Proyecto Fin de Carrera
Ingeniería Superior de Telecomunicación

Optmización de inventario en Farmacia Hospitalaria

Autor: César Hoyos Martín
Tutores: José María Maestre Torreblanca
Isabel Jurado Flores

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera
Ingeniería Superior de Telecomunicación

Optmización de inventario en Farmacia Hospitalaria

Autor:
César Hoyos Martín

Tutores:
José María Maestre Torreblanca
Isabel Jurado Flores

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Proyecto Fin de Carrera: Optmización de inventario en Farmacia Hospitalaria

Autor: César Hoyos Martín

Tutores: José María Maestre Torreblanca
Isabel Jurado Flores

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2015

El Secretario del Tribunal

A mi familia, pues han estado siempre en lo necesario y que saben que, aún habiendo estado perdido, siempre he encontrado y seguido el camino para llegar hasta aquí.

A mis amigos, los que siguen y los que no, pues cada uno ha sido imprescindible para componer quien soy.

A mis compañeros de clase, pues han sido más de cinco duros años, pero vuestra compañía siempre ha hecho menos dura la estancia.

Resumen

Este Proyecto Fin de Carrera no es ni más ni menos que una de las partes en cuanto a la implementación se refiere de un sistema real que servirá para reducir costos y mejorar el servicio en nuestros hospitales. La realización del mismo se ha hecho en conjunto con el Hospital Universitario Reina Sofía de Córdoba y el Hospital San Juan de Dios, los cuales pondrán a prueba el programa en cuanto estén finalizadas cada una de las partes.

El Proyecto es la realización de un programa informático que trata de optimizar y calcular de forma rápida, y eficiente, cuales serán los mejores días de pedidos de un cierto producto farmacéutico a partir del histórico de consumo y stock del mismo, en conjunto con las estimaciones facilitadas por la interfaz con el usuario.

Su realización completa en lenguaje de bajo nivel C, asegura los objetivos antes mencionados en cuanto al cálculo, a la par que facilita ser ejecutado en un servidor por un lenguaje de un nivel más avanzado.

El resultado final que devuelve al programa padre y, consecuentemente al usuario son las fechas con las cantidades del producto a solicitar, dejando siempre que la última decisión la tome el farmacéutico. No obstante hay que tener presente que las futuras evoluciones del sistema podrían llegar a automatizar todo el proceso, realizando las solicitudes del fármaco cuando el programa lo considere conveniente.

Palabras clave

Optimización, inventario, costes, stock, posibilidades, función objetivo, procesos, lenguaje C

Abstract

This Project is one of the main parts in the implementation of a real system which would serve to reduce cost y improve the service in our hospitals. The development of itself has been done in ensemble with Reina Sofia Hospital in Córdoda and San Juan de Dios Hospital in Bormujos, which will probe the complete program eventually.

The Project is the implementation of an informatic program that will try to optimize, compute fastly and efficiently, which would be the better days of orders and also the quantities of a certain pharmaceutical product from the historical of demand and stock of itself, with estimation facilitated by the user interface.

The entire realization has been done in C Languaje, that assure us the main scores mentioned before about the computation, and in the other hand it simplifies its execution in a server by a higher level language.

The final result that returns to the parent program and, consequently the user, are the dates with the quantities of the product to order, remaining always that the apothecary takes the last decision. Although, we have to keep in mind that future developes of the system might automatice the whole process, making orders by itself when the program decide that it is convenient.

Key words

Optimization, inventory, expense, stock, posibilities, objective function, process, C Languaje

Índice

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El Proyecto que enfrentamos es un problema clásico de optimización de inventario, aunque en este caso que nos compete, aplicado al ámbito sanitario, en tanto que el sistema pretende ser una solución eficiente que permita a los usuarios, que serán los farmacéuticos de los hospitales, mejorar y facilitar su labor en cuanto a obtener las cantidades a solicitar en cada pedido y cuando deben solicitarlo.

Teniendo siempre presente que este programa debe facilitar al usuario su labor, que no sustituirla, se ofrecerá al operario las soluciones por pantalla en función de los días de horizonte para los que desea el cálculo y el número de días en el horizonte en los que desea realizar un pedido de un cierto fármaco con sus respectivos parámetros. La respuesta del programa serán las fechas en las que es conveniente pedir en ese intervalo, así como la cantidad a solicitar de dicho fármaco. En ese punto, el farmacéutico será quien decida si seguir o no el resultado del cálculo.

1.1 Proyecto

El sistema global, se puede descomponer en los siguientes problemas, dando en su conjunto el producto final:

- **Estimación**
  Con los datos del histórico del consumo de un fármaco, y pudiendo tener cuenta diferentes factores que pudiesen influir, obtener una estimación de cual va a ser el consumo en los días de horizonte definidos por el farmacéutico.
- **Optimización de costes**
  Se trata de obtener todos los costes asociados a realizar pedidos y almacenar el producto. Utiliza la estimación de los datos, puesto que realiza un seguimiento de estos costes a futuro, inspeccionando cada solución posible hasta alcanzar la óptima.
- **Interfaz al usuario**
  Toda la información del sistema debe tener una entrada de datos sencilla, que facilite la labor del farmacéutíco. En este sentido, el usuario puede registrar los datos de consumo y pedido de un cierto fármaco. Estos serán los datos de partida del problema. Dicha interfaz también deberá presentar los resultados después de la optimización de los costes.

Este proyecto se centra en alcanzar un algoritmo que automatice el cálculo de la **Optimización de costes**.

Se fundamenta en dos grandes subconjuntos:

- Por un lado es capaz de generar todas las combinaciones posibles de días y cantidades de pedido para

un cierto producto. Esto lo realiza descartando posibilidades y aplicando una serie de restricciones que simplifica el número de posibles soluciones del problema.

- Por otro, tras obtener los parámetros del fármaco de la interfaz con el usuario, se prueban todas las posibilidades hasta obtener el menor de los costes, y ligado a este el resultado óptimo.

A pesar de la carga computacional que pudiese tener, se ha hecho en C precisamente para, al trabajar a bajo nivel, reducir dichos costes y tener un mayor control sobre todo el proceso de cálculo. Así mismo, al no conocer que lenguaje de programación utilizará el servidor, el lenguaje C es el que más versatilidad ofrece a la hora de ser ejecutado por otro programa.

