés tanto por el carácter económico que supone a

las empresas como por el interés científico que presenta el desarrollo e investigación de técnicas capaces de encontrar soluciones en espacios de búsqueda grandes.

1.3 Objetivos

A continuación se listan los objetivos de este proyecto que pretenden representar el estado final que se desea alcanzar con la implementación de este sistema y tratando de ser una fiel y realista perspectiva del entorno en el que se desarrolla el trabajo, siendo conscientes de su complejidad tanto computacional como temporal:

1. Lograr que el sistema final sea capaz de resolver problemas de entornos reales empresariales con al menos un 90% de tasa de optimalidad, midiéndola en términos de: “demanda cubierta por la solución dada”/ “demanda total”.

2. Conseguir que este porcentaje de acierto se pueda alcanzar en tiempos inferiores a los treinta minutos de cómputo, incluyendo todos los posibles postprocesos y mejoras a la solución ofrecida, pues este tiempo de cómputo es compatible con el tiempo de proceso habitualmente requerido en los escenarios de aplicación.

3. Proporcionar una tecnología generalizable para la resolución de este tipo de problemas con un alto grado de adaptabilidad a diferentes entornos y sujeta a mejoras fácilmente implementables para cubrir necesidades específicas de cada entorno.

4. Crear una interfaz gráfica de usuario que acompañe al sistema y que proporcione una forma sencilla e intuitiva de parametrizar y adaptar la aplicación a los entornos requeridos.

5. Implementar una base de datos capaz de soportar el almacenamiento de los distintos problemas que el usuario quiera introducir en el sistema mediante la interfaz e integrarla con ésta, de forma que los problemas introducidos se almacenen de forma persistente, pudiendo volver a ejecutarlos sin tener la necesidad de volver a introducirlos.

2. Caracterización del problema

Antes de abordar la resolución al problema de asignación de turnos, conviene detenerse a clarificar en qué consiste exactamente el problema y qué pretende conseguir. En este apartado estudiaremos a fondo el problema a afrontar, las características que tiene y cuáles de ellas son generalizables y cuáles no. Además, justificaremos su coste computacional y su complejidad temporal.

2.1 Elementos del problema

La idea general del problema de asignación de turnos ronda en torno a cubrir una demanda de puestos de trabajo con una plantilla de trabajadores finita (o de forma genérica, un conjunto de recursos cualquiera, no necesariamente humanos) de forma óptima para así no generar la necesidad de pagar por más recursos para cubrir la demanda que la entidad debe satisfacer. Básicamente, consiste en organizar un grupo de trabajadores para que cumplan todos los puestos necesarios durante un periodo de tiempo, a la vez que se cumplen las restricciones legales (o empresariales) con respecto a lo que un trabajador puede o no hacer.

Aunque la visión general del problema pueda parecer clara, cabe caracterizar en profundidad el problema para poder distinguir qué elementos comunes tienen todos los problemas de este tipo, por lo que utilizaremos una definición más formal del mismo.

Así pues, una instancia de problema de asignación de turnos, que llamaremos P, es un conjunto formado por 5 elementos que desarrollaremos a continuación:

$$P = \{\mathrm{T}, \mathrm{N}, \mathrm{E}, \mathrm{D}, \mathrm{R}\}$$

- T: Es un número entero que representa el número de trabajadores de los que se dispone en la plantilla.

- N: Es el número de días en los que se quiere realizar la asignación, también llamado horizonte de asignación.

- E: Es un conjunto de letras o palabras que llamaremos etiquetas. Estas etiquetas representan los turnos que un trabajador puede hacer y que se le asignarán para que durante un día se dedique exclusivamente al turno asignado. Todos los problemas tienen un turno común, el descanso, que representaremos con la etiqueta "-". La etiqueta "-" hace referencia a los días en que el trabajador libra. Por ejemplo, en un entorno empresarial 7/24 donde los trabajadores pudieran hacer turnos de mañana, tarde o noche, el conjunto E para este problema podría ser E={M,T,N,-} o incluso E={Mañana, Tarde, Noche,-}. La representación del los turnos recae en la responsabilidad del usuario, siendo indiferente cuáles y cuántas etiquetas utilice.

- D : Es una matriz de tamaño |E| x N en la que, en cada posición *(e, d)* de ésta, se indica cuánta demanda existe del turno representado por la etiqueta *e*, en el día *d* correspondiente. Con $e \in$ E, d<=N.

- R: Son un conjunto de restricciones (o condiciones) que la solución al problema P debe cumplir y que generalmente se describen en lenguaje natural. Dentro de R se distinguen 2 grupos de restricciones que suelen tener la mayoría de instancias de problemas de asignación de turnos:

    - **Restricciones generales**: Son restricciones que tienen todos los problemas de asignación de turnos que abordaremos, y que por lo

general, suelen tener todos los problemas de asignación de turnos. Estas restricciones son:

- Longitudes de tamaño de secuencia: Indican cuál es el tamaño máximo y mínimo de una secuencia de turnos iguales que puede hacer un trabajador. Ejemplo: "Un trabajador no puede hacer menos de dos noches seguidas ni más de cinco noches seguidas". También suelen representarse en forma de tabla.

- Turnos totales máximos: Es un número entero que indica cuántos turnos como máximo puede hacer un trabajador dentro del horizonte de asignación, y generalmente van ligadas a condiciones legales. Ejemplo: "Un trabajador no puede hacer más de 225 turnos en un año".

- Descansos obligatorios entre secuencias: Indican cuántos días de descanso debe haber como mínimo entre dos secuencias de turnos de un trabajador. Por ejemplo "Es obligatorio que después de un turno de guardia se realice, al menos, un turno de descanso antes de realizar otro tipo de turno". Estas restricciones también suelen representarse en forma de tabla

- **Restricciones específicas**: Son condiciones concretas de cada entorno, que a diferencia de las anteriores, no son generalizables a la mayoría de contextos. Cada entidad tiene las suyas propias y hay tantas como entidades pueda haber. Este tipo de restricciones supone el mayor reto a la hora de crear un software generalizable para el problema en cuestión, dada la alta diversidad que existe de estas condiciones. Algunas restricciones específicas muy extendidas en distintos entornos son:

  - Un fin de semana libre al mes: Esta restricción, si está en uso, exige que los trabajadores tengan al menos un fin de semana libre al mes.

- Un solo fin de semana de trabajo: Esta restricción, si está en uso, exige que los trabajadores sólo puedan trabajar un fin de semana al mes.

- Prohibido empezar secuencia en: Este tipo de restricciones prohíben que se empiece a trabajar los días indicados por la restricción. Ejemplo: "Un trabajador no puede empezar una secuencia de trabajo en Domingo. En su lugar, debe empezar el Sábado y extenderla todo el fin de semana o empezar el Lunes".

- Prohibición de turnos a trabajadores concretos: En algunos entornos, existen unos trabajadores que no realizan algunos turnos específicos. Esta restricción pretende modelar esta situación en la que se añade de forma específica la prohibición de que el trabajador indicado realice el turno señalado.

- Prohibición de fechas a trabajadores concretos: De forma similar a la estricción anterior, esta restricción expresa la imposibilidad de que un trabajador especificado trabaje en los días indicados, independientemente del turno que realice.

- Prohibición de festivos o fines de semana a trabajadores concretos: Como extensión de la restricción anterior, puede darse la situación en la que se desee que algún trabajador no se le asignen turnos en festivos o fines de semana. Esta restricción permite marcar un trabajador si se desea que cumpla esta condición.

Además de estas restricciones, también encontramos una restricción más que está en todos los problemas de asignación de turnos: "Toda la demanda debe quedar cubierta". Evidentemente, esta restricción representa la finalidad del problema y no se considera como una restricción concreta de algunos entornos, si no como una restricción intrínseca al problema. Aunque esta restricción no siempre es posible cumplirla en tiempos computacionales

aceptables, la finalidad es obtener un buen porcentaje (90%, como es especificó en los objetivos. Punto 1.3) de cubrimiento de demanda.

Dada esta definición del problema y esta taxonomía de las restricciones, cabe destacar que con el fin de que el sistema sea lo más adaptable posible para abarcar un amplio espectro de entornos, el proyecto resultante cumplirá todas las restricciones generales, pues son comunes a todos los problemas. Sin embargo, no estará adaptado a todas las restricciones especiales, pues presentan un alto grado de variabilidad, lo que imposibilita un software genérico en ese aspecto. Pese a ello, el sistema final será fácilmente adaptable a todas estas restricciones especiales permitiendo así ser un buen software genérico utilizable en muchos entornos distintos con un bajo esfuerzo de adaptabilidad.

2.2 Función resolutiva y representación de la solución

Dado una instancia P del problema que cumple la definición anterior, definiremos una función F(P) capaz de resolver P de la siguiente forma:

$$F(\{T, N, E, D, R\}) = S$$

- S: Es una matriz de tamaño T x N, en la que en cada posición *(t, n)* de la misma se incluye une etiqueta *e* contenida en E. La semántica de esta representación se entiende como:"El trabajador t realiza el turno representado por la etiqueta e el día n". Con $t <= T$, $n <= N$, $e \in E$.

Las soluciones que la función F da como salida son siempre válidas. Diremos que una solución es válida (o factible) cuando no viola ninguna restricción contenida en R. Además, la función pretende optimizar el porcentaje de demanda sin cubrir y los recursos humanos necesarios para hacerlo, sin embargo, debido a las limitaciones temporales y computacionales, no se asegura la solución óptima, aunque empíricamente los resultados muestran soluciones cercanas a la óptima y eficientes en cuanto a recursos consumidos.

Por último, nótese que no se precisa de una función que evalúe la factibilidad (validez) de una solución, pues por la definición dada a la función F, las soluciones siembre serán válidas.

2.3 Criterios de optimalidad

Como se indica en el punto anterior, una solución es factible (válida) si respeta las restricciones generales del problema. Sin embargo, que una solución sea factible no implica que sea la óptima, ni si quiera que sea una buena aproximación a ella. De la misma forma, dos o más soluciones pueden ser factibles pero radicalmente distintas entre ellas. Esto nos obliga a definir criterios de optimalidad, que expresan las condiciones deseables en una solución factible para poder discernir, entre todas las que lo son, cuál de ellas debe ser elegida como "mejor".

Estos criterios de optimalidad, al igual que las restricciones específicas, son distintos en cada entorno, pues en función de éste, pueden decidirse distintos motivos por los que dar preferencia a unas situaciones frente a otras. A continuación se exponen algunos criterios de optimalidad que pueden ayudar en la tarea de evaluación de soluciones factibles:

- **Nivel de cubrimiento de demanda:** Puede considerarse que una solución es más óptima cuanto más alto es el nivel de cubrimiento de demanda. Así pues, la aplicación de este criterio sugiere que se prefieren soluciones que cubran más demanda.

- **Equilibrios en la asignación:** Frente a 2 soluciones factibles, puede preferirse aquella que ofrezca un mejor equilibrio entre las asignaciones de trabajadores para no caer en situaciones desbalanceadas que pudieran molestar al personal. Por ejemplo: "Un trabajador realiza quince turnos de guardia mientras que otro realiza solamente uno". Puede preferirse que el equilibrio se aplique a distintos criterios. Algunos de ellos son:

- **Equilibrio entre el total de turnos realizados por cada trabajador:** Este equilibrio trata de evitar que la solución otorgada tenga situaciones en las que un trabajador realice muchos turnos anuales mientras que otro realiza una cantidad notablemente inferior. Así pues, la aplicación de este criterio dará preferencia a las soluciones cuyos trabajadores realicen una cantidad de turnos similar.

- **Equilibrio entre los diferentes tipos de turnos:** Para evitar que ciertos turnos, generalmente aquellos que son más rechazados por el personal de plantilla (por ejemplo, los turnos de guardia, o de noches), se acumulen sobre un pequeño conjunto de trabajadores, se puede preferir aquellas soluciones que representen unas distribución homogénea de todos los turnos disponibles.

- **Equilibrios entre los fines de semana o festivos trabajados:** Pueden no desearse situaciones en las que un pequeño conjunto de trabajadores realicen turnos de trabajo un alto porcentaje de los festivos o fines de semana anuales. Así pues, este criterio trata de dar preferencia a aquellas soluciones que ofrecen una mejor distribución de los fines de semana y festivos que realiza cada trabajador.

Por lo general, la aplicación de estos criterios, o de cualquier otro que decida la entidad en cuestión, no se aplican de forma aislada y única, si no que por lo general se realiza una evaluación de la solución en función de varios criterios simultáneamente y de forma ponderada entre ellos. De esta forma, se precisa de una función, que llamaremos “FO”, que evalúe el nivel de cumplimiento de los criterios de optimalidad que tiene una solución.

Si se desean aplicar N criterios de optimalidad y realizar una combinación lineal de ellos en función de sus coeficientes de ponderación, una posible definición de FO sería:

$$FO(S) = \sum_{i=0}^{N-1} coeficienteDePonderación_i * evaluaciónDelCriterio_i$$

Cumpliendo que:

$$\sum_{i=0}^{N-1} coeficienteDePonderación_i = 1$$

Dada una solución S a un problema P, diremos que S ofrece un nivel C de cumplimiento de criterios de optimalidad si:

$$FO(S) = C$$

Nuestro objetivo será minimizar el valor C para la solución obtenida, es decir, seleccionaremos como "mejor" aquella solución S con menor valor en su función FO(S). De esta manera conseguimos reducir la suma ponderada de los criterios de ponderación, lo que en nuestro caso se traduce en minimizar la demanda no cubierta y minimizar la desviación típica de los criterios de equilibrio.

2.4 Tamaño del espacio de búsqueda

Dadas las dimensiones de una solución, conviene detenerse a estudiar el coste computacional ligado a explorar el espacio de soluciones completo para justificar el uso de técnicas heurísticas que guíen la búsqueda, en un espacio tan grande que imposibilita por su propia naturaleza el recorrido exhaustivo de soluciones.

Como se ha explicado en el punto 2.2 una solución es una matriz de dimensiones T x N. En cada una de las posiciones de la matriz puede haber una de las etiquetas contenidas en E. El número total de combinaciones posibles es, por lo tanto:

$$Combinaciones\ totales = |E|^{TxN}$$

Como se puede observar, el espacio de búsqueda crece de forma exponencial con el número de trabajadores que existe en la plantilla y con el horizonte de asignación. Teniendo en cuenta que el horizonte de asignación es (o suele ser) un año completo y que las plantillas de trabajadores en las que el problema es aplicable son de un tamaño considerable, el resultado es un espacio de búsqueda cuya talla es prohibitiva para realizar una búsqueda exhaustiva de soluciones, pues esta requeriría, en los casos estudiados, un tiempo del orden de miles de años de tiempo para completarse.

Ya que el recorrido exhaustivo no es posible, se requieren heurísticas que guíen la búsqueda de soluciones a lo largo del inmenso tamaño que presenta la totalidad de éstas. La utilización de las técnicas heurísticas que utilizaremos no garantiza encontrar la solución óptima al problema, sin embargo, es capaz de dar soluciones considerablemente buenas en tiempos razonables (en nuestro caso, tal y como se describe en los objetivos, inferiores a 30 minutos de cómputo total).

2.5 Ejemplo ilustrativo de la representación de un problema y su solución

Para clarificar la representación del problema y su solución, propondremos un problema simple pero ilustrativo donde diferenciar cada componente.

Supongamos un entorno de negocio pequeño, con una plantilla de 6 trabajadores que pueden hacer 3 turnos distintos: mañana, tarde y noche. Durante 7 días se necesita cubrir un puesto de cada turno, pues es un sistema 7/24 y siempre debe de haber alguien trabajando, excepto el día 3, que no se necesita ninguna mañana pero se necesitan 2 tardes. Además, un trabajador no puede hacer más de 4 turnos iguales seguidos, ni menos de 2 turnos iguales seguidos. Otra restricción añadida es que un trabajador que esté haciendo un turno cualquiera, debe hacer un turno de descanso antes de poder cambiar a

otro turno, excepto para cambiar del turno de noche al turno de mañana, que se exigen como mínimo 2 descansos (por razones de descanso del trabajador). No se permite que un empleado trabaje más de 5 días a la semana. Por último, por razones externas, no se permite que los trabajadores empiecen secuencia de turnos en martes ni que un trabajador haga 2 secuencias distintas de noches seguidas.

Dado este enunciado, los parámetros del problema P serían los siguientes:

- T=6
- N=7
- E={M,T,N,-} (O cualquier otro conjunto de etiquetas representativo)
- D= (representación tabular). Recordemos que en cada posición de la tabla, se almacena la demanda de un turno concreto para un día.

M	1	1	0	1	1	1	1
T	1	1	2	1	1	1	1
N	1	1	1	1	1	1	1

Tabla 1. Representación de la demanda.

- R:
  - Restricciones generales: Utilizaremos una representación tabular de las mismas.

    - Longitudes del tamaño de secuencia: en el enunciado se nos especifica que ningún trabajador puede hacer más de 4 turnos iguales ni menos de 2 turnos iguales seguidos. Por lo tanto, estas restricciones tendrán la siguiente forma:

Etiqueta	Tamaño máximo	Tamaño mínimo
M	4	2
T	4	2
N	4	2

Tabla 2. Representación de la restricción de longitudes de secuencia.