1.2 Inventarios

El trabajo a desempeñar depende fuertemente de la gestión de inventarios y de no perder el foco en que estamos trabajando con un sistema real.

El concepto de inventario está muy ligado al concepto inglés *stock* que sirve para definir tanto los inventarios como las **existencias** de un determinado producto, para su explotación o transformación.

En este sentido, hemos de ser conscientes de que el propósito del proyecto en sí no es otro que el de minimizar los costes de un determinado producto, incluyendo el coste de oportunidad, lo cual permite a la compañía en cuestión, en este caso la administración pública, hacer una redistribución de su espacio y recursos, maximizando la explotación de los mismos.

Siempre se debe tener presente que al ser un caso real y de este calado, no se puede permitir tampoco minimizar a todo coste puesto que es la salud de las personas con lo que estamos tratando en última instancia. Por este motivo, todos los productos tendrán un mínimo stock permitido, penalizando fuertemente durante el cómputo del coste, lo cual descartará esas posibilidades como válidas.

Aunque pueda resultar evidente, la importancia de controlar el stock de un producto, radica en la imposibilidad de poder adquirir una determinada cantidad del mismo, en un lugar deseado, de forma instantánea. Por tanto, hay realizar un seguimiento de lo consumido para poder estimar cuánto se va a consumir.

1.3 Optimización

El problema de optimización, es un problema clásico de las matemáticas. Se trata de calcular para una función objetivo, el valor que a la entrada de esta función maximiza o minimiza el valor a la salida.

Este problema se aplica a cualquier ámbito, y es muy frecuente su empleo en ingeniería, economía, física..., dónde se tratan de obtener los mejores valores, y resultados, de una función, entre una serie de entradas posibles.

Hay diferentes enfoques para la resolución de este problema, desde el más simple, que se trata de conocer todas las posibles entradas e ir probando hasta obtener la entrada clave del problema, hasta diferentes métodos númericos que permiten un cálculo más eficiente del resultado en cuestión. Cada enfoque será más o menos útil en función del problema en cuestión, puesto que para un caso en el que, el dominio de entradas sea continua, será mejor utilizar un método númerico que garantice un resultado con precisión.

Para el caso que atañe, la función que tenemos que optimizar es el coste total del stock del producto al final de un horizonte de días dado, y la entrada que debemos hallar será un vector de dimensión igual al horizonte de días dado, donde cada elemento representará la cantidad a solicitar del producto siendo el índice de este elemento el día a realizar el pedido a partir del actual. Siendo mayor o menor la matriz que contenga todos los vectores posibles, es seguro que la dimensión de la misma será siempre discreta y calculable. Es por este motivo, añadido a que el tiempo de cálculo del ordenador será reducido, lo que indica que debe ser una solución correcta obtener primeramente todas las posibles soluciones, para luego proceder a encontrar el vector que minimiza los costes asociados al stock.

El resto de datos del problema en cuestión serán parámetros que el usuario del programa deberá proporcionar para que el programa pueda realizar, por un lado el cálculo de todas las entradas, y por otro, el cálculo de los costes para cada una de estas entradas, seleccionando de todas ellas la entrada que genera un menor coste de stock.

1.4 Estructura del proyecto

Cada una de las partes del presente apartado será detallada a lo largo de la presente memoria, pero es importante tener una visión global de la estructura de cara a estar ubicado siempre respecto al objetivo final del mismo.

En primer lugar, tendríamos un **análisis del problema** en su conjunto y enfoncandolo siempre al caso concreto que nos atañe de optimización de inventario de fármacos para hospitales, dónde la gestión estará realizada por personas, hay ciertas limitaciones a la hora de realizar pedidos, y, en definitiva, estamos ante un caso real, cuya generalización a lo ideal no compete, ni está previsto alcanzar dichos resultados, pero cuya abstacción no ha de ser costosa a partir de lo aquí presentado.

Pudiendo resultar evidente dicho apartado, este será la base sobre la que se desarrolle el problema, pudiendo simplificar su realización y al mismo tiempo la comprensión del mismo.

Una vez realizado este análisis, deberemos proceder al grueso de nuestro trabajo:

- **Cálculo de las posibles soluciones** para pasarlas a continuación a nuestra función objetivo
- **Obtención de la función objetivo** que modele el proceso de solicitud y almacenamiento, la cual nos devolverá en cada instante de tiempo cual es la cantidad almacenada de un cierto producto.

Además de estos procesos, también hay que tener en cuenta que tratándose de un problema real, debemos realizar la gestión de los tiempos con fechas reales, para lo cual tendremos que profundizar en la gestión y trabajo de fechas con C, lo cual puede resultar engorroso en ocasiones, pero ofrecerá una solución más certera del problema y permite mantenernos en el problema real al que se hace frente.

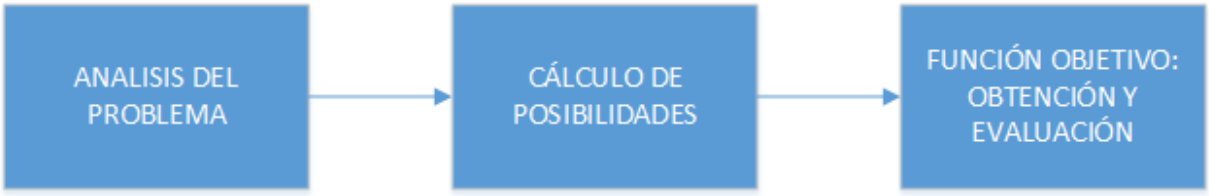

Diagrama de flujo 1-1 Estructura de realización del Proyecto Fin de Carrera

2 ANÁLISIS DEL PROBLEMA

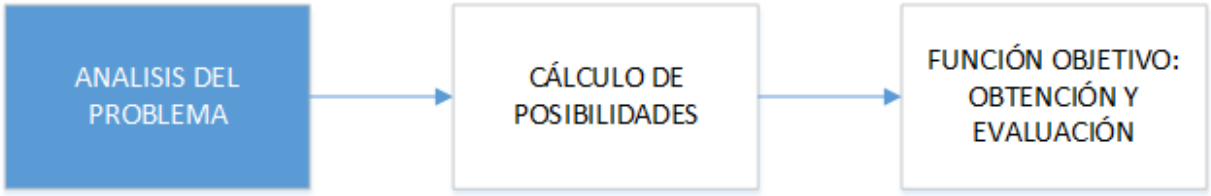

Diagrama de flujo 2-1 Fase de análisis

Analizar un problema debe ser siempre el primer paso en su resolución.

De nada nos valdría enfrentar este trabajo si directamente nos lanzamos con la programación sin comprender por completo la motivación del mismo, ni los objetivos que se pretenden alcanzar.

Una vez habiendo comprendido estos dos puntos, se deberá proceder a hacer un análisis más en profundidad sobre cómo alcanzar dichos objetivos en función de qué parámetros tengo, cuales necesito, y más importante aún, los algoritmos que pueden ser útiles para ir alcanzando objetivos parciales una vez descompuesto el problema global en distintas tareas parciales, más sencillas de abarcar.

2.1 Motivación

La motivación principal del proyecto es el problema de la gestión de inventarios en los hospitales.

La gestión de los mismos, se ha realizado por los técnicos farmacéuticos de las farmacias hospitalarias a partir de las estimaciones que ellos han ido realizando, en función del consumo medio de un cierto producto durante un cierto periodo de tiempo.

En la actualidad, con la entrada del software y los avances teóricos en automatización, es viable pensar que puede haber una solución más certera, que permita al usuario aproximar cuándo y cuánto debe pedir de un cierto producto.

Dicha solución, al ser un problema computacional a resolver por un ordenador, deberá ser presentado al farmacéutico en forma de un programa informático con el cual, de una forma sencilla, pueda interacturar ofreciendo él los parámetros y recibiendo la solución óptima a su problema.

2.2 Objetivos

Siendo el sistema real a implementar de unas dimensiones mayores a las abarcables por un proyecto fin de carrera, el objetivo del programa a implementar en este proyecto es:

- Obtener una solución óptima expresada en forma de vector en función de una entrada dada por dos números a introducir por el usuario, que se corresponden con el número de días en el horizonte en los que se desee pedir y el número de días dentro de ese horizonte en el cual se vaya a pedir.

No obstante, aparte de este objetivo principal, también tendremos una serie de objetivos secundarios que nos permitirán visualizar de una forma más intuitiva cómo desarrollar el trabajo:

- Debe ser interoperable con otros programas, puesto que al usuario el sistema se le presentará a través de una interfaz gráfica.
- Debe ser sencillo de manejar, puesto que en caso contrario, antes de aprender como utilizar un nuevo programa, el farmacéutico se decantaría por seguir utilizando el sistema anterior.
- Debe ser escalable, permitiendo realizar ampliaciones del programa a partir de llamadas desde otro, lo cual se ha realizado como añadido al presente proyecto.
- El código del programa debe ser legible, así como tener funciones que aumenten la legibilidad del mismo, moduladas en librerías dependiendo del objetivo u aplicación de cada función.
- El tiempo de cálculo debe ser el mínimo, puesto que la respuesta va a ser esperada por el usuario en tiempo real.
- Debe ser un ejecutable lo menos pesado posible, puesto que va a ir almacenado en un servidor

2.3 Desarrollo del programa

En primer lugar hay que decir que el desarrollo integro del programa se ha elegido realizar en C.

Este programa será lo más sencillo de lanzar, y que su uso y lectura sea de lo más intuitivo puesto que será un servicio a la interfaz gráfica al farmaceútico. Al ejecutarlo desde otro programa, este debe ser capaz de adaptarse a las necesidades del desarrollador de la interfaz.

Así pues, nos planteamos como requisito para el desarrollo, la salida de forma legible de la información a través de la salida estándar.

También tenemos que tener en cuenta la cantidad de datos con los cuales vamos a trabajar. Es por esto que se deberá hacer una gestión dínamica de la información, pudiendo ser deseable crear funciones propias para simplicar la lectura a la hora de realizar reservas de memoria dinámicas.

Teniendo en cuenta la legibilidad del código, hemos de plantear una serie de librerías con funciones agrupadas en función de las temáticas a las cuales se aplican.

Durante la fase de desarrollo finalmente se abrieron las siguientes librerías:

- Fechas.h

  Incluye todas las funciones para la gestión de fechas que han sido necesarias crear durante el proyecto.

- Ficheros.h

  Gestión y lectura de ficheros externos. Sólo tiene una función, pero se vió conveniente a la hora de hacer el acceso y lectura de los parámetros de los medicamentos, sin engordar en exceso el código de la función main del programa.

- Matrices.h

  Tiene las funciones para realizar las reservas de memoria dinámica para vectores y matrices, así como funciones para imprimir por salida estándar estos tipos de estructura de datos.

- Evalua.h

  Incluye la función que realiza una evaluación de la función objetivo para obtener el coste de cada una de las posibilidades, pudiendo obtener así la óptima.

- Typedef.h

  Tiene los tipos definidos que se han hecho uso a lo largo del proyecto.

Al contar con todas estas librerías, se vio oportuno realizar un archivo para la compilación del mismo. El fichero makefile hace una llamada de compilación en la cual se incluyen todos estos ficheros, con sus dependencias para obtener así el ejecutable deseado, sólo escribiendo en un terminal que se encuentre en el directorio con todos los ficheros la sentencia "make".

Así mismo, a la hora de recibir los datos, sobretodo en cuanto a los parámetros de cada uno de los medicamentos se refiere, deberá contar con un sistema estandarizado acerca de como leer dichos datos, que sea simple a la hora de desarrollar el programa y por otro lado, simple a la hora de pasar dichos datos.

La solución adoptada, no pudiendo ser la más legible por un usuario cualquiera, es la de pasar dicha información a través de un fichero con nombre "datos.pha", que debe encontrarse alojado en el mismo directorio que el ejecutable del archivo.