    - Turnos totales máximos = 5

- Descansos obligatorios entre secuencias: (representación tabular). El problema nos indica que se requiere un descanso entre dos turnos distintos, exceptuando el caso de noche-mañana y noche-noche. La representación tabular de esta restricción es la siguiente:

Cambio	M-M	M-T	M-N	T-M	T-T	T-N	N-M	N-T	N-N
Descansos	1	1	1	1	1	1	2	1	Prohibido

Tabla 3. Restricción de descansos obligatorios entre secuencia.

- Restricciones especiales (recordemos que se expresan en lenguaje natural):
  - Restricción 1: Prohibido empezar secuencia en Martes.

Dado el problema P descrito, una posible solución podría ser:

- S:

	Día 1: Lunes	Día 2: Martes	Día 3: Miércoles	Día 4: Jueves	Día 5: Viernes	Día 6: Sábado	Día 7: Domingo
Trabajador 1	M	M	-	-	N	N	N
Trabajador 2	T	T	-	-	M	M	M
Trabajador 3	N	N	-	-	T	T	T
Trabajador 4	-	-	T	T	-	-	-
Trabajador 5	-	-	T	T	-	-	-
Trabajador 6	-	-	N	N	-	-	-

Tabla 4. Solución al problema de ejemplo.

Como se puede observar, esta solución no viola ninguna restricción, por lo tanto es una solución válida. Además, esta solución cubre el 100% de la demanda existente.

Por último, para reafirmar la necesidad de las búsquedas guiadas que utilizaremos en el método de resolución, cabe destacar el tamaño del espacio de búsqueda de este pequeño problema. Con sólo 4 etiquetas, 6 trabajadores y 7 días, el espacio de búsqueda total ya asciende a:

$$Combinaciones\ totales = 4^{6*7} = 19.342.813.113.834.066.795.298.816$$

Como se puede observar, incluso en este problema de tamaño irrisorio, el espacio de búsqueda también presenta un tamaño completamente prohibitivo para realizar una búsqueda que explore todas las soluciones posibles.

3. Método de resolución

La resolución a un problema P que sigue la descripción descrita en el punto anterior, consta de tres fases diferenciadas, que explicaremos detalladamente en los epígrafes siguientes:

1) Creación de un grafo representativo de las restricciones generales.
2) Proceso de búsqueda de la solución. Se devuelve una solución factible.
3) Ejecución de postprocesos que optimizan los criterios de optimización de la solución obtenida en el paso 2).

Durante la primera fase, creamos un grafo que representa todos los posibles cambios de secuencia y turnos que puede realizar cada trabajador. Este grafo se construye únicamente a partir de las restricciones de “descansos entre secuencias” y “longitudes máximas y mínimas de secuencia”.

Una vez creado el grafo, el proceso de búsqueda recorre este grafo, para cada trabajador y para cada día, de forma iterativa generando soluciones factibles. El proceso de búsqueda se ejecuta múltiples veces y tras la finalización del proceso, éste presenta como solución el mejor estado final que haya encontrado en todas las iteraciones.

Por último, en la tercera fase, se lanzan de forma iterativa distintos postprocesos que mejoran la solución obtenida por el proceso de búsqueda, pues ésta genera algunas situaciones que empeoran las soluciones pero que pueden mejorarse sin excesivo esfuerzo adicional.

3.1 Fase 1: Grafo de restricciones generales

Es en esta primera fase en la que se construye el núcleo del proceso resolutivo del problema. Crearemos un grafo etiquetado y dirigido que utilizaremos para guiar la búsqueda de la solución. Este grafo, asegurará que se cumplen las restricciones generales del problema, lo que permite que no sea necesario emplear tiempo extra en la validación de soluciones, ahorrando así una gran cantidad de tiempo computacional.

El algoritmo de construcción del grafo etiquetado dirigido para la resolución de un problema P dado se expone a continuación:

1) Crear un nodo inicial con la etiqueta "–".
2) Añadir una arista a sí mismo, creando un bucle en el nodo inicial.
3) Para cada etiqueta e ∈ E, crear un nodo etiquetado con e. A estos nodos les llamaremos nodos de inicio de secuencia.
4) Desde el nodo inicial, añadir una arista hacia cada uno de los nodos de inicio de secuencia.
5) Para cada nodo de inicio de secuencia *n* con etiqueta *e*, consultar su talla máxima N. Añadir N-1 nodos partiendo del nodo n, uno tras otro, etiquetados con *e*. A cada camino desde el nodo inicial hasta cada hoja del grafo actual, lo llamaremos secuencia (excluyendo el nodo inicial).
6) Para cada secuencia, consultamos el número máximo de descansos necesarios *N* antes de iniciar otra secuencia de turnos. Al final de cada secuencia, añadimos una secuencia de *N* nodos etiquetados con "-".
7) Para cada etiqueta, consultamos su tamaño mínimo *N*. Recorremos N-1 nodos desde el nodo inicial de la secuencia. Desde todos los nodos etiquetados con

etiquetas distintas a "–" que se puedan alcanzar desde el nodo actual, se añaden aristas hasta el primer descanso tras la secuencia.

8) Por último, para cada cambio de turnos *e1-e2* (*e1,e2* ∈ *E*) se consulta su valor *N*.

   8.1) Si *N* es mayor que cero, añadimos una arista desde el final de la secuencia de *e1* + *N* descansos, hacia e2 y otra en cada uno de los descansos posteriores.

   8.2) Si *N* es cero, se añaden aristas desde todos los nodos etiquetados con e1 y de todos los descansos posteriores hasta e2.

Para ejemplificar el algoritmo de creación del grafo realizaremos una traza del mismo. Supongamos un problema P caracterizado por las siguientes restricciones generales:

Etiqueta	Tamaño máximo	Tamaño mínimo
M	2	1
T	4	1
N	3	1

Tabla 5. Restricción de tamaño de secuencias ejemplo.

Cambio	M-M	M-T	M-N	T-M	T-T	T-N	N-M	N-T	N-N
Descansos	2	2	2	1	1	0	3	1	Prohibido

Tabla 6. Restricción de descansos obligatorios entre secuencia ejemplo.

La traza del algoritmo sería la siguiente:

Pasos 1 y 2) Crear un nodo inicial con la etiqueta "-". Añadir una arista a sí mismo.

Imagen 1. Pasos 1 y 2 de construcción del grafo.

Pasos 3 y 4) Para cada etiqueta e ∈ E, crear un nodo etiquetado con e. Desde el nodo inicial, añadir una arista hacia cada uno de los nodos de inicio de secuencia.

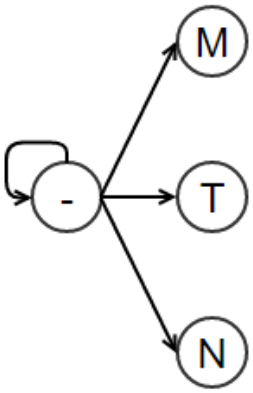

Imagen 2. Pasos 3 y 4 de construcción del grafo.

Paso 5) Para cada nodo de inicio de secuencia *n* con etiqueta *e*, consultar su talla máxima N. Añadir N -1 nodos partiendo del nodo n, uno tras otro, etiquetados con *e*.

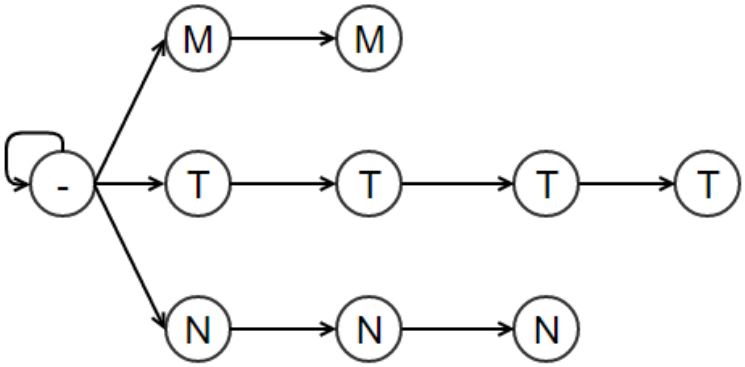

Imagen 3. Paso 5 de construcción del grafo.

Paso 6) Para cada secuencia, consultamos el número máximo de descansos necesarios *N* antes de iniciar otra secuencia de turnos. Al final de cada secuencia, añadimos una secuencia de *N* nodos etiquetados con "-".

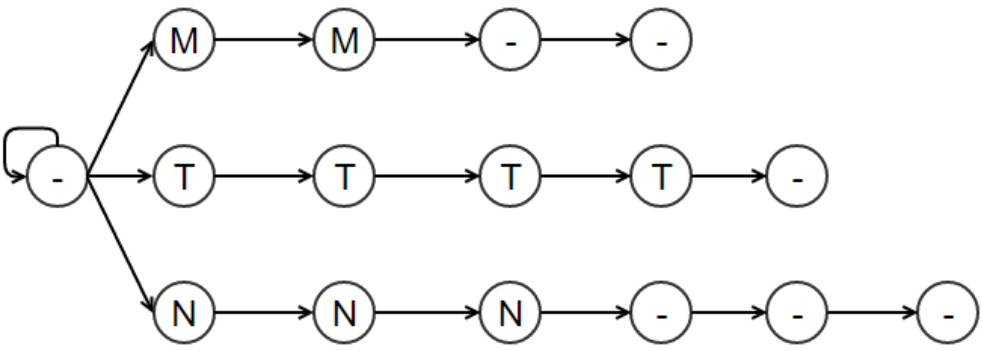

Imagen 4. Paso 6 de construcción del grafo.

Paso 7) Para cada etiqueta, consultamos su tamaño mínimo *N*. Recorremos N - 1 nodos desde el nodo inicial de la secuencia. Desde todos los nodos etiquetados con etiquetas distintas a "-" que se puedan alcanzar desde el nodo actual, se añaden aristas hasta el primer descanso tras la secuencia.

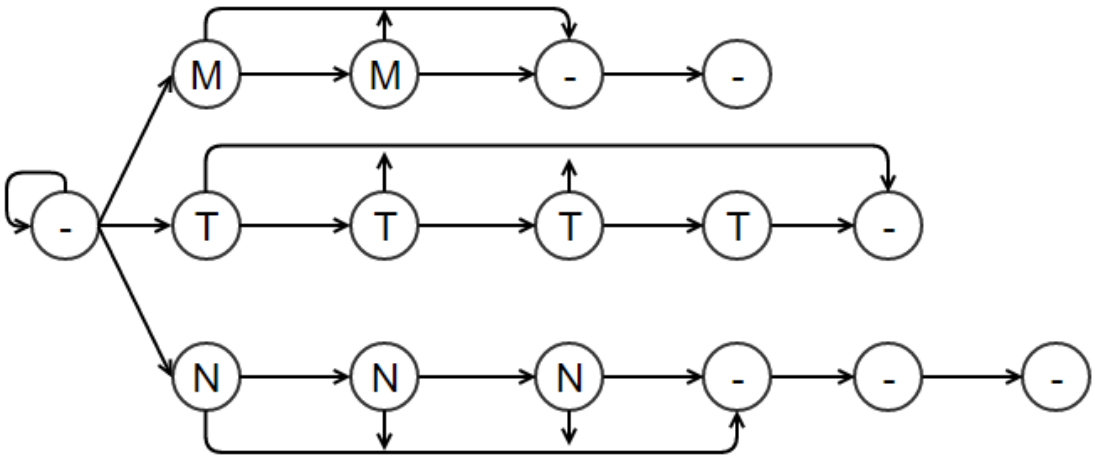

Imagen 5. Paso 7 de construcción del grafo.

Pasos 8.1 y 8.2) Por último, para cada cambio de turnos e1-e2 (e1,e2 ∈ E) se consulta su valor N. Si N es mayor que cero, añadimos una arista desde el final de la secuencia de e1 + N descansos, hacia e2 y otra desde cada uno de los descansos posteriores. Si *N* es cero, se añaden aristas desde todos los nodos etiquetados con e1 y de todos los descansos posteriores hasta e2.

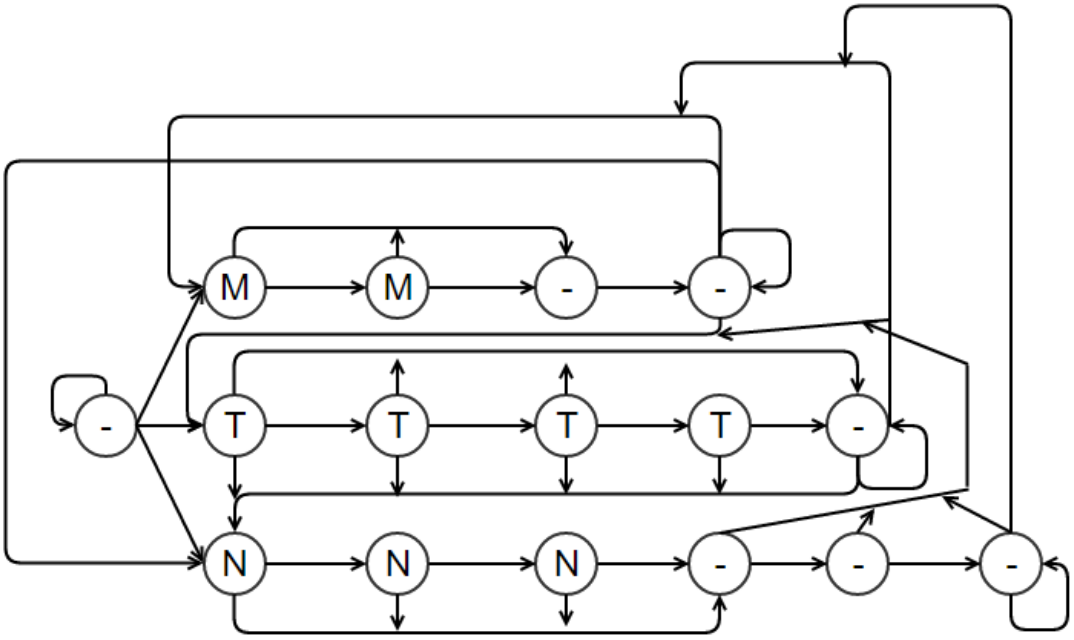

Imagen 6. Paso 8 de construcción del grafo. Resultado final.

Como se puede observar, la construcción del grafo es un proceso determinista y, aunque algorítmicamente puede suponer un reto, computacionalmente tiene un coste de orden lineal con el número de restricciones del sistema, por lo que es un proceso computacionalmente aceptable al no requerir un tiempo excesivo.

3.2 Fase 2: Proceso de búsqueda de la solución

3.2.1 Metaheurística utilizada: GRASP

GRASP (*Greedy Randomized Adaptive Search*) es una metaheurística constructiva propuesta inicialmente por Feo y Rezende en 1995. Esta metaheurística trata de dar buenas soluciones a problemas de alta complejidad combinatoria, como el que nos ocupa en este trabajo.

GRASP, en su versión básica, propone como método de resolución una metodología en dos fases, que se repiten alternativamente de forma iterativa y que al combinarse logran proporcionar buenas soluciones a problemas complejos.

Primero tiene lugar una fase constructiva en la que se obtiene como producto una solución factible al problema mediante una heurística constructiva. Esta primera solución se considera razonablemente buena, aunque no es ni óptima ni tan siquiera un óptimo local. Esta fase no requiere de una solución inicial y la longitud (tamaño) de la solución debe ser limitada, así como los elementos a añadir a la solución deben pertenecer a un conjunto finito y considerablemente pequeño para lograr una baja ramificación.

En segundo lugar, se ejecuta una fase de mejora con búsqueda local en la que se modifica la solución que se ha obtenido tras la primera fase. Esta fase tiene como finalidad alcanzar soluciones cercanas, llamadas “soluciones vecinas”, a la que se obtuvo en la fase previa pero que mejoren sus condiciones tratando de aproximarla a la solución óptima.

Cuando se ha obtenido el óptimo local durante la segunda fase, la solución se almacena y se vuelve a comenzar otra iteración con la primera fase. Este proceso continúa ejecutándose, almacenando siempre la mejor solución encontrada hasta el momento, por lo que al finalizar, se tendrá guardada la mejor solución que se haya encontrado a lo largo del proceso.

Para nuestra búsqueda utilizaremos una variante del esquema GRASP básico en el que ejecutaremos la fase de búsqueda de forma iterativa almacenando la mejor solución encontrada. Cuando esta fase termine todas sus iteraciones, entonces aplicaremos la segunda fase, de búsqueda local, sólo sobre la mejor solución obtenida durante la primera fase, almacenando finalmente la solución que optimice la funcione de optimalidad, pues recordemos que la de factibilidad se da por satisfecha por la definición que le hemos dado a la función que resuelve el problema.

3.2.2 Búsqueda implementada

Una vez creado el grafo del apartado anterior, lo utilizaremos para realizar una búsqueda eficiente de una solución. El planteamiento general consiste en utilizar el grafo para guiar el proceso de búsqueda de forma que se generen sólo soluciones válidas para que no sea necesario una validación de soluciones y por lo tanto el tiempo computacional asociado. Cabe destacar que esta fase es la que corresponde a la implementación de la fase constructiva de la variante de la metaheurística GRASP que hemos utilizado.