Este contendrá una ristra de números que atenderán al siguiente formato:

1. Número del stock actual
2. Número del precio de compra del medicamento
3. Número del precio de almacenaje del medicamento
4. Número del precio del coste de pedido
5. Número del precio del coste de recogida
6. Número de la penalización por quedarse sin medicamentos
7. Número del precio del coste de oportunidad
8. Vector de dimensión igual al horizonte explorado de la estimacion de reparto del medicamento
9. Número del máximo stock almacenable
10. Número del mínimo stock almacenable
11. Número de cantidades posibles a pedir
12. Vector de dimensión igual al anterior número con las cantidades que se pueden solicitar

A pesar de no ser un fichero fácilmente legible por ser únicamente números, el hecho es que la comunicación se realiza de forma correcta siempre y cuando el programa que lance la ejecución del mismo conozca de antemano como ha de ser el fichero que tiene que preparar al programa objeto de análisis.

2.4 Estructura de desarrollo del programa

El programa principal deberá ir estructurado de una forma que nos permita obtener soluciones parciales a diferentes tareas que vayamos a ir enfrentando.

Así pues en términos muy amplios, el desarrollo del programa se podría dividir en tres fases principales:

- Definición de tipos

  En esta primera parte se realiza un análisis de los parámetros que van a ser utilizados para el calculo de la optimización del inventario de cada medicamento, y se define una estructura que contenga a todos, para utilizarlos de una forma más intuitiva. Sobre esta fase no nos vamos a extender, puesto que la definición del tipo que utilizamos principalmente a lo largo del proyecto no es más que pasar a código C parte de la estructura del fichero datos.pha, ya que serán almacenados en esta estructura.

La definición de la misma sería la siguiente:

```
typedef struct NODEMEDICINE{
	int stock;			/*Stock actual*/
	float precio_med;			/*Precio de compra del medicamento*/
	float precio_alm;			/*Precio de almacenamiento del medicamento*/
	float coste_pedido;			/*Coste de realizar un pedido*/
	float coste_recogida;	/*Coste de recibir un pedido*/
	float coste_sin_stock;	/*Coste por quedarse sin stock*/
	float coste_oportunidad;/*Coste por tener en stock*/
	int* repartidos;		/*Estimacion de repartidos*/
	int maxStock;			/*Stock maximo almacenable*/
	int minStock;			/*Stock minimo para que no haya desabastecimiento*/
	int nTamPedidos;		/*Numero de posibilidades de pedidos*/
	int* vTamPedidos;		/*Diferentes posibilidades de pedidos*/
}MEDICINE;
```

- Cálculo de posibilidades

  Tal y como nos hemos planteado el problema a resolver, primero vamos a obtener todas las combinaciones posibles para después comprobar cual es la que ofrece unos costes mínimos con nuestra función objetivo. Este sería el grueso real de nuestro problema, en el cual nos enfrentamos a plantearnos diferentes métodos de de análisis matemáticos y/o algorítmicos con el fin de cubrir todas las posibilidades que sean lógicas para nuestro proyecto, sin aumentar, ni por un lado la carga computacional, ni por otro, el tiempo de ejecución del mismo.

- Evaluación de la función objetivo

  Con todas las posibilidades calculadas y todos los parámetros necesarios, se realizará una evaluación de las distintas combinaciones. En este apartado también debemos realizar una definición correcta y aproximada a la realidad de la función objetivo de coste.

Hay que tener en cuenta que estas tres partes no van a ser correlativas en cuanto a esfuerzos y tiempo, pero tiene lógica plantearlos de esta manera en función de los objetivos que se alcanzan en cada una de ellas.

2.5 Mejoras

Además de ceñirnos al problema propiamente definido y explicado, es menester plantearnos posibilidades para mejoras en el futuro, aunque estén fuera del alcance del proyecto.

Es por eso que se detallarán diferentes posibilidades a llevar a cabo para mejorar este proyecto, algunas influyendo al sistema global y otras que podrían afectar a la forma en la que se ha realizado la estructuración del mismo.

Estas mejoras deben ser siempre enfocadas a una mayor funcionalidad del sistema para el farmacéutico, donde en el caso ideal incluso se podría plantear un sistema en el cual no hubiese ningún tipo de interacción directa con ninguna persona, ya que el sistema podría ser lo suficientemente autónomo gracias a los avances tanto en el campo de la automática como de la robótica, para que el farmacéutico sólo interviniese en los apartados de reparto y control del sistema instalado.

Si bien es cierto que en el caso que se plantea de mejora del sistema, habría que tener por otro lado en cuenta los costes derivados de la instalación y mantenimiento del sistema robótico que se encargue de la recepción, y almacenaje de los medicamentos, que pudiendo ser una inversión inicial de mayor importancia, esta se amortizaría con el paso del tiempo, disminuyendo en gran medida los costes al requerir los hospitales de menos personal a la hora de llevar a la práctica estas tareas.

Todo este planteamiento surge a partir de que el núcleo duro del sistema descrito, se encuentra realizado ya

entre este proyecto y los relacionados “Optimización de Estimación de la Demanda en Servicio Farmacéutico Hospitalario” y “Sistema de información para gestión en farmacia de servicio hospitalario” donde se presenta todo la automatización en la obtención de las mejores fechas de pedido, tras toda la carga computacional del mismo.

Haría falta para ampliar un sistema que automáticamente llevase el control del stock, y realizase los pedidos a las empresas farmacéuticas cuando fuese óptimo.

Sin ir tan lejos en los própositos, hay otras soluciones planteables en el corto plazo, como por ejemplo, un programa que lance la ejecución del programa aquí presentado, y que facilite conocer también cual es la cantidad de días óptimos para solicitar un producto.

Este programa en concreto se ha llevado a cabo su desarrollo y se expondrá en un cápitulo individual.

También podría ser ampliable el horizonte máximo de cálculo, que como se verá en su apartado correspondiente tendrá una limitación computacional.

Este problema en concreto se ha despreciao y se ha decidido trabajar con horizontes de pocos días, ya que en otras partes del sistema se realizan estimaciones y la desviación entre el stock real de un determinado fármaco y su estimación aumentará conforme aumente el periodo de estimación.

Siendo este proyecto a aplicar en un caso real, no serán despreciables nunca estas desviaciones, puesto que, en última instancia, sobre quien repercutiría un error en el cálculo será directamente sobre los pacientes del hospital.