Durante el proceso de búsqueda, cada uno de los trabajadores del grafo se representa como un puntero hacia un nodo del grafo. Todos los trabajadores se representan sobre el mismo grafo y son únicamente un puntero, por lo que el proceso requiere muy poca memoria en cuanto a representación se refiere. Para realizar la asignación, los trabajadores (en concreto, sus punteros asociados), se mueven a lo largo del grafo. Todos los punteros empiezan apuntando al nodo inicial etiquetado con “-”. Cuando se quiere asignar un turno para todos los trabajadores en un día concreto, el proceso que se basa en el grafo se realiza de la siguiente forma, expresado en pseudocódigo:

```
function asignarUnDia(dia){
disponibles=todos_los_punteros
Mientras diponibles.length > 0:
      trabajador = disponibles.remove(random(disponibles.length ))
      if trabajador.turnosTotalesRealizados >= maximoAnualDeTurnos
            trabajador.asignarDescanso()
            siguiente_iteración
      posibles=trabajador.nodos_sucesores_en_el_grafo().getEtiquetas()
      necesarias= getEtiquetasDemandaDelDia(dia)
      válidas = interection(posibles,necesarias)
      etiquetaElegida = válidas.randomChoice().remove()
      trabajador.asignarTurno(etiquetaElegida)
      trabajador.avanzarEnElGrafoALaSiguiente(etiquetaElegida)
      getDemandaDelDia(dia).restarUnTurnoDe(etiquetaElegida)
}
```

Este proceso de asignar los turnos de un día se repite de forma iterativa hasta que se han asignado todos los turnos para todos los trabajadores en el horizonte de asignación indicado por el usuario. Nótese que las asignaciones se hacen sólo si el trabajador no ha superado los turnos anuales posibles, por lo que el algoritmo asegura el cumplimiento de la restricción general de máximos turnos anuales. Además, como sólo se pueden asignar etiquetas contenidas en nodos a los que el trabajador puede llegar en el grafo, y el grafo está construido en función de las restricciones generales de longitud y descansos, se puede asegurar que este proceso provoca soluciones válidas, pues no violan ninguna de las restricciones generales.

Una vez se ha realizado la asignación del horizonte completo, esa solución se guarda y el proceso de búsqueda se repite de forma iterativa. Cada vez que acaba una iteración se calcula el error obtenido por la solución de ésta, y si es mejor que la solución que el sistema había guardado, se actualiza dicha solución a la nueva y el proceso se repite de nuevo. Esto se realiza un número de iteraciones variable, en función de las especificaciones del usuario. Se le permite

al usuario especificar una precisión en la búsqueda en valores de “alta”, “media” o “baja” que estiman tiempos aproximados inferiores a diez, tres y un minuto, respectivamente. En función de la elección del usuario y basándose en la complejidad temporal del problema, se estima el número de iteraciones necesario para intentar respetar dichos tiempos.

Cabe también destacar que al utilizar el grafo para guiar la búsqueda, estamos reduciendo el espacio de búsqueda de las soluciones a un nuevo espacio considerablemente más pequeño, pues todas las soluciones contenidas en el espacio de búsqueda definido en el punto 2.3 no son válidas. De hecho, la mayoría de ellas no lo son, por lo que el espacio de búsqueda resultante queda reducido a un tamaño más explorable porque el grafo sólo explora soluciones válidas.

3.3 Fase 3: Postprocesos de mejora local

Esta fase es la que corresponde a la segunda parte de la variante GRASP que estamos utilizando. Conceptualmente, los postprocesos tratan de mejorar la solución obtenida por el algoritmo descrito en el apartado 3.2. Debido a la aleatoriedad del algoritmo, puede suceder, y de hecho sucede, que demanda que pudiera haberse cubierto de forma trivial, no se cubra porque la aleatoriedad haya decidido asignar un descanso en lugar de un turno de trabajo. Por el mismo motivo, sucede la situación contraria, en la que se asignan turnos que no son necesarios, provocando así un exceso de demanda que provoca que el trabajador en cuestión alcance antes el máximo de turnos anuales y provocando que el final de las soluciones quede sin asignar. Por lo tanto, se han implementado 4 postprocesos que trataran de mejorar la solución obtenida por la búsqueda. Se debe señalar que la aplicación de los postprocesos se realiza sólo sobre la solución final obtenida por el proceso de búsqueda y no sobre cada una de las soluciones que se encuentran en las consecutivas iteraciones del algoritmo de búsqueda.

A continuación describiremos cada uno de los postprocesos implementados y qué situaciones que se dan en las soluciones obtenidas en la

búsqueda han motivado su estudio e implementación. Dada la alta casuística de la tipología del problema, no se pueden prever todas las situaciones que se darán en las soluciones, por lo que es posible que ni siquiera estos postprocesos de arreglo sean capaces de solucionar algunos conflictos. Sin embargo, la implementación de estos postprocesos es el resultado de un estudio exhaustivo de múltiples soluciones y de las situaciones que se repiten más comúnmente en las soluciones encontradas durante la búsqueda.

3.3.1 Postproceso1: Liberación de demanda sobrante

A continuación, a fin de expresar la motivación de la necesidad de este postproceso, mostramos un extracto de una solución real obtenida tras el proceso de búsqueda inicial, sin aplicar aún ningún postproceso de arreglo:

Trabajador 0	-	-	-	-	-	
Trabajador 1	-	M	M	M	M	
Trabajador 2	-	M	M	-	-	
Trabajador 3	-	-	M	M	M	
Trabajador 4	-	-	M	M	M	
Trabajador 5	M	M	-	-	T	
Trabajador 6	-	-	T	T	-	
Trabajador 7	T	T	-	-	N	
Trabajador 8	-	T	T	T	T	
Trabajador 9	N	N	-	-	-	
Trabajador 10	-	-	N	N	-	
M	0	0	-1	0	0	
T	0	0	0	0	0	
N	0	0	0	0	0	

Imagen 7. Ejemplo de demanda sobrante.

Como se puede observar en la imagen 7, la búsqueda de la solución ha provocado una situación en la que se ha asignado en exceso un turno de mañanas, es decir, se ha asignado a un trabajador un turno aunque no es necesario. Las posiciones marcadas en rojo son asignaciones que podrían sustituirse por descansos, lo que no empeoraría la solución, pues la demanda estaría aún cubierta, pero daría un turno libre más a ese trabajador.

Esta situación empeora mucho las soluciones otorgadas por el sistema, dado que es muy común y se repite con un alto índice de probabilidad, porque provoca que los trabajadores alcancen de forma prematura el máximo de turnos anuales. Recordemos que, cuando esto sucede, la asignación de turnos al trabajador en cuestión se detiene. Así pues, si observamos la asignación al final del horizonte de asignación, en la imagen 8, observamos que hay un alto porcentaje de demanda sin cubrir debido a que los trabajadores ya han alcanzado el máximo anual de turnos, establecido en 194:

Dic	Dic	Dic	Dic	Dic	Dic	Dic	Dic	Dic	
23 D	24 L	25 M	26 X	27 J	28 V	29 S	30 D	31 L	Total
-	-	M	M	M	M	M	-	-	191
-	-	-	-	-	-	-	-	-	194
-	-	-	-	-	-	-	-	-	194
M	M	-	-	M	M	M	M	M	193
-	-	-	-	-	-	-	-	-	194
-	M	M	-	-	-	-	-	-	194
-	T	T	T	T	T	-	-	T	190
-	-	-	-	-	-	-	-	-	194
T	T	T	-	-	T	T	T	T	190
-	-	N	N	N	-	-	-	-	194
N	-	-	-	-	-	-	-	-	194
0	1	-1	2	1	1	1	0	2	
0	0	-1	1	1	0	1	0	0	
0	1	0	0	0	1	1	1	1	

Imagen 8. Ejemplo de falta de cubrimiento al final del año.

Se ha marcado en rojo la demanda que ha quedado sin cubrir así como los trabajadores cuya asignación ha quedado bloqueada a causa de alcanzar el máximo anual permitido. Tal y como se muestra en la figura, si estos trabajadores no estuvieran bloqueados, la demanda podría haberse cubierto en su totalidad.

Con el fin de dar solución a esta situación, se ha implementado el postproceso1 de arreglo de soluciones. Este proceso recorre la matriz solución y libera todos los turnos que se haya asignado en exceso, lo que provoca una disminución de los turnos anuales que ha realizado cada trabajador, otorgando así un margen de turnos que aún se puede asignar sin violar la restricción de turnos máximos anuales. Estos nuevos turnos disponibles serán utilizados por

los postprocesos siguientes para intentar cubrir la demanda que en primera instancia no haya podido cubrirse.

3.3.2 Postproceso2: Cubrimiento mediante relleno de huecos

Este postproceso se implementa con el fin de poder dar solución a situaciones trivialmente solucionables de demanda sin cubrir, como la que se muestra a continuación, extraída de una solución real a la que ya se le ha aplicado el postproceso1 de arreglo:

Trabajador 0	-	-	-	-	-	-	N	N	-	-	T	T	-	-
Trabajador 1	-	M	M	M	M	-	-	N	N	-	-	T	T	-
Trabajador 2	-	M	M	-	-	-	M	M	M	M	M	-	-	-
Trabajador 3	-	-	M	M	M	-	-	M	M	-	-	M	M	M
Trabajador 4	-	-	M	M	M	M	-	-	M	M	M	M	M	-
Trabajador 5	M	M	-	-	T	T	-	-	N	N	N	N	-	-
Trabajador 6	-	-	T	T	-	-	-	M	M	M	-	-	-	-
Trabajador 7	T	T	-	-	N	N	-	-	T	T	-	-	N	N
Trabajador 8	-	T	T	T	T	-	-	-	-	-	-	-	T	T
Trabajador 9	N	N	-	-	-	-	-	T	T	T	-	-	-	-
Trabajador 10	-	-	N	N	-	-	T	T	-	-	M	M	M	-
M	0	0	-1	0	0	0	0	1	1	1	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
N	0	0	0	0	0	0	0	-1	-1	0	0	0	0	0

Imagen 9. Ejemplo de hueco sin cubrir.

En la imagen 9 se marca en rojo tres demandas que han quedado sin cubrir junto con un trabajador que presenta 1 hueco de tamaño 3 que podría haberse rellenado con el turno de mañanas para cubrir esa demanda insatisfecha. De nuevo, esta situación también es muy común en las soluciones encontradas durante la búsqueda, lo que provoca que un alto porcentaje de la demanda quede sin cubrir.

Así pues, el postproceso 2 trata de lidiar con estas situaciones. Este proceso, al ejecutarse, recorre la matriz solución detectando demanda sin cubrir y comprobando si esta puede ser cubierta mediante el rellenado de huecos en la asignación de un trabajador. Cabe destacar que este postproceso sólo rellenará

el hueco con el tamaño mínimo permitido de la secuencia, pues será labor de otro postproceso posterior alargar las secuencias si fuera necesario. También se debe tener en cuenta que este proceso sólo rellena huecos, pero no alarga secuencias ya existentes. Por lo tanto, este postproceso sólo es capaz de rellenar huecos que puedan cubrir déficit de demanda pero que estén rodeados de los descansos necesarios a ambos lados para no violar las restricciones de descansos mínimos necesarios.

3.3.3 Postproceso3: Cubrir demanda alargando secuencias

De nuevo, presentaremos una situación real extraída de una solución proporcionada por el sistema tras la aplicación de los postprocesos anteriores para su posterior análisis tratando de justificar la implementación del postproceso de arreglo asociado:

Trabajador 0	-	-	-	-	-	-	N	N	-	-	T
Trabajador 1	-	M	M	M	M	-	-	N	N	-	-
Trabajador 2	-	M	M	-	-	-	M	M	M	M	M
Trabajador 3	-	-	M	M	M	-	-	M	M	-	-
Trabajador 4	-	-	M	M	M	M	-	-	M	M	M
Trabajador 5	M	M	-	-	T	T	-	-	N	N	N
Trabajador 6	-	-	T	T	-	-	-	-	M	M	-
Trabajador 7	T	T	-	-	N	N	-	-	T	T	-
Trabajador 8	-	T	T	T	T	-	-	-	-	-	-
Trabajador 9	N	N	-	-	-	-	-	T	T	T	-
Trabajador 10	-	-	N	N	-	-	T	T	-	-	M
M	0	0	0	0	0	0	1	1	1	1	0
T	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	1	0	0	-1	-1	0	0

Imagen 10. Ejemplo de alargamiento de secuencias.

La imagen 10 presenta una situación en la que se observa demanda sin cubrir que podría cubrirse si se alargan las secuencias de turnos que rodean a las posiciones señaladas en rojo. Nótese que el postproceso 2 no ha arreglado esta situación pues, como indicábamos en el apartado 3.3.2, este postproceso sólo es capaz de arreglar situaciones que requieran rellenar huecos, entendiendo como huecos aquellos lugares rodeados de descansos. A esta situación no le es aplicable ese método, pues al rellenar estas demandas sin cubrir, el resultado no queda rodeado de descansos a ambos lados.

Es decir, la naturaleza de esta situación es distinta a la situación anterior, pues en la anterior pretendíamos insertar nuevos datos en la solución sustituyendo lugares etiquetados con descanso por etiquetas de turnos, pero en esta sólo pretendemos hacer más largos los datos que ya existen.

Así pues, con el fin de alargar las secuencias existentes para cubrir demanda que no quedó cubierta en la solución inicial, se implementa el postproceso 3. Éste, se dedica a consultar todas las columnas de la solución de manera aleatoria, comprobando en cada caso si ésta tiene un déficit de demanda que pudiera ser cubierto por algún trabajador mediante la extensión de alguna de las secuencias que rodean al hueco en cuestión. De ser así, el postproceso realiza las operaciones necesarias para lograr que el hueco quede relleno repitiendo la etiqueta de la secuencia de alrededor que consiga este efecto.

3.3.4 Postproceso4: Eliminar turnos en exceso

Como ya se ha comentado anteriormente (punto 3.2), durante la búsqueda no se va a permitir que se excedan los turnos máximos anuales especificados como parámetros en el problema. Sin embargo, los postprocesos2 y 3 no tienen en cuenta este máximo y pudiera suceder que al arreglar algunos casos de la solución inicial, esto supusiera un aumento de turnos tal que excediera el máximo permitido.

Ante esta situación se plantean 2 soluciones posibles: impedir que los postprocesos no superen el máximo anual o permitir que lo excedan pero generar luego un postproceso que vuelva a reducir los turnos realizados por cada trabajador si superan el máximo anual permitido.

Habiendo realizado pruebas experimentales mediante ejecuciones sucesivas, la segunda opción muestra mejores resultados al cubrir un mayor porcentaje de la demanda. Esto se debe a que, si bloqueamos la opción de exceder los turnos máximos anuales, el sistema tiene una menor adaptabilidad a distintos entornos, pues es posible que para rellenar un hueco vacío necesite colocar una secuencia entera debido a las restricciones de tamaño mínimo, pero si al colocar la secuencia entera, excediera los máximos anuales, la secuencia no

se colocaría y por lo tanto la demanda quedaría sin cubrir. Sin embargo, al permitir el exceso de turnos, el algoritmo permitiría suplir esa demanda, y a su vez, esto podría generar un exceso en otro lugar que el postproceso de eliminar excesos se encargaría de eliminar, quedando así la demanda cubierta y los turnos asignados por debajo del máximo.

Así pues, para que esta funcionalidad sea correcta, se precisa la especificación de un método que elimine asignaciones de turnos a trabajadores aunque se deje demanda sin cubrir. Nótese la diferencia con el postproceso1, ya que éste sólo elimina asignaciones si exceden la demanda. Sin embargo, el postproceso 4 que proponemos elimina asignaciones aunque éstas empeoren la solución, con el fin de garantizar que los máximos turnos anuales por trabajador no se excedan.

Darle preferencia al máximo de turnos anuales frente al cubrimiento de demanda es una decisión de implementación que se ha tomado y no puede modificarse mediante parámetros de usuarios. Esto es así porque, en la mayoría de entornos estudiados, los máximos turnos anuales por trabajador son una restricción de carácter legal que no puede violarse, mientras que la demanda sin cubrir puede solucionarse añadiendo un trabajador más.

3.4 Implementaciones adicionales

Tal y como se explicó en el punto 2, los problemas tienen dos tipos de restricciones. El método de resolución basado en el grafo y los posteriores postprocesos aseguran que las restricciones generales no se violan en ninguna medida, lo que ya provoca que el software sea adaptable a distintos entornos.

Pero además de la satisfacción de estas restricciones, también se han implementado otras restricciones que, aunque son específicas de cada entorno y por lo tanto no generalizables, sí que se encuentran en un alto número de situaciones. Las implementaciones que describiremos a continuación son opcionales y pueden desactivarse si el usuario así lo prefiere, por lo que no

empeora la adaptabilidad del sistema pero sí mejora en algunos entornos específicos donde estas restricciones sean aplicables.

En concreto, se han implementado cuatro restricciones especiales que se listan a continuación.

- **Fines de semana libre:** en algunos entornos empresariales donde se realizan turnos rotatorios se exige que cada trabajador tenga, al menos, un fin de semana libre al mes. Esta restricción, en lo que a nosotros concierne, se traduce como que, al menos una vez al mes, un trabajador debe tener asignado turnos de descanso en un sábado y un domingo consecutivos. Cuando el usuario activa esta restricción, el sistema se adapta e intenta que se cumpla en todos los trabajadores, aunque no siempre lo consigue, debido a que esta restricción es muy prohibitiva en cuanto a las asignaciones disponibles y estropea en gran medida las soluciones. Para ofrecer al usuario una respuesta con respecto a esta restricción, el sistema ofrece una salida al usuario, en forma de tabla, en la que se especifica para cada par mes-trabajador si se ha podido satisfacer la restricción del fin de semana libre mínimo o no.

- **Turnos prohibidos antes de fin de semana libre:** esta restricción está relacionada con la anterior, y tanto es así, que ésta segunda no es aplicable a no ser que lo sea también la primera. Esta restricción hace referencia a qué turnos no pueden realizarse en viernes para considerar como fin de semana libre los dos descansos posteriores. Es muy común que en entornos 7/24, si un trabajador realiza turno de noche un viernes, aunque el sábado y domingo descanse, estos no se consideren fin de semana libre para el cómputo de un fin de semana libre al mes, por razones de descanso de personal. Así pues, el sistema puede recibir como entrada un conjunto de etiquetas de turnos y asegurar que un trabajador no tendrá un fin de semana libre después de realizar alguno de los turnos especificados.

- **Prohibido empezar secuencia en:** esta restricción implementada hace referencia a unos días en los que no se permite que un trabajador empiece una secuencia de turnos. Esta situación se da con asiduidad en entornos en los que no se permite empezar una secuencia de turnos en domingo y se exige que se trabaje el sábado si se requiere trabajar el domingo siguiente. Como extensión a esa restricción concreta, la implementación llevada a cabo permite que se prohíban los inicios de secuencia cualquier día de la semana. Al usuario se le ofrece la posibilidad de especificar un conjunto de días, que puede contener tantos como se requieran, y el sistema asegura que ninguno de los días introducidos se empezará una secuencia. Esta restricción se contempla durante la búsqueda y los postprocesos, los cuales comprueban, antes de asignar un turno, si el día especificado está entre los que el usuario ha marcado como prohibidos. Dado que la restricción se comprueba durante la búsqueda y los arreglos posteriores, se asegura su cumplimento al 100% al igual que las restricciones generales.

- **Bloquear turnos a trabajadores:** también se incluye la posibilidad de bloquear a cualquier trabajador un conjunto cualquiera de turnos, permitiendo que a ese trabajador no se le asigne un turno concreto. Esta implementación trata de cubrir una situación muy extendida en entornos empresariales en los que, por motivos internos o externos, se encuentran trabajadores que no realizan nunca algún turno especificado.

Además de estas restricciones especiales, también se han implementado criterios de optimización del equilibrio de las asignaciones, pues además de que la solución cubra toda la demanda, también se desea que se haga de la forma más equilibrada posible, pues no son deseables situaciones en las que un trabajador, o varios, realicen muchos turnos iguales, si no que se buscan soluciones en las que la asignación de turnos sea lo más homogénea posible. Para lograrlo, se le permite al usuario especificar una ponderación de optimización de los criterios “cubrimiento de demanda” y “equilibrio de turnos”. En función de los parámetros especificados por el usuario, la búsqueda evalúa de forma ponderada las soluciones que genera en cada iteración, quedándose

siempre con la que optimice la función de evaluación ponderada con los criterios del usuario.

4. Interfaz gráfica de usuario

A continuación expondremos el trabajo realizado con respecto a la construcción de la interfaz gráfica de usuario (IGU) que acompaña al sistema que resuelve el problema de la asignación de turnos. Toda la implementación de dicha interfaz ha sido llevada a cabo con la tecnología de JavaFX-2.0.2.

4.1 ¿Qué es JavaFX?

JavaFX es una tecnología software basada en Java que permite el desarrollo de interfaces gráficas de usuario ofreciendo facilidades visuales y herramientas de implementación al desarrollador.

Además, JavaFX viene acompañada de la creación de ficheros FXML que representan mediante lenguajes de marcado la interfaz desarrollada y de herramientas externas para editar dichos ficheros. También permite la utilización de cualquier librería de Java, lo que proporciona también toda su potencia expresiva.

Como principal objetivo de JavaFX encontramos el hecho de que sus implementaciones y todos los productos desarrollados con su tecnología puedan ser utilizados en múltiples tipos de dispositivos, ofreciendo un soporte de tipo multiplataforma y siendo fiel al paradigma heredado de java “escribir una vez, ejecutar en cualquier parte” (del inglés “*write once, run anywhere”),* lo que nos beneficia en la realización de este proyecto, pues tal y como se describió en los objetivos, tratamos de crear un sistema generalizable y adaptable a múltiples entornos distintos.

4.2 Interfaz implementada

A continuación expondremos el aspecto visible de la interfaz de la aplicación implementada, numerando y nombrado cada ventana. Cuando algún evento sobre la interfaz provoque una navegación a otra ventana, lo indicaremos con una flecha sobre el elemento receptor del evento y el nombre y número de la ventana hacia la que se dirige la navegación.

1) Ventana principal

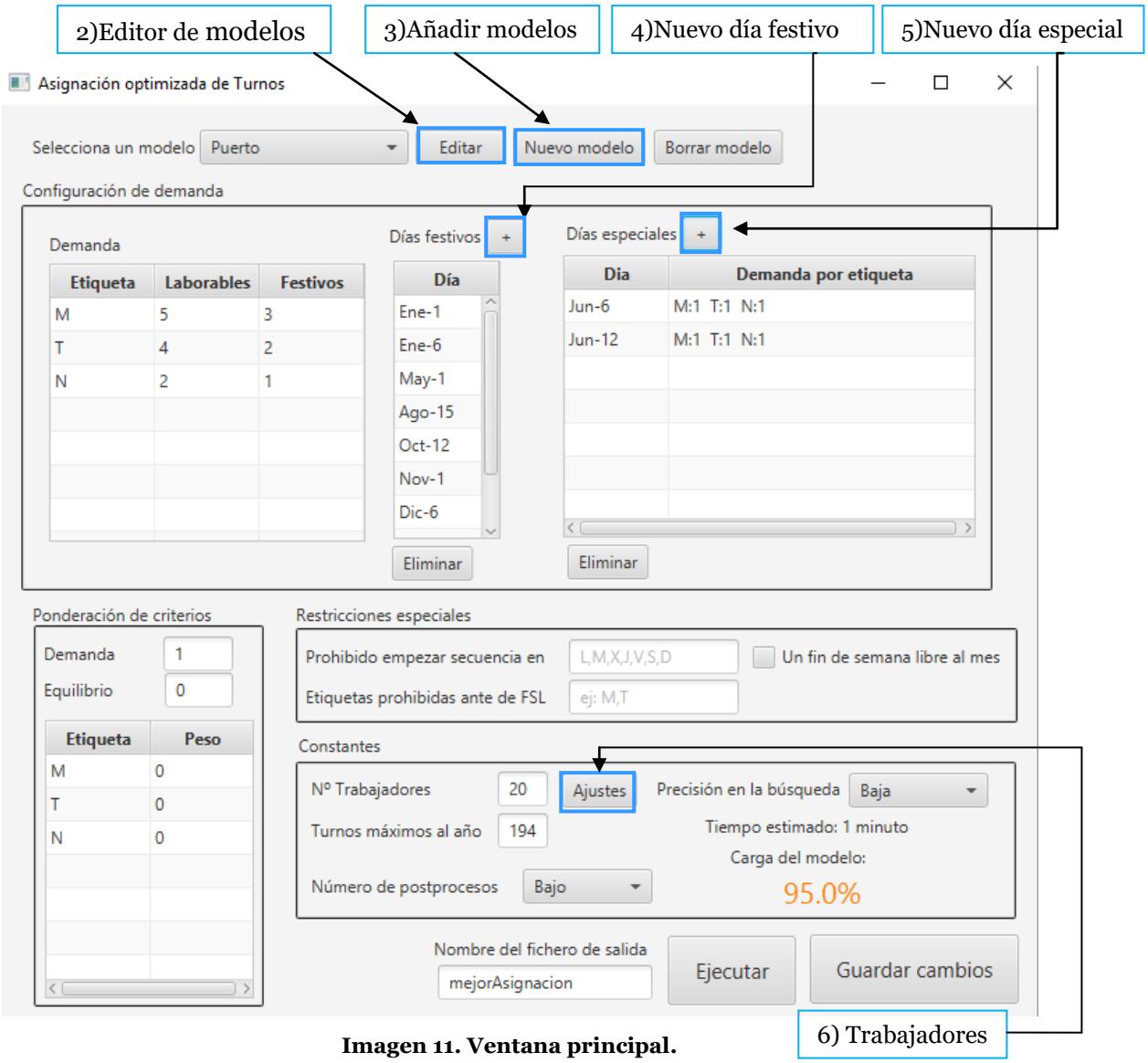

Imagen 11. Ventana principal.

La ventana principal es la primera interfaz que se muestra al arrancar la aplicación. En ella se muestran las distintas opciones de configuración y parametrización del modelo (o problema) que se le permiten modificar al usuario, así como la posibilidad de crear nuevos modelos o cambiar entre los ya existentes. En concreto, cada uno de los elementos de la interfaz y su funcionalidad se describen a continuación:

- **Selector de modelos:** esta herramienta permite seleccionar un modelo de entre todos los que el usuario tiene creados. Al hacerlo, todos los parámetros que se muestran en la interfaz son cambiados por los que posea el modelo que ha sido seleccionado.

- **Botón editar:** permite la navegación a la ventana 2)Editor de modelos.

- **Botón nuevo modelo:** permite la navegación a la ventana 3)Añadir modelos.

- **Botón borrar:** permite borrar el modelo que está actualmente seleccionado en el selector de modelos. Al borrar un modelo, se selecciona el siguiente que esté disponible en el selector. Si no hay más modelos disponibles, la interfaz queda completamente deshabilitada a excepción del "botón nuevo modelo", tal y como muestra la imagen 12.

- **Grupo configuración de demanda:** en esta sección se le permite al usuario especificar la demanda que existe de cada turno en cada día concreto. Los elementos que permiten esta funcionalidad son:

    - **Tabla demanda:** este elemento permite que se especifique la demanda que existe de cada turno en los días laborables y festivos.

    - **Tabla día festivos:** es la tabla que recoge los días que tendrán asociada la demanda de los días festivos que se haya especificado en la tabla anterior.

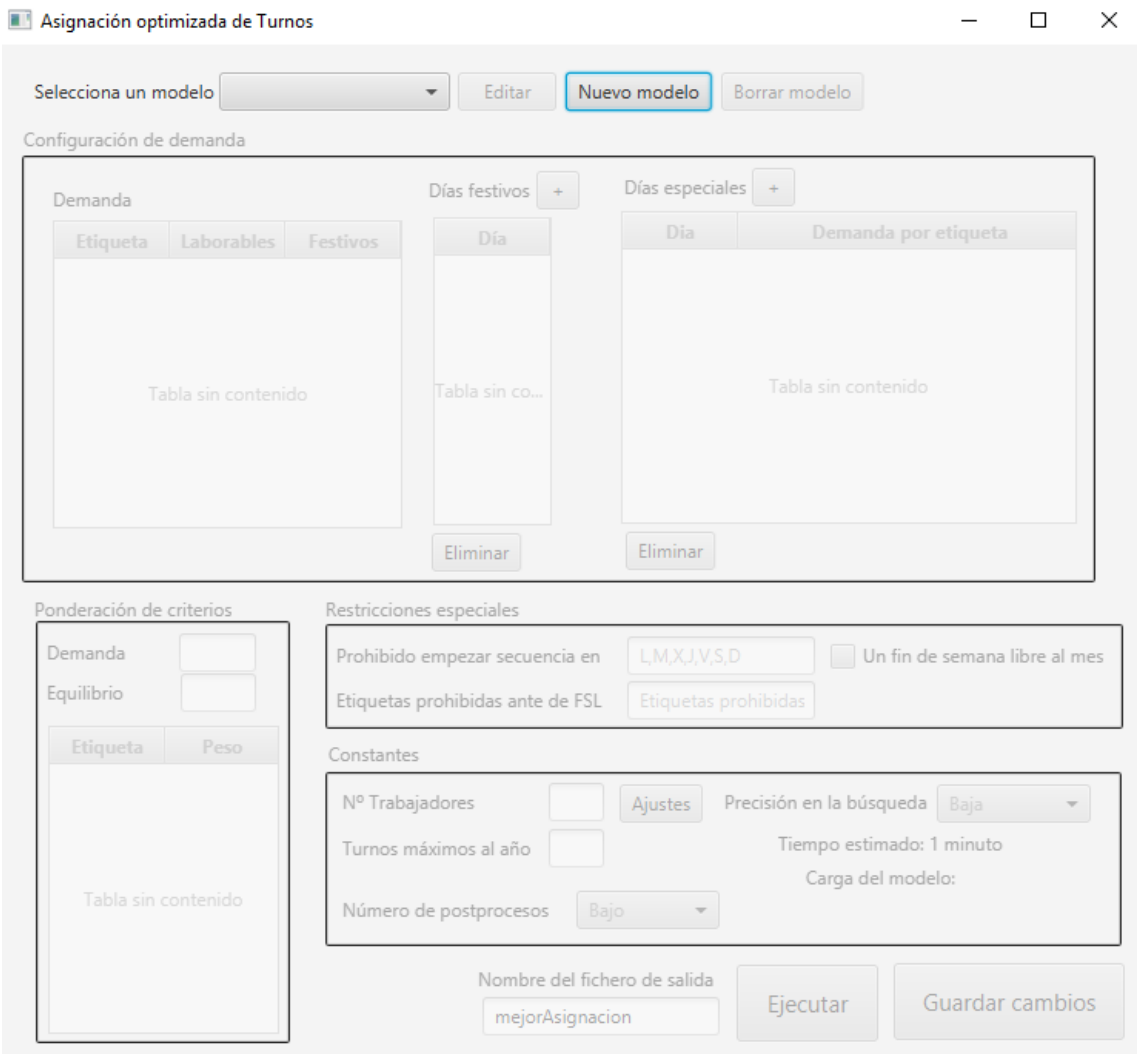

Imagen 12. Interfaz bloqueada.

- **Botón nuevo día festivo**: permite la navegación a la interfaz 4)Nuevo día festivo.

- **Tabla días especiales:** en esta tabla se especifican los días especiales. Un día especial es un día que, sin ser festivo, tiene una demanda diferente a la demanda especificada en días festivos. Esta herramienta permite indicar la demanda por días de forma selectiva y unitaria, previendo situaciones en las que la demanda sea muy irregular. La tabla muestra tanto el día especificado como la demanda por turnos.

- **Botón nuevo día especial:** permite la navegación a la interfaz 5)Nuevo día especial.

- **Grupo de restricciones especiales:** en esta sección se ofrece la funcionalidad de parametrizar la utilización de las restricciones especiales implementadas, ya comentadas en el punto 3.4:

    - **Prohibido empezar secuencia en:** en este campo de texto se permite indicar un conjunto cualquiera de días. El sistema asegura que ninguno de los días especificados en esta restricción se comenzará una secuencia de turnos. Si esta restricción no es aplicable, basta con dejar el campo de texto vacío.

    - **Prohibido antes de FSL:** este campo de texto ofrece la posibilidad de indicar qué turnos está prohibido realizar un viernes para que los 2 descansos posteriores se consideren fin de semana libre (FSL). Si el campo de texto se deja vacío, la restricción no se aplica.

    - **Un fin de semana libre al mes:** si se marca este campo de validación, el sistema intentará que en la solución se cumpla la restricción de un fin de semana libre al mes para todos los trabajadores. Aunque no siempre consigue cumplir esta restricción al 100%, en todos los casos sí que ofrece soluciones aceptablemente buenas con respecto a este aspecto.

- **Grupo de ponderación de criterios:** en este conjunto, se ofrece la posibilidad de ponderar distintos criterios de evaluación de la solución, como ya se indicó en el punto 3.4. En todos los campos de este conjunto se recibe como entrada un número entero positivo. Todos estos números serán después normalizados dando lugar a coeficientes en la función de evaluación que evalúa cuan buena es una solución con el fin de almacenarla o no durante el proceso de búsqueda. Se permite establecer también un coeficiente por cada etiqueta por si se considera que el equilibrio de algún turno en concreto debe prevalecer sobre los demás equilibrios.

- **Grupo de constantes:** en esta sección se permite parametrizar las constantes numéricas del problema así como algunos criterios a aplicar durante la ejecución del proceso que resuelve el problema.

    - **Número de trabajadores:** una constante numérica mayor que cero que indica cuantos trabajadores tiene el problema que vamos a resolver.

    - **Turnos máximos al año:** esta constante numérica representa la restricción general que limita los turnos máximos que puede realizar un trabajador. Este campo no puede dejarse en blanco, pues se exige un valor numérico mayor o igual a cero.

    - **Precisión en la búsqueda:** se le permite elegir al usuario entre 3 opciones: baja, media o alta. Según el valor elegido, las iteraciones que realiza el proceso de búsqueda aumentan o disminuyen. Esta herramienta permite tener soluciones rápidas poco precisas o por el contrario, si se quiere más precisión, se puede aumentar el tiempo de cómputo. En ningún caso el tiempo de cálculo supera los 30 minutos, pues el número de iteraciones que realiza la búsqueda se calcula para respetar este tiempo máximo.

    - **Número de postprocesos:** es un planteamiento similar al anterior. Se permite que el usuario elija cuántas veces quiere que se ejecuten los postprocesos. Cuantos más postprocesos se ejecutan, más mejorará la solución, a costa de un mayor tiempo de cálculo.