3 CÁLCULO DE POSIBILIDADES

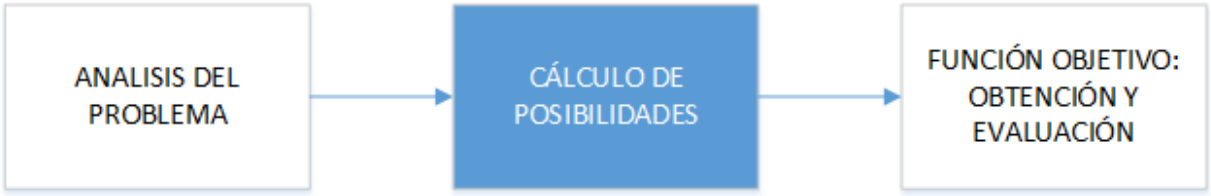

Diagrama de flujo 3-1 Fase del cálculo de posibilidades

Sin llegar a obviar la fase del proyecto en la cual se realiza la definición de las estructuras de datos a utilizar por parte del sistema, el cálculo de las posibilidades puede considerarse como la primera gran parte del proyecto. Dejar a un lado la fase de definición de estas estructuras viene motivado, en gran medida, a que estas estructuras sufrirán modificaciones cuando lleguemos a la fase de obtención de la función objetivo.

Es por eso que nos debemos centrar en los algoritmos para obtener la matriz con todas las posibilidades, que además ha sido lo más costoso en cuanto a dedicación total para la realización y resolución del proyecto planteado.

Así pues, en las sucesivas secciones iremos visibilizando las cribas y lógicas aplicadas para tener una solución total.

El problema planteado aquí podría ser un problema con una entrada y solución propias, más allá de la aplicación al proyecto. Por un lado tendríamos como entradas al sistema el número de días en el horizonte, así como el número de días en los que el farmacéutico desea pedir dentro de ese horizonte.

También será necesario saber las cantidades en las que son posibles solicitar los distintos medicamentos, a fin de obtener una solución completa.

Igualmente esta parte del proyecto es una tarea bastante genérica donde entran más en valor cuestiones puramente combinacionales y algoritmos para obtener dichas combinaciones. Todo esto se hará siguiendo el flujo en el cual se van obteniendo los diferentes resultados parciales del proyecto.

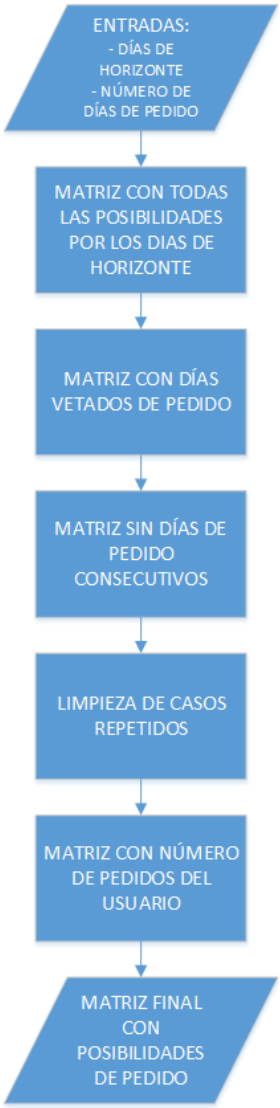

Diagrama de flujo 3-2 Cálculo de posibilidades

El resultado parcial de esta parte del proyecto, será una matriz cuyos vectores tendrán una dimensión igual al horizonte de días pasados por la línea de comandos.

Cada vector estará compuesto por una serie de cifras que indicará el número de pedidos a realizar en cada uno de esos días donde el índice '0' del vector es hoy, el '1' mañana y así sucesivamente hasta dentro de un número igual al horizonte.

3.1 Obtención de matriz con todas las posibilidades

Esta primera matriz a obtener será una matriz que cumplirá que cada vector corresponderá con la codificación en biranrio del índice al que corresponda.

Es por esto que nos planteamos la forma de realizar un contador que nos permita obtener todas las posibilidades y la dimensión de la matriz será $2^n xn$ siendo n el valor de días en el horizonte.

Con todo esto sólo nos queda plantear el algoritmo que llevar a código.

Para poder codificarlo en C tenemos que plantear el contador.

El diagrama de flujo del programa sería el siguiente:

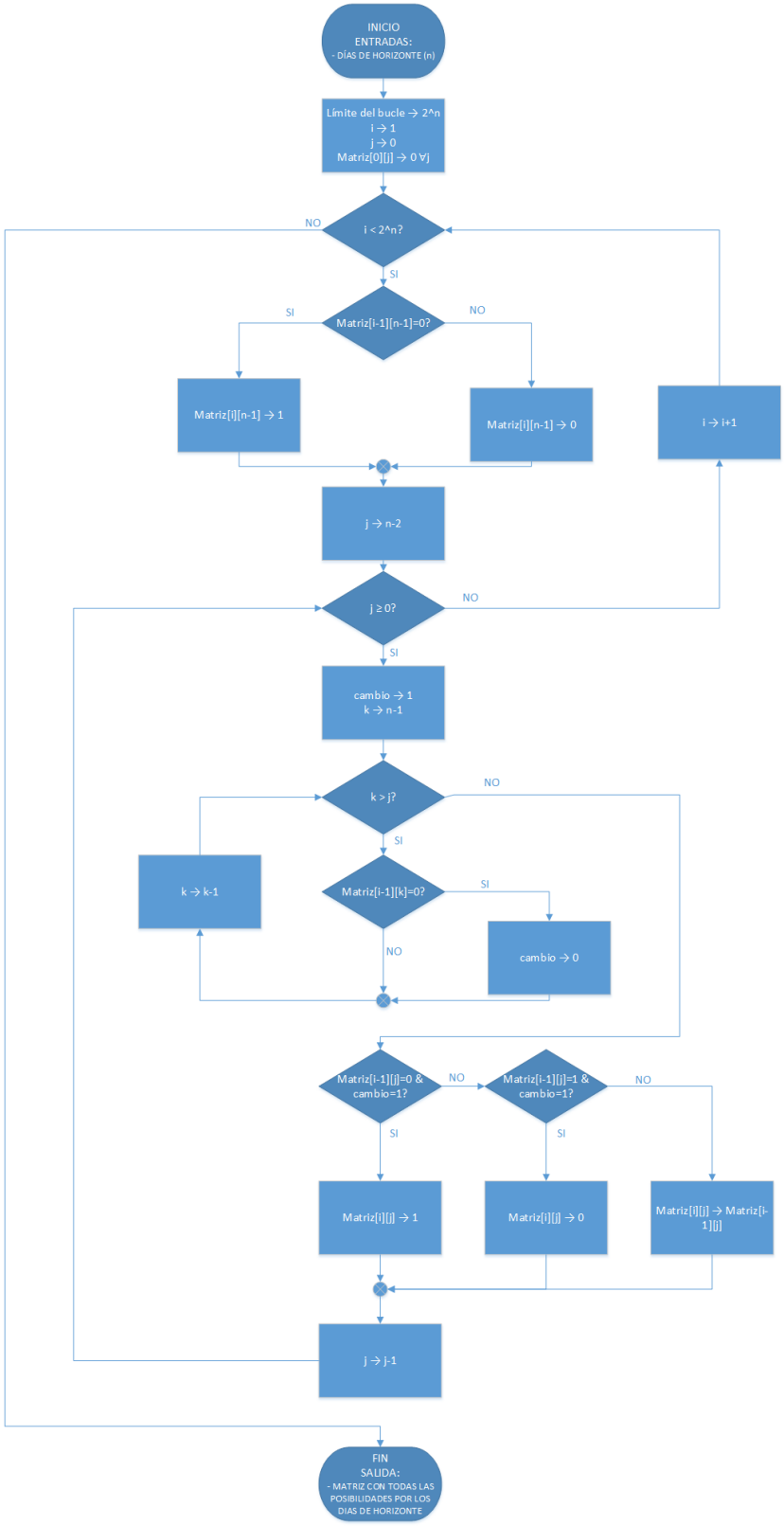

Diagrama de flujo 3-3 Cálculo de matriz con todas las posibilidades por días de horizonte

Como podemos ver, se trata de ir obteniendo cada una de las filas de la matriz completa a partir del estado anterior, conociendo siempre una condición inicial y los cambios que hay entre una fila y la siguiente.

Así pues, primero a partir del número de días en el horizonte n, definimos el límite de nuestro bucle for, que será el 2^n que definimos de forma previa. Esto es útil de cara a realizar la reserva de memoria y trabajar con

esta de forma dinámica, antes de empezar con el llene de la misma.

Hacemos el llenado de la primera fila con todo a 0 y procedemos a entrar en el bucle de cálculo y llenado de la matriz.

La lógica aplicada para cada una de las filas será, tal y como se muestra en el diagrama de flujo, la siguiente:

- Si el primer bit de la fila anterior, entendiendo primer bit como el menos significativo, era '1' ahora será '0', y viceversa. Esto fuerza la alternancia del primer bit, tal y como funcionan los contadores binarios.
- A continuación procedemos a ver como quedarían el resto de bits. Para esto vamos a recorrer todos los índices de cada vector. Utilizamos una variable auxiliar a la cual hemos denominado cambio y que inicializamos a valor '1'.
- Para cada elemento, comprobamos la fila anterior buscando aquellos elementos que marcan el signo del nuevo elemento.
- Primero vemos el bit de índice anterior al actual para ver si estando todos los anteriores a '1' hay que cambiar en la misma posición el bit anterior de '1' a '0' o viceversa.
- En caso de no estar todos los bits anteriores al de la posición actual a '1', se copia el valor en ese índice de la fila anterior a la actual.
- Vemos como así se consigue el contador binario tal y como estaba previsto.

El resultado final de este apartado sería la matriz siguiente con las dimensiones antes explicadas.

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 1 & 0 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 & 1 \end{bmatrix}_{2^n}$$

Matrices 3-1 Matriz resultante del contador binario

3.2 Obtención de matriz con los días vetados de pedido

Una de las funcionalidades que se implementaron tras realizar un análisis detallado del problema, fue que se debían permitir que el usuario (o un sistema externo) indicase los días en los cuales no se puede realizar ningún pedido.

Esta funcionalidad se introdujo teniendo en cuenta el caso real, en el cual, previsiblemente, no se van a realizar pedidos de forma casi segura los domingos, y también que permita al sistema evitar que se deba realizar un pedido un día festivo o por cualquier otro motivo que suponga una imposibilidad en la realidad.

Esto en la práctica se implementó incluyendo la posibilidad de que se introdujesen por la línea de comandos, tras los primeros comandos obligatorios, una serie de fechas que deben ser indicadas en un formato concreto `dd/mm/aaaa`.

Estas fechas, tras ser gestionadas por el programa, se convertirían finalmente en un vector al que hemos denominado `diasNO` y cuya dimensión será igual al número de días en el horizonte. Cada día en el cual se haya prohibido realizar un pedido, se indicará con un '1', actuando este vector como máscara.

La forma en la que trabajaremos será la de recorrer todas las filas de la matriz, pasando la máscara a cada

posibilidad y poniendo a ‘0’ los valores de cada fila que coincidan con el ‘1’ del vector `diasNO`.

El diagrama de flujo para esta fase quedaría de la siguiente manera:

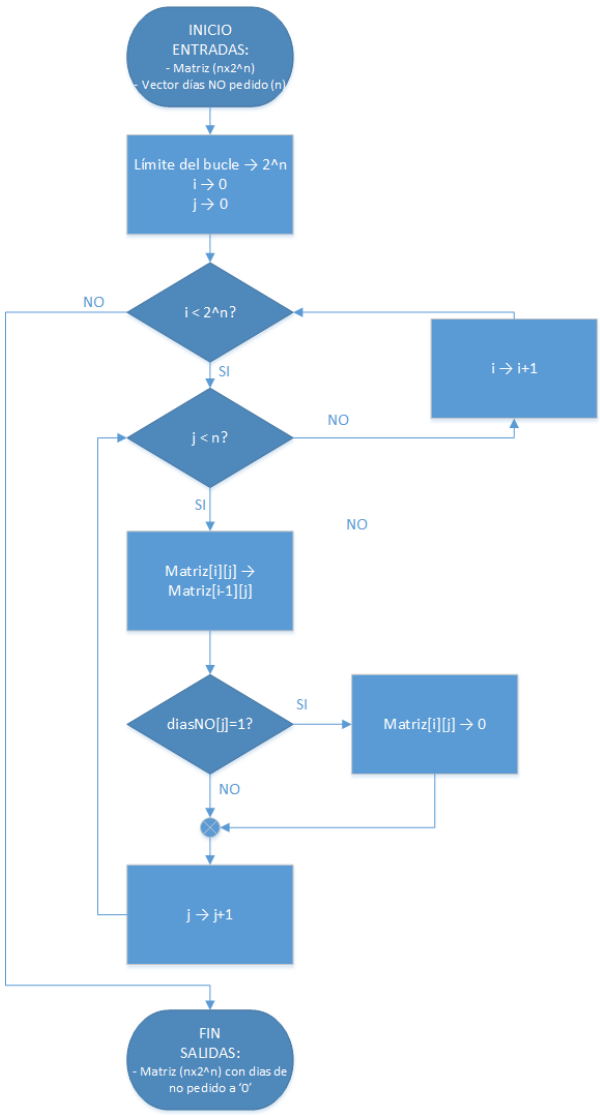

Diagrama de flujo 3-4 Algoritmo de obtención de matriz con los días de NO pedido asignados

La matriz tras aplicarle la máscara a todas sus filas tendrá las columnas correspondientes a los ‘1’ del vector de `diasNO` iguales a ‘0’.

Si por ejemplo tuviésemos que `diasNO[1]` está a ‘1’, obtendríamos una matriz como la siguiente:

$$\begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 1 \end{bmatrix}$$

3.3 Obtención de matriz sin días de pedidos consecutivos

Con la mira siempre puesta en la realidad de un hospital analizamos cuales son las necesidades reales con las que se puede encontrar un farmaceútico en su trabajo diario.

Un planteamiento que hemos desarrollado y llevado a la práctica en el presente proyecto es que no se van a plantear posibilidades en las cuales un fármaco sea necesario solicitarlo en dos días consecutivos. Es más, en la práctica se trata de espaciar lo máximo posible los días de pedido, sin caer tampoco en solicitar todo lo que se prevee que se va a consumir en un año de golpe, puesto que ahí entran en juego el coste de oportunidad, como veremos más adelante.

Por esto mismo, se realiza un paso en el cálculo de la matriz final en la cual se descartan todas aquellas filas que tengan días de pedido, es decir, ‘1’s consecutivos, de la matriz.

El proceso que se ha implementado sigue el siguiente diagrama de flujo:

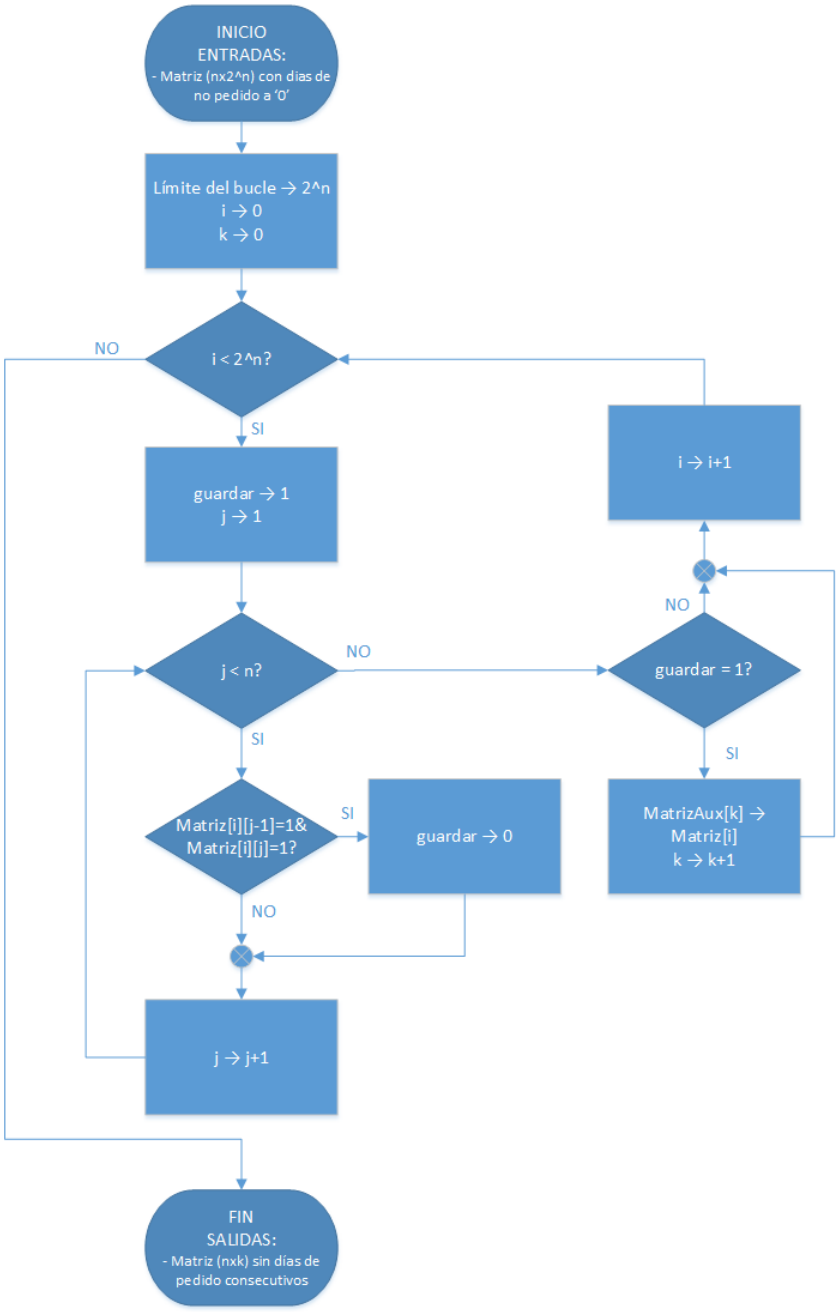

Diagrama de flujo 3-5 Algoritmo para obtención de matriz sin días de pedido consecutivos

Como puede observarse se sigue el siguiente proceso:

- A partir de la matriz obtenida en el apartado anterior, se recorren cada una de las filas.
- En cada fila, se utiliza una variable `guardar` para indicar si se guarda o no dicha fila en la matriz.
- Al recorrer la fila, si se encuentran dos '1's consecutivos la variable `guardar` se pone a 0.
- Una vez se ha terminado de recorrer la fila, se procede a almacenar la fila si `guardar` es igual a 1 y, en caso contrario, no se haría nada, luego se descartaría.