    - **Carga del modelo:** en esta etiqueta se muestra la relación que existe entre los turnos posibles que pueden realizar los trabajadores con la demanda total que requiere el sistema, expresado en porcentaje. Este dato es muy útil para que el usuario pueda prever si el problema dado tiene o no solución y, en caso de tenerlo, cuán presionado está el sistema.

- **Nombre del fichero de salida:** la ejecución del sistema provoca como salida un fichero en formato .xlsx (hoja de cálculo libre de versiones) con la solución obtenida. El nombre que tendrá ese fichero será el indicado en este campo de texto.

- **Botón guardar cambios:** este botón permite guardar los cambios que se realicen sobre un modelo o un modelo nuevo creado. Al pulsarlo, los datos se actualizan en la base de datos que soporta el almacenamiento persistente de los datos que se muestran en la interfaz.

- **Botón ejecutar:** este botón permite lanzar a ejecución el problema seleccionado en el selector de modelos con los parámetros indicados en la ventana principal. Al pulsarlo, se lanza a ejecución el proceso de resolución y se produce una navegación a la ventana mostrada en la imagen 13:

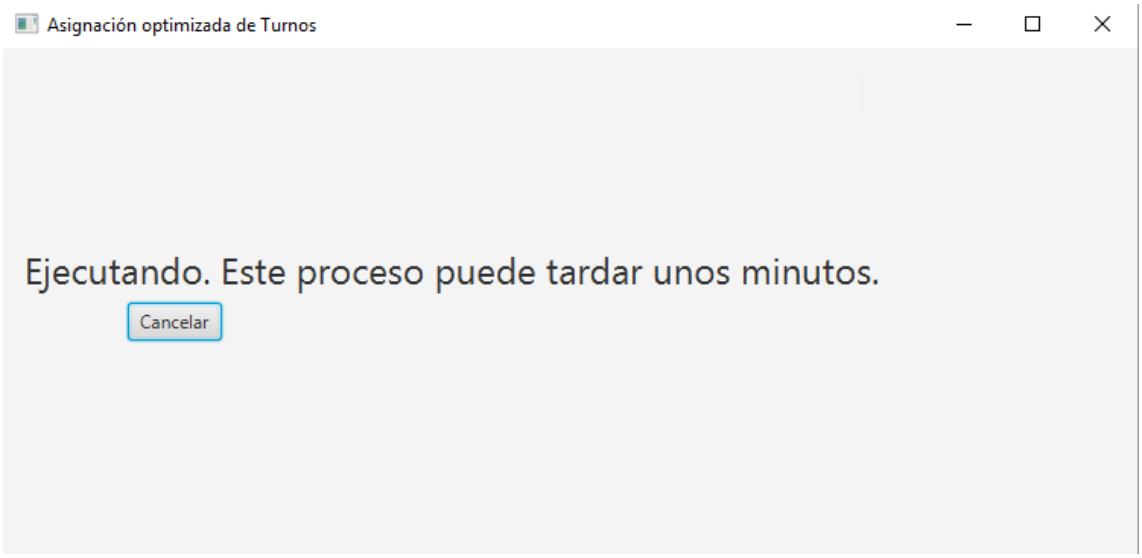

Imagen 13. Interfaz durante la ejecución.

Si durante la ejecución el usuario pulsara el botón cancelar, el proceso termina sin dar solución alguna y la interfaz regresa a la ventana principal. Si por el contrario, el proceso termina y completa la solución de forma correcta sin

que el usuario aborte el proceso, se muestra la interfaz que mostramos en la imagen 14, donde al pulsar el botón aceptar se regresa a la ventana principal.

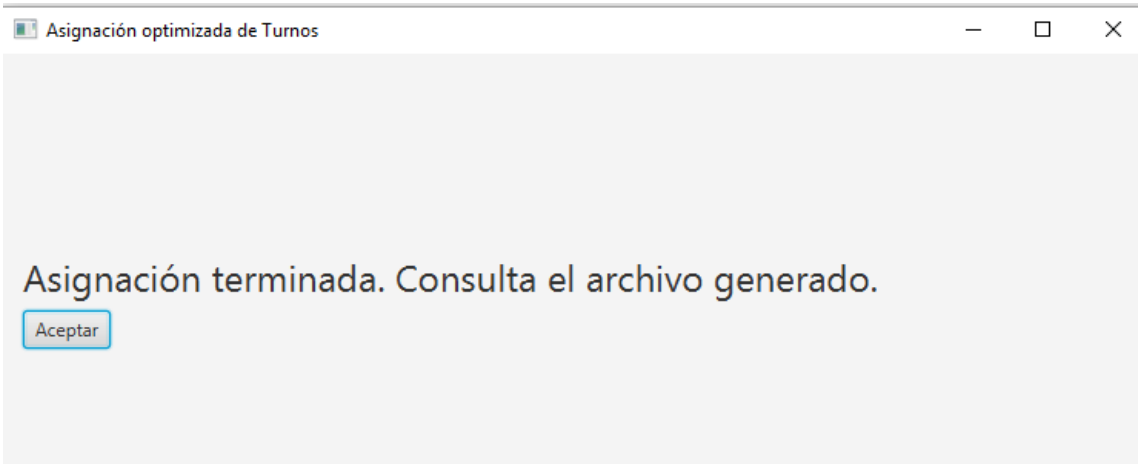

Imagen 14. Proceso de asignación terminado correctamente.

2) Editor de modelos: esta sección de la interfaz se alcanza cuando el usuario pulsa el botón editar en la ventana principal. Cuando esto ocurre, se lanza la interfaz que se muestra en la imagen 15 y en ella se cargan los datos del modelo que estuviera elegido en el selector de modelos en el momento en el que el usuario pulsara el botón de edición.

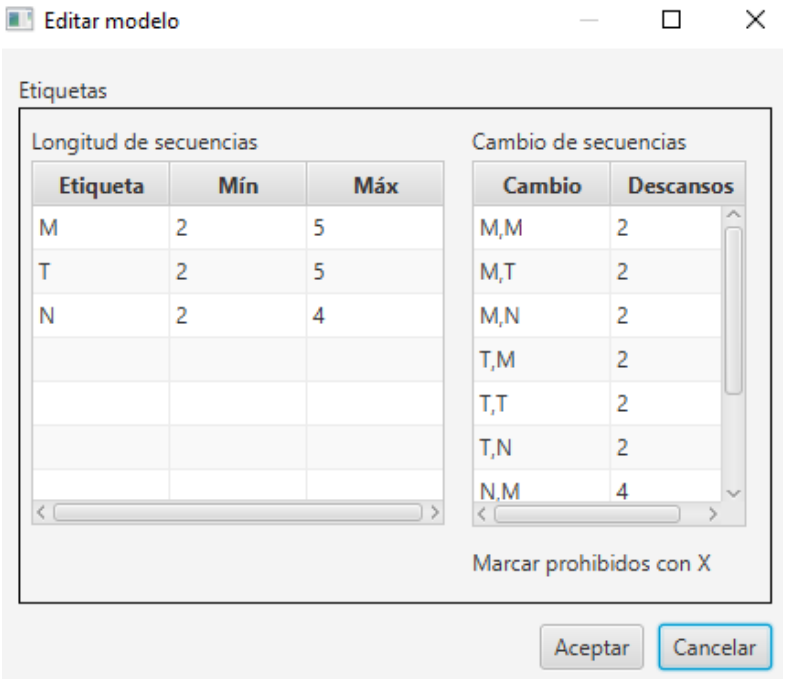

Imagen 15. Editor de modelos.

- **Tabla de longitud de secuencias:** en esta tabla se le permite al usuario introducir la restricción general referente a las longitudes de secuencia de turnos. Será válido cualquier número entero positivo siempre y cuando se cumpla que el máximo es mayor o igual que el mínimo y que ambos sean valores positivos.

- **Tabla de cambios de secuencia:** esta tabla tiene como finalidad la introducción de los datos relativos a la restricción general de descansos entre cambios de secuencia. Si algún cambio quisiera prohibirse, puede marcarse la casilla con el símbolo "X" o "x". Con esto el sistema entenderá que ese cambio no puede producirse y en la solución otorgada no existirá el cambio prohibido. Por último, señalar que no se puede poner cero entre secuencias del mismo turno, pues conceptualmente eso es una secuencia de longitud infinita.

- **Botón aceptar:** guarda los cambios en la base de datos y regresa a la ventana principal.

- **Botón cancelar:** regresa a la ventana principal sin guardar los cambios realizados.

3) Añadir modelos: es la parte de la interfaz utilizada para crear nuevos modelos o problemas y almacenarnos de manera persistente en la base de datos. Esta sección es alcanzable desde el botón nuevo modelo. Su aspecto se muestra en la imagen 16.

- **Nombre:** el modelo debe tener un nombre, que se introduce en este campo de texto. El nombre se introduce en la base de datos y es el que posteriormente se mostrará en la herramienta para seleccionar modelos.

- **Etiquetas:** en este campo de texto se introducen las etiquetas del modelo, separadas por coma. Al hacerlo, el sistema introduce de forma dinámica la primera columna de las tablas etiquetas y cambios de secuencia.

- **Tablas de longitud y cambio de secuencias:** tienen un comportamiento, funcionalidad y finalidad idénticos a los descritos en el editor de modelos, pues son, de hecho, los mismos elementos.

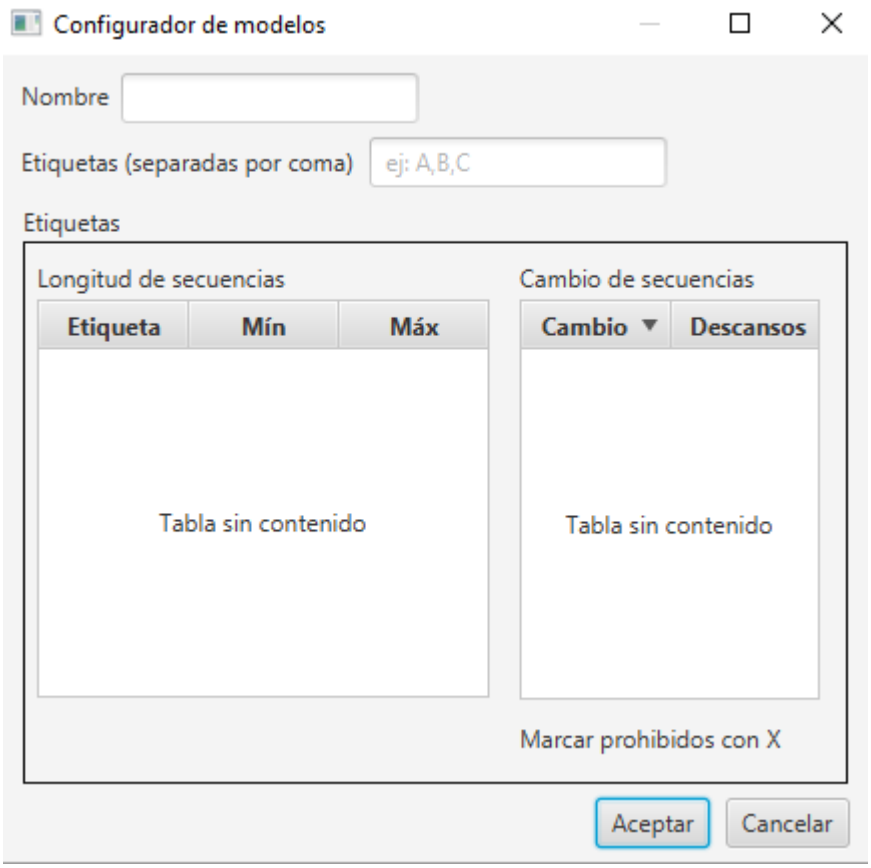

Imagen 16. Interfaz de adición de modelos

4) Nuevo día festivo: esta sencilla interfaz mostrada en la imagen 17 sólo tiene como finalidad la selección de un día que será marcado como festivo en el calendario. Puede alcanzarse desde el botón añadir nuevo día festivo que se encuentra en la ventana principal.

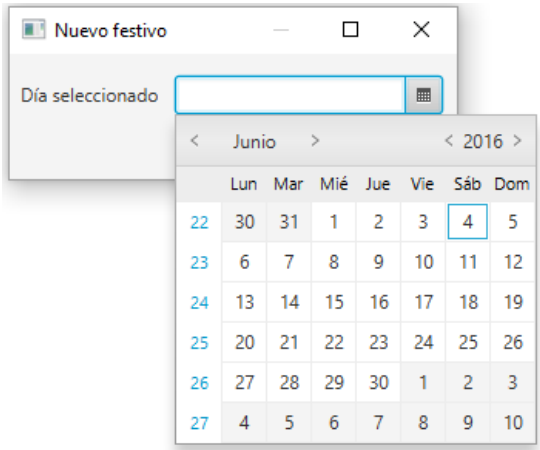

Imagen 17. Selector de días festivos.

5) Nuevo día especial: de forma similar a la interfaz anterior, ésta tiene como finalidad la inserción en el modelo de un día especial (descritos en este mismo punto, en la sección 1)Ventana principal). Como añadido al anterior, dispone de una tabla en la que se debe especificar la demanda que se tiene de cada turno en el día seleccionado. Su aspecto se muestra en la imagen 18.

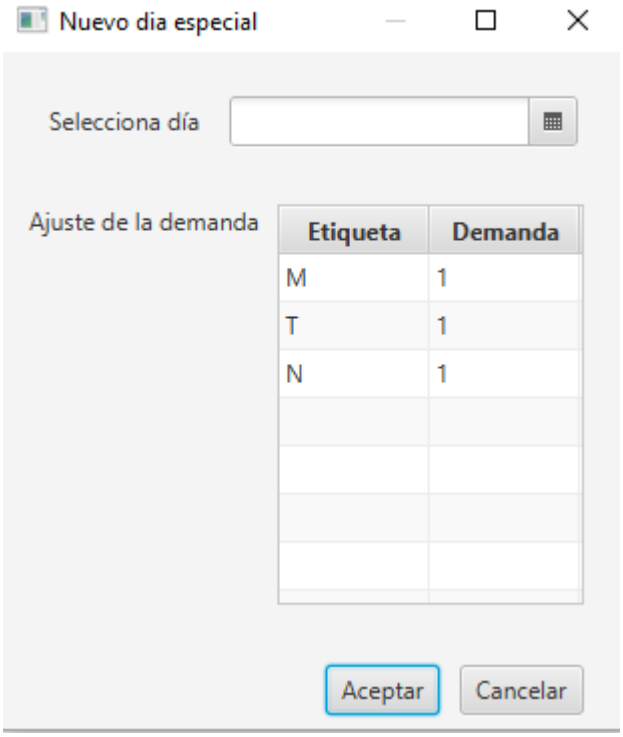

Imagen 18. Selector de días especiales

6) Trabajadores: esta sección de la interfaz, mostrada en la imagen 19, contiene la implementación de la funcionalidad extra que permite el bloqueo de la asignación de algunos turnos a los trabajadores especificados (descrita en el punto 3.4), así como la asignación de nombres a cada trabajador. Las etiquetas prohibidas deben indicarse separadas por coma. Los nombres de los trabajadores no influyen en el algoritmo de asignación y sólo son utilizados para la muestra de datos final al usuario. Si el usuario no especifica nombres, por defecto, los nombres se numeran automáticamente.

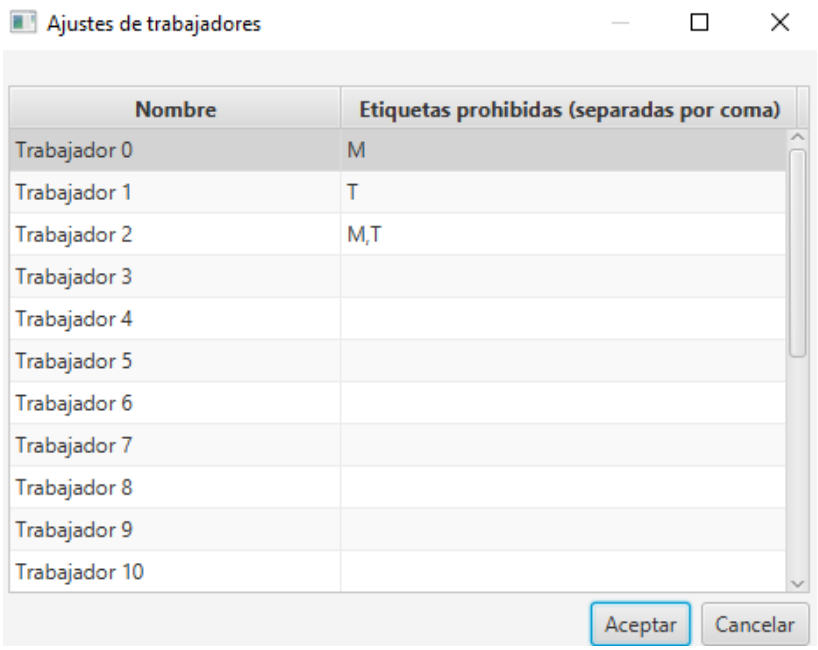

Imagen 20. Configuración de trabajadores.

4.3 Presentación de las soluciones

Cuando el proceso de búsqueda termina y encuentra una solución, escribe un fichero en disco en formato .xlsx (hoja de cálculo libre de versiones) con el nombre que haya especificado el usuario.

La solución presentada tiene distintos apartados que detallaremos a continuación. En primer lugar, el cuerpo de la solución, que contiene una parte en la que se representa el horizonte de asignación completo (generalmente, un año completo) con los turnos que se le asignan a cada trabajador cada día. Además también representa cómo ha quedado la demanda de cada turno tras realizar la asignación de cada día concreto, en la que, un número mayor que cero indica que faltan por cubrir tantos turnos como indique dicho número, un número negativo indica que hay un exceso de cubrimiento de ese mismo número de turnos y por último, un número igual a cero indica que la demanda ha sido cubierta a la perfección. También se incluye el número total de turnos que han quedado sin cubrir y el porcentaje de error que esto supone con respecto a la totalidad de turnos requeridos por la demanda. Esta situación se muestra en la imagen 21, que es un extracto de una solución real de los primeros quince días del año.

-	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene	Ene
Sin cubrir=3->0.07%	1 L	2 M	3 X	4 J	5 V	6 S	7 D	8 L	9 M	10 X	11 J	12 V	13 S	14 D	15 L
Trabajador 0	-	-	M	M	-	-	-	T	T	-	-	-	M	M	-
Trabajador 1	-	M	M	-	-	-	-	M	M	M	-	-	N	N	N
Trabajador 2	-	-	T	T	-	-	N	N	N	N	-	-	T	T	T
Trabajador 3	-	-	-	-	T	T	-	-	M	M	-	-	M	M	M
Trabajador 4	-	-	-	-	T	T	-	-	N	N	N	N	-	-	T
Trabajador 5	M	M	-	-	M	M	-	-	T	T	T	T	T	-	-
M	0	0	-2	0	0	-2	0	0	-2	0	0	-1	0	-1	-1
T	0	0	-1	0	0	-2	0	0	-2	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	-1	0	1	0	0	-1	0

Imagen 21. Extracto del cuerpo de la solución.

En segundo lugar, al final del cuerpo de la asignación se incluye un resumen de los turnos que ha realizado cada trabajador, realizando una clasificación por etiquetas y la suma del total de turnos. Este resumen se muestra en la imagen 22.

Total	M	T	N
195	90	67	38
194	85	68	41
194	99	57	38
194	107	57	30
194	81	89	24
194	57	94	43

Imagen 22. Resumen completo de turnos realizados.

En tercer lugar, y por último, si el usuario ha activado la restricción que exige que cada trabajador tenga, al menos, un fin de semana libre al mes, el sistema también añade en el fichero de salida una tabla en la que indica para cada par trabajador-mes si se ha podido cumplir esta restricción. En caso afirmativo, se marca la posición con el símbolo “X”, en caso contrario la casilla queda vacía indicando que no se ha podido cubrir el fin de semana libre de ese mes para el trabajador asociado. Esto se puede observar en la imagen 23 extraída también de una solución real.

T	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
0	X	X	X	X	X	X	X	X	X	X	X	X
1	X	X	X	X	X	X	X		X	X	X	X
2	X	X	X	X	X	X	X	X	X	X	X	X
3	X	X	X	X	X	X	X	X	X	X	X	X
4	X	X	X	X	X	X	X	X	X	X	X	X
5	X	X	X	X	X	X	X	X	X	X	X	X

Imagen 23. Resumen de fines de semana libres de cada trabajador.

5 Base de datos

La base de datos desarrollada en este proyecto se utiliza con la única finalidad de almacenar los datos de la interfaz para que el usuario pueda almacenar sus modelos y sus cambios en ellos sin necesidad de tener que repetir la tediosa tarea de introducción de datos cada vez que arranca la interfaz. La base de datos y su integración con la interfaz gráfica de usuario ha sido desarrollada en su totalidad gracias a la tecnología SQLite.

5.1 ¿Qué es SQLite?

SQLite es una tecnología que ofrece un sistema de gestión de base de datos compatible con operaciones que respetan ACID (características básicas de un sistema de gestión de bases de datos, a saber: atomicidad, consistencia, aislamiento y durabilidad). El sistema ofrecido está contenido en una pequeña biblioteca escrita en C, de apenas 275 kB, y es un proyecto de software libre que ha sido mejorado por la comunidad de usuarios de forma constante.

A diferencia de otros sistemas de gestión de bases de datos, SQLite no precisa de un proceso que esté ejecutándose en la máquina con el cual la aplicación debe comunicarse para realizar las operaciones adecuadas de consulta, inserción, actualización o borrado, sino que se integra dentro de la aplicación como una parte más de la misma, reduciendo así la latencia de acceso.

Las bases de datos SQLite se representan mediante un único fichero cifrado que contiene toda la especificación y los datos introducidos por el usuario, por lo que consultar la base de datos es equivalente a realizar una lectura sobre ese fichero. Además, el fichero queda bloqueado para cualquier operación una vez comienza una transacción, lo que permite ejecuciones concurrentes con una única base de datos común sin que esto suponga un riesgo para el sistema.

Por último añadiremos que SQLite utiliza un sistema de tipado fuera de lo común, pues permite que una misma columna contenga elementos de varios tipos. Aunque se puede especificar el tipo de una columna completa, SQLite permite que se introduzca de forma individual datos de tipo distinto al especificado, realizando él internamente todas las conversiones necesarias para el correcto funcionamiento del sistema.

5.2 Base de datos implementada

En la imagen 20 se presenta el diagrama UML de la base de datos creada y utilizada para el soporte del almacenamiento persistente de los datos introducidos en la aplicación mediante la interfaz gráfica de usuario. La base de datos que se expone implementa los requerimientos especificados en la siguiente descripción.

La base de datos se utilizará para almacenar modelos (problemas). Un modelo tiene un número especificado de trabajadores, unos turnos máximos anuales, un nombre, dos strings que representen las restricciones relativas a empezar turnos y a prohibirlos antes de los fines de semana y 2 enteros que representan el peso de ponderación de los criterios de optimización durante la búsqueda.

Un modelo está compuesto por un conjunto de trabajadores, cada uno con su respectivo nombre y con un conjunto de etiquetas que representan los turnos prohibidos para ese trabajador. Un trabajador sólo puede estar en un modelo. También consta de un conjunto de cambios que, a su vez, están compuestos cada uno por 2 etiquetas distintas. Las etiquetas deben tener cada una un nombre, así como los cambios entre ellas deben tener un valor numérico que representará los descansos necesarios entre ellas.

Los modelos también contienen una demanda especificada, que debe tener una fecha, un valor indicando a cuánto asciende la demanda, una etiqueta que indique qué recursos se necesitan cubrir en esa demanda y un entero que representa el tipo de demanda, a saber: festivo, laboral o especial.

Por último, se necesita también almacenar, para cada secuencia posible de turnos, sus longitudes máxima y mínima permitidas por las restricciones generales del problema. Cada secuencia debe estar formada por exactamente una etiqueta.

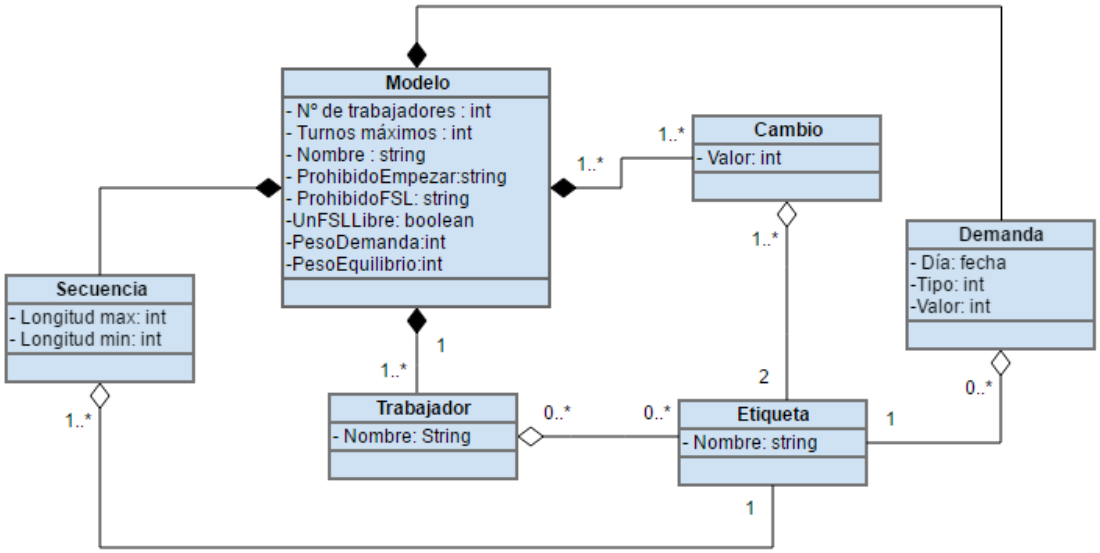

Imagen 24. Diagrama UML de la base de datos implementada.

6 Detalles de implementación

Por último detallaremos con qué tecnologías se han implementado cada parte de este proyecto y cómo se han realizado las comunicaciones entre ellas. Por lo que respecta a la interfaz gráfica de usuario y a la base de datos, como ya se ha indicado, se han implementado en JavaFX y SQLite respectivamente. Por su parte, la implementación del algoritmo de búsqueda, los posteriores postprocesos y la generación de datos de salida se han realizado utilizando Python.

Las comunicaciones entre SQLite y JavaFX se han realizado gracias a la librería de integración que proporciona SQLite y utilizando Java como lenguaje de programación de las llamadas a la base de datos, haciendo uso, a su vez, de objetos de tipo *Data Access Layer* (DAL) en la que se recogen todas las llamadas a la base de datos y siguiendo un patrón *singleton* para optimizar recursos en cuanto a memoria y consistencia de datos. A su vez, como es común en la

implementación de este tipo de software, dentro del DAL se encuentran los distintos *Data Access Object* (DAO) que representan mediante código cada uno de los objetos que se pretenden almacenar en la base de datos y que realizan las llamadas de cada uno de sus respectivos recursos.

La comunicación entre la interfaz, en JavaFX, y el proceso de asignación, en Python, se ha resuelto mediante la ejecución de comandos de la consola de comandos mediante Java. En concreto, el proceso Java realiza una escritura en el disco duro de un fichero tipo *json* que contiene todos los parámetros que haya introducido el usuario mediante la interfaz y que son necesarios para la resolución del problema. Una vez escrito el fichero, se realiza la ejecución del comando de consola que ejecuta el *script* en Python que resuelve el problema, cuya primera acción es leer el fichero en formato *json* que escribió la interfaz. Al leerlo, carga todos los parámetros en el sistema y comienza el proceso de búsqueda. Por último, cuando el proceso en Python termina, escribe en disco la solución y devuelve el control de la aplicación a Java, que se encargará de notificar al usuario la correcta finalización del proceso de búsqueda.

7 Análisis de resultados

En este apartado realizaremos un estudio de las soluciones que da el sistema a problemas reales que especificaremos y nos detendremos a hacer un análisis del comportamiento del algoritmo en función de la precisión en la búsqueda y en los postprocesos que haya elegido el usuario.

7.1 Resultados a problemas reales

Para probar el sistema y si sus soluciones son lo suficientemente viables como para que el sistema pudiera implantarse en un entorno real, nos disponemos a analizar los resultados obtenidos de la resolución de distintos problemas.

En concreto, analizaremos cómo se comporta el sistema en 2 entornos reales. En primer lugar, aplicaremos el método de resolución al problema de asignación que tiene lugar en la empresa encargada de la gestión de la policía portuaria de un puerto marítimo español, la cual nos ha facilitado la información necesaria para la modelización y resolución del modelo que tienen implantado en su entorno. La especificación del problema se presenta a continuación:

- T=74
- N=365 (un año completo)
- E={M,T,N,-}
- D=

Etiqueta	Laborables	Festivos
M	15	9
T	14	9
N	9	9

Tabla 6. Demanda del problema de policía portuaria.

- R:
  - Restricciones generales: Utilizaremos una representación tabular de las mismas.
    - Longitudes del tamaño de secuencia:

Etiqueta	Máximo	Mínimo
M	5	2
T	5	2
N	4	2

Tabla 7. Longitudes de secuencia del problema de policía portuaria.

    - Turnos totales máximos = 194
    - Descansos obligatorios entre secuencias

Cambio	M-M	M-T	M-N	T-M	T-T	T-N	N-M	N-T	N-N
Descansos	2	2	2	2	2	2	4	2	Prohibido

Tabla 8. Cambios de secuencia en el problema de policía portuaria.

- Restricciones especiales (recordemos que se expresan en lenguaje natural):
    - Restricción 1: Prohibido empezar secuencia en Domingo.
    - Restricción 2: Para que un fin de semana libre sea considerado como tal, no se permite que el viernes inmediatamente anterior se haya realizado un turno etiquetado con N.
    - Restricción 3: Se exige un fin de semana libre al mes.

Las restricciones generales del problema provocan que el modelo tenga una carga total del 91%. Se pretende lograr el máximo equilibrio posible turnos de noche, que suelen resultar los más molestos a los trabajadores. Tras proporcionar como entrada al sistema el problema indicado y ejecutarlo, se han obtenido los siguientes resultados extraídos directamente de la solución proporcionada por el algoritmo de resolución:

- **Tiempo total de cómputo:** 40.72 segundos.
- **Porcentaje de cubrimiento:** El 97.4% de los turnos se han cubierto.
- **Porcentaje de fines de semana libres cubiertos:** El 99% se han respetado.
- **Equilibrio de turnos de noche:** Las noches realizadas por cada trabajador presentan una desviación típica de 8.83.

Como se observa, en términos de equilibrio, las soluciones podrían mejorar notablemente, ya que en turnos de noches se observan notables desequilibrios, lo que indica que existe un conjunto pequeño de trabajadores que está asumiendo un mayor porcentaje de la carga de trabajo. Sin embargo, la solución otorgada está cerca de la perfección en lo que a turnos cubiertos se refiere, por lo que nos planteamos si quizá sean las restricciones especiales del problema las que imposibilitan que la solución perfecta se alcance. Para comprobarlo, realizamos una ejecución del mismo problema pero eliminando sus restricciones especiales, creando así un problema similar al anterior pero más relajado. Los resultados del problema han sido:

- **Tiempo total de cómputo:** 27.32 segundos.
- **Porcentaje de cubrimiento:** El 100% de los turnos se han cubierto.
- **Equilibrio de turnos de noche:** Las noches realizadas por cada trabajador presentan una desviación típica de 8.94.

En estos nuevos datos no se observan mejoras considerables en cuanto a equilibrio de noches se refiere. Por último, sí que se muestra cómo el porcentaje de cubrimiento ha ascendido hasta el 100%. Esto evidencia que las restricciones especiales de cada entorno pueden provocar que los modelos asociados sean especialmente restrictivos, dificultando así la búsqueda de las soluciones óptimas al problema.

Como segundo entorno de pruebas utilizaremos un caso realista aplicable en entornos sanitarios en el que se pretende que los trabajadores alternen sus días de trabajo entre los distintos lugares que requieran sus servicios dentro de la entidad: sala, ambulatorio, guardia, general y quirófano. La especificación del problema es la siguiente:

- T=17
- N=365 (un año completo)
- E={S,A,G,B,Q}
- D=

Etiqueta	Laboral	Festivo
S	4	0
A	3	0
G	0	2
B	3	0
Q	2	0

Tabla 9. Demanda de el problema de entorno sanitario.

- R:
  - Restricciones generales:
    - Longitudes del tamaño de secuencia:

Etiqueta	Tamaño máximo	Tamaño mínimo
S	1	1
A	1	1
G	1	1
B	1	1
Q	1	1

Tabla 10.Longitudes de secuencia del problema de entorno sanitario.

    - Turnos totales máximos = 225

    - Descansos obligatorios entre secuencias: Para evitar una representación tabular excesivamente grande, las indicaremos mediante lenguaje natural. Todos los descansos entre secuencias son cero exceptuando 2 casos: Los cambios entre 2 secuencias iguales están prohibidos y después de una G siempre se debe hacer, al menos, 1 descanso.

Las restricciones generales del problema provocan que el modelo tenga una carga total del 98%. Se pretende conseguir el máximo equilibrio posible entre las guardias que realiza cada trabajador. Tras proporcionar como entrada al sistema el problema indicado y ejecutarlo, se han obtenido los siguientes resultados extraídos directamente de la solución proporcionada por el algoritmo de resolución (que se adjunta en el Anexo 3):

- **Tiempo total de cómputo:** 66.31 segundos.
- **Porcentaje de cubrimiento:** El 99.84% de los turnos se han cubierto.
- **Equilibrio de guardias:** El número de guardias por trabajador presentan una desviación típica de 2.22.

Como se puede observar, pese a que el modelo está muy sobrecargado, aproximándose al 100% de la carga, se ha conseguido una solución muy cercana a la óptima en un tiempo cercano al minuto de cómputo sobrepasándolo por poco más de 6 segundos. También se aprecia cómo el equilibrio entre guardias se podría mejorar notablemente, pues se pretende que el número de guardias sea lo más equilibrado posible.

Tanto en este problema como en los 2 modelos anteriores, se han conseguido soluciones muy cercanas a la óptima en tiempos muy reducidos, pese a que los problemas son de naturalezas y entornos muy distintos, lo que puede indicar que el problema se adapta considerablemente bien a entornos dispares y a distintas exigencias de demandas.

Por último, se han realizado pruebas en las que se prioriza, mediante los criterios de optimización, el equilibrio entre los turnos que realizan los trabajadores frente al porcentaje de cubrimiento de demanda, intentado conseguir así soluciones más aplicables a un entorno real.

En concreto, en el problema del entorno sanitario, se espera obtener un equilibrio en cuanto a las guardias que realiza cada trabajador del problema, por ser considerado el turno más incómodo. Para satisfacer esta situación, se le ha dado un peso máximo al equilibrio y un peso nulo al cubrimiento de demanda. Los resultados han sido:

- **Tiempo total de cómputo:** 57.11 segundos.
- **Porcentaje de cubrimiento:** El 98.12% de los turnos se han cubierto.
- **Equilibrio de guardias:** El número de guardias por trabajador presentan una desviación típica de 1.54.

Por otro lado, en el problema del puerto, se desea que el equilibrio se encuentre entre los turnos de noche que realiza cada trabajador. Tras darle todo el peso al equilibrio de estos turnos, los resultados han sido:

- **Tiempo total de cómputo:** 51.26 segundos.
- **Porcentaje de cubrimiento:** El 94.83% de los turnos se han cubierto.
- **Equilibrio de turnos de noche:** El número de noches por trabajador presentan una desviación típica de 4.15.

A raíz de los resultados, en los dos casos, podemos afirmar que darle un peso mayor a los equilibrios es un método efectivo para que las soluciones sean más equitativas en este aspecto. Sin embargo, esto se hace a costa de perder precisión en el porcentaje de cubrimiento, por lo que se evidencia una vez más la dificultad de encontrar soluciones equilibradas en todos los aspectos. Para

ilustrar esta situación se han realizado pruebas que muestran cómo evolucionan los resultados obtenidos a medida que cambian los parámetros de peso entre demanda y equilibrios.

Estas pruebas se han realizado dejando un peso constante e igual a cinco al cubrimiento de la demanda y realizando sucesivas pruebas con variaciones en el peso de los equilibrios con todos los valores enteros comprendidos entre cero y diez. Las pruebas se han realizado sobre el modelo de entorno sanitario, pero los resultados, que se muestran a continuación, son generalizables:

Peso Equilibrio	Turnos sin cubrir	Desviación típica
0	5	2,27
1	5	2,21
2	8	2,097
3	9	1,91
4	9	1,86
5	11	1,53
6	11	1,47
7	11	1,36
8	14	1,02
9	14	0,98
10	16	0,89

Tabla 11. Evolución de resultados.

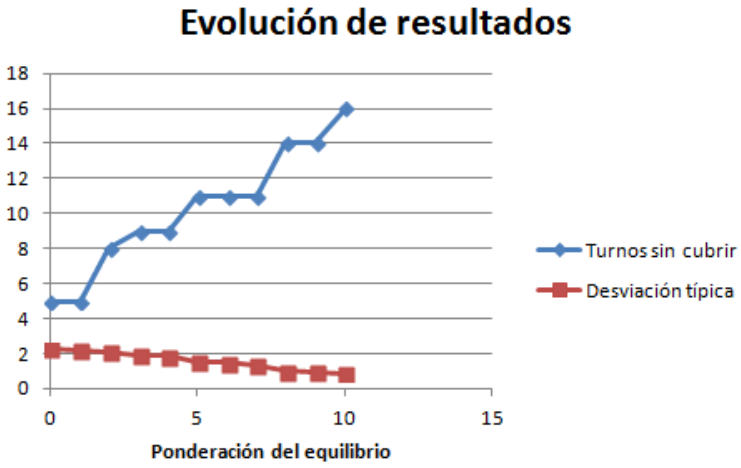

Imagen 25. Evolución de resultados.

7.2 Análisis del comportamiento del algoritmo

A continuación realizaremos un estudio de cómo se comporta el algoritmo en los 3 entornos estudiados anteriormente, tratando así de justificar la necesidad de los postprocesos de arreglo implementados en la solución para lograr una óptima solución al problema.

En las tablas 9, 10 y 11 y en las imágenes 24, 25 y 26 se puede observar, tanto tabular como gráficamente, cómo se comporta el proceso de búsqueda si vamos realizando aumentos en el número de iteraciones que se realizan en la misma analizando cómo evoluciona el error a medida que las iteraciones aumentan.

Búsqueda en puerto relajado	
iteraciones	demanda sin cubrir
1	346
100	242
200	242
300	236
400	233
500	230
600	230
700	230
800	230
900	230
1000	230
1100	222

Tabla 12. Búsqueda en Puerto relajado.

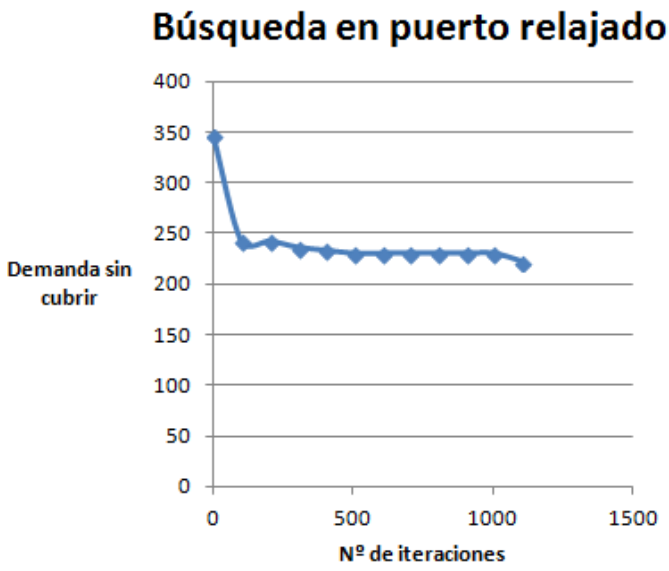

Imagen 26. Búsqueda en puerto relajado.

Búsqueda en puerto	
iteraciones	demanda sin cubrir
1	277
100	229
200	227
300	227
400	227
500	227
600	227
700	226
800	226
900	226
1000	221
1100	221

Tabla 13. Búsqueda en puerto.

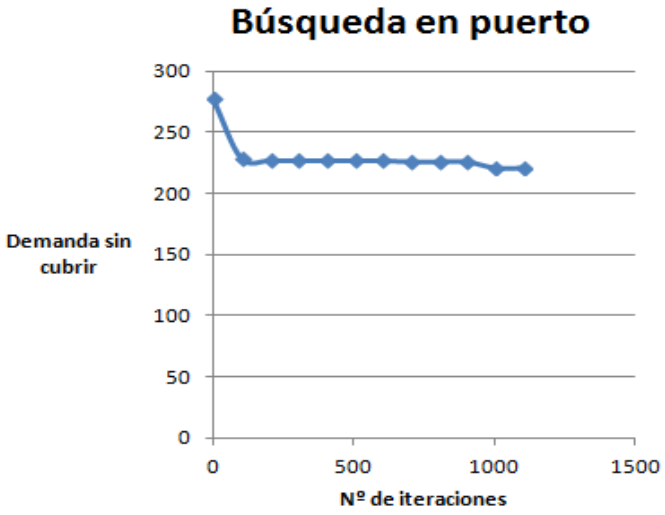

Imagen 27. Búsqueda en puerto.

Búsqueda en entorno sanitario	
iteraciones	demanda sin cubrir
1	53
100	27
200	24
300	13
400	13
500	9
600	7
700	7
800	7
900	7
1000	7
1100	7

Tabla 14. Búsqueda en entorno sanitario.

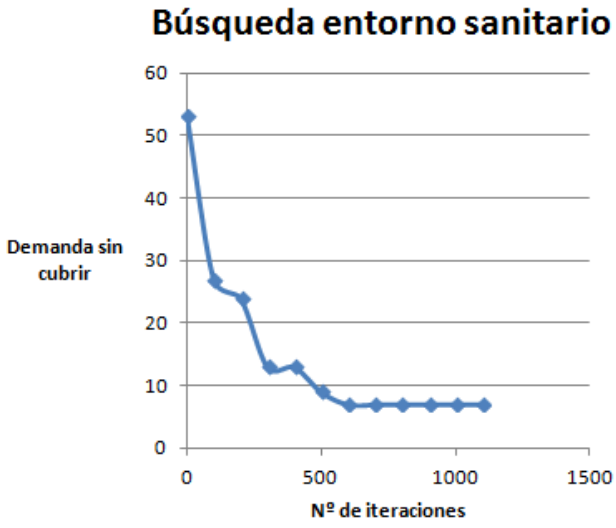

Imagen 28. Búsqueda en entorno sanitario.

A raíz de estos resultados se puede afirmar que el aumento del número de iteraciones que se realiza en la búsqueda mejora progresivamente la corrección de las soluciones que se obtienen. Además, también puede deducirse que las primeras iteraciones ya logran unas soluciones que cubren un alto porcentaje de la demanda y que las posteriores iteraciones logran mejorar lentamente las soluciones obtenidas. Además, como se puede observar, la búsqueda deja aún mucha demanda por cubrir en los 2 primeros casos. La aplicación de los postprocesos pretende disminuir este error, tratando que el porcentaje de cubrimiento de demanda mejore. Para que se observe cómo evolucionan las soluciones en la aplicación de los procesos posteriores de arreglo, se exponen los resultados obtenidos del análisis de progresión de las mismas.

Postprocesos en puerto relajado	
iteraciones	demanda sin cubrir
0	222
100	5
200	5
300	5
400	5
500	5
600	5
700	5
800	3
900	0
1000	0
1100	0

Tabla 15. Postprocesos en puerto relajado.

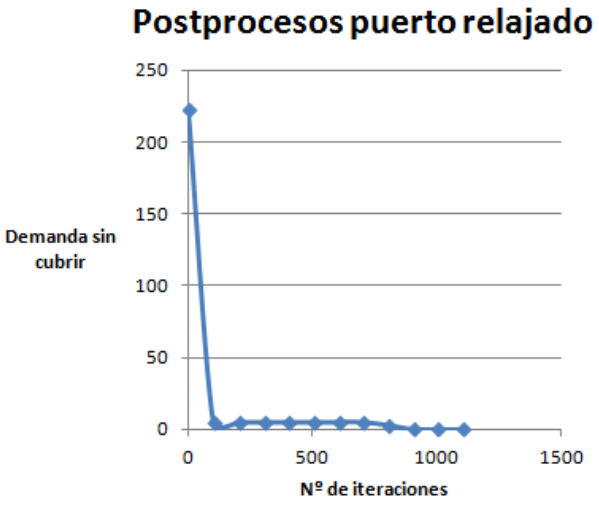

Imagen 29. Postprocesos en puerto relajado.

Postprocesos en puerto	
iteraciones	demanda sin cubrir
0	221
100	59
200	59
300	59
400	59
500	59
600	59
700	59
800	59
900	59
1000	59
1100	59

Tabla 16. Postprocesos en puerto.

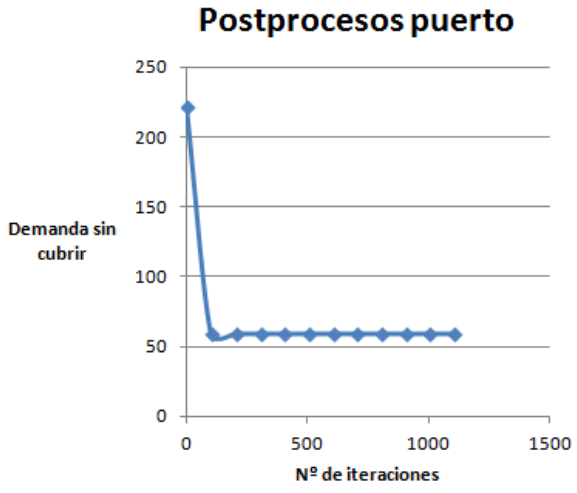

Imagen 30. Postprocesos en puerto.

Postprocesos en entorno sanitario	
iteraciones	demanda sin cubrir
0	7
100	5
200	5
300	5
400	5
500	5
600	4
700	4
800	3
900	3
1000	3
1100	3

Tabla 17. Postprocesos en entorno sanitario.

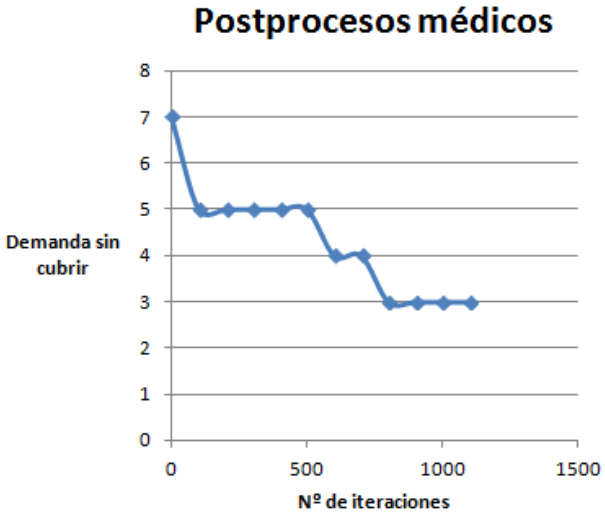

Imagen 31. Postprocesos en entorno sanitario.

Estos resultados muestran cómo la aplicación iterativa de los procesos posteriores de arreglo consigue mejorar las soluciones iniciales que genera el proceso inicial de búsqueda. Además, se aprecia como los postprocesos se adaptan a entornos muy dispares, como lo son los 3 expuestos, logrando dar buenas soluciones en todos ellos, a pesar de que la situación de partida que ofrece el proceso inicial de búsqueda es muy distinta en cada uno de ellos.

Señalar que en todos los casos se han logrado, tras la ejecución del algoritmo completo, soluciones cercanas al 0% de error final y con un tiempo de cómputo total cercano al minuto.

8 Conclusiones

El problema de la asignación óptima de turnos es ampliamente conocido y utilizado en entornos empresariales que cuentan con grandes plantillas de empleados donde la optimización del número de trabajadores a financiar, así como su planificación correcta, pueden suponer cuantías económicas ingentes para las entidades que deban de correr con estos grandes gastos.

A pesar de ello, no existe un software fácilmente adaptable a un alto número de entornos distintos, provocando que las soluciones informáticas que las empresas reciben a este conflicto sean confeccionadas a su medida, con los altos costes asociados que esto supone, tanto económicos como temporales.

En este contexto, nos planteábamos los objetivos del proyecto (descritos en el punto 1.3) con respecto a adaptabilidad, corrección de soluciones y tiempos de cómputo aceptables para la resolución de un problema con estas características.

Si analizamos los objetivos que nos planteábamos al inicio del proyecto, podemos observar que todos han sido superados con buenos resultados, que dejan lugar a posibles futuras mejoras que logren perfeccionar más aún las soluciones requeridas a estos problemas.

Por lo que respecta al 90% de tasa de optimalidad que nos planteábamos con el primer objetivo, podemos asegurar, como se ha evidenciado en el punto 7.1, que se ha superado con creces, llegando incluso a soluciones con un porcentaje de acierto muy cercano o incluso igual al 100%.

También nos planteábamos como segundo objetivo, lograr que todas las soluciones obtenidas se consiguieran en menos de treinta minutos de cómputo total. En este aspecto, los resultados han sido más que satisfactorios, pues hemos logrado soluciones cercanas al 0% de error en tiempos cercanos a un minuto de cómputo, pudiendo asegurar así que este objetivo ha sido correctamente alcanzado.

En el tercer objetivo enfatizábamos el hecho de que la tecnología ofrecida debe ser generalizable y adaptable a distintos entornos. Con respecto a este punto, podemos afirmar que, a pesar de que los entornos estudiados en este proyecto eran de naturalezas muy distintas, el sistema implementado ha conseguido en todos los casos dar buenas soluciones a ellos, lo que sugiere que la adaptabilidad del sistema es alta y que puede aplicarse a problemas de características conceptualmente distintas sin sacrificar precisión en las soluciones, siempre que el modelado sea correcto.

En los objetivos cuarto y quinto, se planteaba la creación, tanto de una interfaz gráfica como de una base de datos que permitiera adaptabilidad, portabilidad y facilidad de uso al sistema. A lo largo del proyecto (en los puntos 4 y 5) se ha mostrado el trabajo realizado en torno a este aspecto, lo que nos permite también afirmar que, a raíz de lo expuesto, estos objetivos se han alcanzado también satisfactoriamente.

Por último, señalar que, pese a la dificultad de medición de este aspecto, se puede afirmar que el sistema otorgado como solución es fácilmente adaptable a nuevas mejoras y ofrece un sencillo mantenimiento, pues durante la realización del proyecto se le han añadido múltiples y distintas funcionalidades sin excesivo coste temporal adicional. Esto deja abierto todo un conjunto de mejoras que pueden ser introducidas en el sistema sin que éste se vea afectado en exceso por los cambios necesarios para hacerles frente.

9 Bibliografía

[1] A. Ruescas Nicolau. Trabajo final de máster: Aplicación de técnicas metaheurísticas para la asignación de turnos de trabajo, Julio 2015.

[2] É. Naudin, P.Y.C. Chan, M. Hiroux, T. Zemmouri, G. Weil. Analysis of three mathematical models of the Staff Rostering Problem, Diciembre 2009.

[3] Haroldo G. Santos, Túlio A. M. Toffolo, Rafael A. M. Gomes, Sabir Ribas. Integer programming techniques for the nurse rostering problem, Mayo 2014.

[4] Martina Seidl, Marion Scholz Christian Huemer, Gerti Kappel. UML @ Classroom, Enero 2015.

[5] L. Doug, B. Barry. JavaFX For Dummies, Septiembre 2014.

[6] G.Allen, M.Owens. The Definitive Guide to SQLite, Enero 2010.

[7] M. Resende, J.L.González. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19.

[8] T.A. Feo, M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. JournalofGlobal Optimization6, 1995.

[9] PATAT 2010. First International Nurse Rostering Competition, recuperado en junio de 2016 de: https://www.kuleuven-kulak.be/nrpcompetition

[10] INRC-II. Second International Nurse Rostering Competition, recuperado en junio de 2016 de: http://mobiz.vives.be/inrc2/

10 Anexos

Anexo 1) Extracto de la solución del problema del entorno sanitario

	Sin cubrir=7 ->0,16%															Enero															
	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
T1	-	S	-	-	A	-	G	-	-	S	Q	S	B	-	S	-	-	-	A	B	-	S	A	B	S	Q	A	G	-	S	B
T2	-	-	S	B	A	-	-	-	S	A	-	-	-	-	-	-	Q	A	S	-	-	-	-	-	-	-	-	-	-	A	S
T3	-	S	B	A	B	-	-	-	A	S	B	S	-	-	Q	S	A	S	-	-	G	-	-	-	S	A	S	-	B	Q	A
T4	-	A	S	Q	S	-	-	-	B	Q	S	B	S	-	A	Q	S	A	-	S	-	-	Q	A	Q	S	-	-	-	-	B
T5	G	-	B	A	-	-	-	S	A	B	A	B	S	-	B	A	S	Q	S	Q	-	A	S	B	S	B	S	-	A	S	A
T6	-	B	A	S	B	-	G	-	Q	B	A	B	A	-	Q	S	A	S	B	Q	G	-	S	Q	B	Q	S	G	-	-	S
T7	-	B	-	-	S	-	-	Q	S	Q	S	A	S	-	-	-	A	B	Q	B	-	S	B	A	S	A	-	-	B	Q	B
T8	G	-	-	-	S	-	-	A	S	-	-	-	Q	G	-	S	B	Q	S	A	-	B	-	S	A	B	A	-	-	-	S
T9	-	-	A	S	-	-	-	B	-	-	Q	A	Q	G	-	A	Q	B	A	S	-	-	-	A	-	-	B	-	S	-	-
T10	-	Q	B	S	Q	G	-	B	-	-	S	Q	B	-	S	B	-	S	B	A	-	S	A	S	-	-	B	-	Q	S	Q
T 10	-	A	-	-	S	G	-	S	-	-	A	S	A	-	B	Q	-	-	-	S	-	Q	A	S	B	S	-	-	A	B	Q
T 11	-	S	-	-	B	-	-	A	S	B	-	S	-	-	B	A	B	A	-	-	-	B	S	-	A	S	B	-	S	B	-
T 12	-	B	S	A	-	-	-	Q	B	A	B	-	A	-	S	B	S	B	S	-	-	A	S	-	Q	B	A	-	Q	A	S
T 13	-	Q	S	Q	-	-	-	A	Q	A	S	Q	-	-	S	-	B	S	B	A	-	Q	B	S	B	A	Q	-	S	A	-
T 14	-	A	Q	S	A	-	-	B	-	-	-	-	-	-	A	B	S	-	Q	B	-	S	Q	B	-	-	-	-	A	-	-
T 15	-	-	A	B	-	-	-	S	A	S	-	-	B	-	A	S	-	-	-	-	-	B	-	-	A	S	Q	-	S	B	-
T 16	-	S	Q	B	Q	-	-	S	B	S	B	A	S	-	-	-	-	-	A	S	-	A	B	Q	-	-	S	-	B	S	A
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Donde las etiquetas representan:

S: Turno de sala.

A: Turno de ambulatorio.

Q: Turno de quirófano.

G: Turno de guardia.

B: Turno general.

Anexo 2) Extracto de la solución del problema del puerto

	Sin cubrir=360->2,6												Ene																		
	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	T	-	-	M	M	M	-	-	-	M	M	FSL	FSL	N	N	-	-	T	T	-	-	T	T	-	-	M	M	-	-	-
T 1	N	N	-	-	T	T	T	-	-	-	T	T	-	-	M	M	M	M	-	FSL	FSL	M	M	M	-	-	M	M	M	M	M
T 2	N	N	-	-	-	-	-	-	T	T	T	T	FSL	FSL	-	-	-	M	M	M	-	-	-	-	N	N	-	-	T	T	T
T 3	-	-	T	T	T	-	-	-	M	M	-	-	M	M	M	M	M	-	-	-	-	M	M	M	-	-	FSL	FSL	N	N	N
T 4	-	-	T	T	-	-	-	T	T	T	-	-	M	M	M	M	-	-	-	FSL	FSL	-	-	-	N	N	N	N	-	-	T
T 5	N	N	-	-	T	T	T	T	T	-	-	-	-	-	T	T	T	T	T	-	-	M	M	M	M	M	FSL	FSL	M	M	M
T 6	M	M	-	-	-	-	-	-	-	-	T	T	T	-	-	M	M	M	-	FSL	FSL	T	T	T	-	-	T	T	T	T	T
T 7	-	-	N	N	-	FSL	FSL	-	-	-	T	T	T	-	-	N	N	N	N	-	-	-	T	T	T	T	T	-	-	-	M
T 8	M	M	M	M	M	-	-	N	N	-	-	-	T	T	T	T	-	-	-	-	-	M	M	M	M	M	FSL	FSL	-	-	T
T 9	-	-	-	-	N	N	N	-	-	T	T	-	FSL	FSL	T	T	-	-	N	N	N	N	-	-	-	-	M	M	M	M	-
T 10	T	T	-	-	T	T	T	-	-	-	T	T	FSL	FSL	-	-	M	M	M	M	M	-	-	-	-	-	M	M	M	M	M
T 11	-	-	-	-	-	-	-	-	N	N	N	N	-	-	T	T	T	T	T	FSL	FSL	N	N	N	-	-	T	T	-	-	-
T 12	-	-	T	T	T	-	-	M	M	-	-	-	M	M	-	-	-	-	-	FSL	FSL	-	M	M	M	M	M	-	-	M	M
T 13	N	N	N	N	-	FSL	FSL	T	T	T	-	-	T	T	-	-	-	-	-	T	T	T	-	-	T	T	T	T	T	-	-
T 14	-	-	M	M	-	-	-	M	M	-	-	T	T	T	-	-	T	T	-	-	-	T	T	T	T	T	FSL	FSL	N	N	-
T 15	M	M	-	-	N	N	N	-	-	-	T	T	-	-	-	-	T	T	-	FSL	FSL	-	M	M	M	M	M	-	-	T	T
T 16	-	-	T	T	T	-	-	T	T	-	-	N	N	N	N	-	-	T	T	FSL	FSL	-	-	-	M	M	M	-	-	M	M
T 17	T	T	T	T	T	-	-	-	-	-	N	N	N	N	-	-	T	T	-	FSL	FSL	-	N	N	-	-	T	T	T	T	T
T 18	M	M	M	M	M	-	-	N	N	N	-	-	T	T	-	-	M	M	M	M	M	-	-	-	-	-	FSL	FSL	M	M	M
T 19	-	-	N	N	-	-	-	T	T	T	-	-	M	M	M	M	M	-	-	T	T	-	-	T	T	-	FSL	FSL	N	N	N
T 20	T	T	-	-	T	T	T	T	T	-	-	M	M	M	-	-	T	T	T	-	-	M	M	M	-	-	FSL	FSL	T	T	T
T 21	T	T	-	-	M	M	M	-	-	M	M	-	-	-	M	M	M	M	-	-	-	T	T	T	T	T	FSL	FSL	-	-	-
T 22	-	-	-	-	T	T	T	T	T	-	-	N	N	N	-	-	T	T	T	-	-	M	M	M	-	-	FSL	FSL	T	T	-
T 23	-	T	T	T	-	-	-	T	T	T	T	T	FSL	FSL	-	-	-	-	-	T	T	T	T	-	-	-	M	M	M	M	-
T 24	-	M	M	M	M	-	-	N	N	N	N	-	FSL	FSL	-	-	T	T	T	T	-	-	N	N	N	N	-	-	T	T	T
T 25	M	M	-	-	N	N	N	-	-	-	T	T	-	-	N	N	N	N	-	FSL	FSL	T	T	-	-	N	N	N	N	-	-
T 26	M	M	-	-	-	-	-	-	M	M	M	M	M	-	-	N	N	N	N	-	-	T	T	T	T	T	FSL	FSL	T	T	T
T 27	-	-	-	-	-	-	-	N	N	N	N	-	-	-	-	M	M	M	-	FSL	FSL	N	N	-	-	-	T	T	-	-	M
T 28	-	-	N	N	N	N	-	-	-	-	T	T	T	T	-	-	M	M	M	M	M	-	-	M	M	-	FSL	FSL	-	-	M
T 29	T	T	T	T	-	-	-	-	M	M	-	-	M	M	M	M	M	-	-	T	T	-	-	-	-	-	FSL	FSL	-	-	T
T 30	-	-	T	T	-	-	-	M	M	-	-	-	FSL	FSL	-	-	N	N	N	N	-	-	T	T	T	T	T	-	-	T	T
T 31	-	-	-	-	-	FSL	FSL	-	-	-	M	M	M	M	M	-	-	T	T	T	T	T	-	-	-	-	N	N	N	N	-
T 32	-	-	-	-	-	-	-	T	T	T	T	-	-	-	-	-	T	T	-	-	-	N	N	N	N	-	FSL	FSL	-	-	M
T 33	M	M	M	M	-	-	-	-	N	N	-	-	FSL	FSL	T	T	-	-	M	M	-	-	-	-	M	M	M	M	-	-	N
T 34	-	T	T	-	-	-	-	T	T	T	-	-	N	N	N	N	-	-	T	T	T	T	-	-	-	-	FSL	FSL	N	N	N
T 35	-	-	N	N	N	N	-	-	T	T	T	T	T	-	-	M	M	M	M	M	-	-	M	M	-	-	N	N	N	N	-
T 36	-	-	-	-	N	N	N	-	-	T	T	T	-	-	M	M	M	M	-	FSL	FSL	T	T	-	-	M	M	-	-	-	N
T 37	N	N	-	-	-	-	-	T	T	T	-	-	M	M	-	-	T	T	-	FSL	FSL	-	T	T	-	-	N	N	N	-	-
T 38	-	M	M	-	-	-	-	-	-	-	-	-	M	M	M	M	M	-	-	N	N	N	N	-	-	-	FSL	FSL	M	M	M
T 39	-	-	-	-	-	FSL	FSL	M	M	M	-	-	M	M	-	-	-	-	M	M	-	-	-	-	M	M	M	M	M	-	-
T 40	-	-	M	M	M	FSL	FSL	M	M	-	-	N	N	N	N	-	-	T	T	T	T	T	-	-	N	N	N	N	-	-	-
T 41	-	-	M	M	M	-	-	M	M	M	M	M	-	-	M	M	M	M	-	-	-	-	N	N	-	-	FSL	FSL	M	M	M
T 42	N	N	N	N	-	FSL	FSL	T	T	-	-	-	T	T	T	T	-	-	N	N	N	N	-	-	T	T	T	T	-	-	M
T 43	-	-	-	-	-	-	-	N	N	N	N	-	FSL	FSL	-	-	-	-	M	M	-	-	N	N	-	-	T	T	-	-	N
T 44	T	T	T	T	T	-	-	M	M	M	M	M	-	-	-	-	-	-	-	M	M	-	-	T	T	-	FSL	FSL	-	-	T
T 45	-	-	-	-	M	M	M	M	-	-	T	T	FSL	FSL	M	M	-	-	M	M	M	-	-	T	T	T	T	T	-	-	M
T 46	-	-	-	-	N	N	N	-	-	T	T	-	-	-	-	-	M	M	-	FSL	FSL	-	M	M	M	M	M	-	-	T	T
T 47	N	N	-	-	T	T	T	T	-	-	M	M	-	-	T	T	-	-	T	T	T	T	T	-	-	-	FSL	FSL	N	N	-
T 48	-	-	-	-	M	M	M	M	M	-	-	-	-	-	N	N	N	N	-	-	-	-	T	T	-	-	FSL	FSL	-	-	-
T 49	-	T	T	-	-	-	-	-	-	-	N	N	N	N	-	-	T	T	-	FSL	FSL	-	-	-	T	T	T	T	T	-	-
T 50	-	M	M	M	-	-	-	-	-	-	-	-	M	M	M	M	-	-	-	FSL	FSL	M	M	-	-	M	M	M	M	M	-
T 51	N	N	-	-	-	-	-	-	M	M	-	-	T	T	T	T	-	-	-	-	-	N	N	N	-	-	FSL	FSL	T	T	-
T 52	-	M	M	M	M	-	-	M	M	M	M	M	-	-	T	T	-	-	-	-	-	M	M	M	M	M	FSL	FSL	T	T	-
T 53	-	-	T	T	-	FSL	FSL	N	N	N	-	-	T	T	T	T	-	-	-	N	N	N	N	-	-	-	T	T	-	-	N
T 54	-	-	-	-	-	-	-	M	M	M	M	M	-	-	M	M	-	-	N	N	N	N	-	-	T	T	FSL	FSL	M	M	M
T 55	T	T	T	T	T	-	-	T	T	T	T	T	FSL	FSL	N	N	-	-	T	T	T	-	-	N	N	N	N	-	-	-	-
T 56	M	M	M	M	-	-	-	N	N	-	-	T	T	T	-	-	N	N	-	-	-	T	T	T	T	T	FSL	FSL	T	T	-
T 57	-	-	N	N	-	-	-	-	-	-	-	-	-	-	T	T	T	T	-	FSL	FSL	-	-	-	N	N	N	N	-	-	T
T 58	-	-	M	M	M	FSL	FSL	-	-	-	N	N	N	N	-	-	-	-	T	T	-	-	-	-	M	M	M	-	-	-	-
T 59	-	-	T	T	T	-	-	-	-	-	-	-	-	-	T	T	T	T	-	-	-	M	M	M	M	-	FSL	FSL	-	-	M
T 60	-	-	N	N	N	N	-	-	-	-	M	M	M	M	M	-	-	M	M	M	M	M	-	-	N	N	N	N	-	-	-
T 61	-	-	T	T	-	-	-	-	M	M	M	M	M	-	-	-	N	N	-	-	-	-	-	-	-	-	FSL	FSL	T	T	-
T 62	-	-	-	-	M	M	M	-	-	N	N	N	-	-	T	T	T	T	T	FSL	FSL	T	T	-	-	M	M	M	M	M	-
T 63	-	-	-	-	M	M	M	-	-	M	M	M	-	-	-	-	-	-	-	-	-	M	M	M	M	M	FSL	FSL	M	M	-
T 64	-	M	M	M	-	-	-	M	M	-	-	-	N	N	-	-	-	-	M	M	M	M	-	-	M	M	FSL	FSL	M	M	-
T 65	N	N	N	N	-	FSL	FSL	-	-	-	M	M	M	M	-	-	-	-	M	M	M	M	-	-	M	M	M	M	-	-	-
T 66	M	M	-	-	M	M	M	-	-	M	M	-	-	-	M	M	M	M	M	-	-	M	M	M	M	M	FSL	FSL	T	T	T
T 67	-	-	M	M	-	-	-	-	T	T	-	-	T	T	-	-	N	N	N	N	-	-	-	-	T	T	FSL	FSL	-	-	N
T 68	-	T	T	T	T	-	-	M	M	M	M	M	-	-	N	N	-	-	T	T	T	-	-	M	M	-	FSL	FSL	-	-	T
T 69	-	-	M	M	M	-	-	M	M	M	M	M	FSL	FSL	-	-	M	M	M	M	M	-	-	N	N	N	N	-	-	T	T
T 70	-	T	T	-	-	FSL	FSL	N	N	N	-	-	T	T	-	-	N	N	N	N	-	-	T	T	T	T	T	-	-	N	N
T 71	-	M	M	M	-	-	-	M	M	M	M	M	FSL	FSL	N	N	-	-	T	T	-	-	T	T	T	T	T	-	-	-	-
T 72	-	-	-	-	N	N	N	N	-	-	-	-	-	-	T	T	-	-	N	N	N	-	-	-	-	-	FSL	FSL	M	M	-
T 73	T	T	-	-	-	FSL	FSL	-	-	-	N	N	N	N	-	-	-	-	M	M	M	M	M	-	-	M	M	-	-	N	N
M	0	0	-1	0	0	2	2	0	-4	-1	-2	-1	0	-4	0	-1	-2	0	0	0	-1	0	-1	-1	-2	-3	-2	-1	0	-1	-1
T	0	0	-3	0	0	3	3	0	-2	0	-2	-1	0	-2	0	0	0	-3	-2	0	-1	0	-3	-1	-2	0	0	-1	0	-2	-3
N	0	0	0	0	0	0	3	0	-1	0	0	0	0	0	0	0	0	0	0	0	3	0	-1	0	0	0	-1	1	0	0	-1

Donde las etiquetas representan:

M: Turno de mañana.

T: Turno de tarde.

N:Turno de noche.