El resultado después de este apartado sería una matriz que no presenta días de pedidos consecutivos.

Para ver un ejemplo de forma gráfica, partiendo de la Matrices 3-2 Matriz con la máscara de días de no pedidos aplicada, se obtendría una matriz similar a la siguiente:

$$\begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 1 & 0 & 1 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 1 \end{bmatrix}$$

Matrices 3-3 Matriz sin días de pedidos consecutivos

Si bien es cierto que no queda del todo claro en la ilustración ya que pretendemos generalizar a una longitud n igual al horizonte de días solicitado, pero si que se percibe la posibilidad de que haya posibilidades de pedidos iguales, debido sobretodo al apartado anterior.

3.4 Limpieza de casos repetidos

Teniendo la matriz anterior, tendremos que hacer una limpieza con los casos repetidos que se dan.

Estas filas que se repiten se deben en exclusiva a la Matrices 3-2 Matriz con la máscara de días de no pedidos aplicada. En el momento en el que dos filas que solo diferían en una columna afectada por la máscara, las dos filas se volverán exactamente iguales, puesto que las dos tendrán ahora ese valor a '0'.

Esto genera además muchas más posibilidades, y por lo tanto más filas a la hora de obtener la Matrices 3-3 Matriz sin días de pedidos consecutivos, puesto que las opciones donde antes había dos unos consecutivos, si en una de esas columnas, ha sido filtrada por la máscara, ahora será una opción viable más.

Es por eso necesario realizar una limpieza de todas estas filas repetidas para que no nos ocupe espacio innecesario en memoria, así como en tiempo de computación, ya que, a la hora de recorrer la matriz, habrá que recorrer un número de opciones superior al necesario, puesto que si una opción se descarta una vez, no es necesario comprobar que no es válida nuevamente.

El diagrama de flujo del proceso llevado a cabo para esta limpieza es el siguiente:

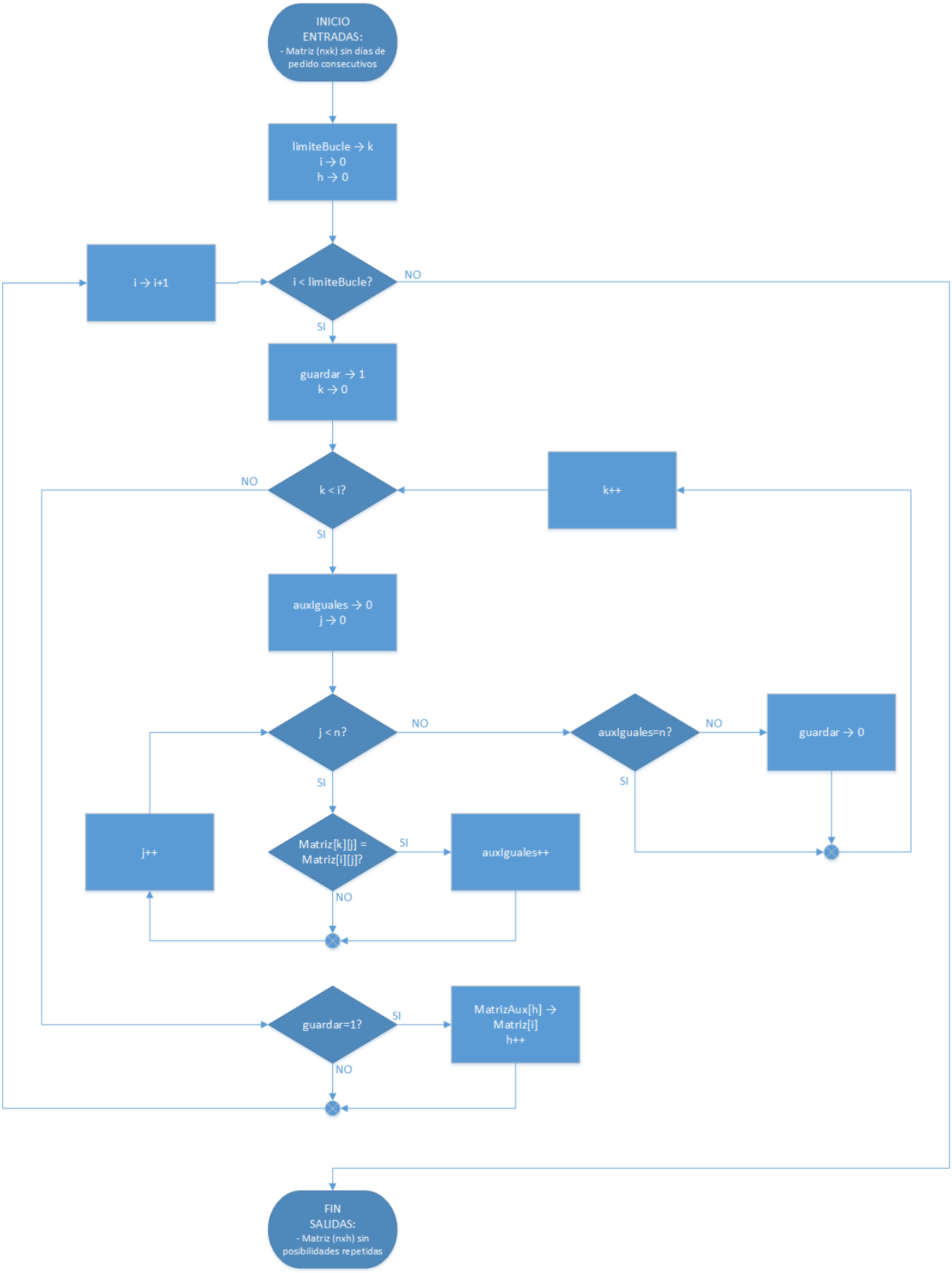

Diagrama de flujo 3-6 Algoritmo para eliminar casos repetidos

Como vemos el proceso se reduce a un recorrido de dos bucles for:

- Un primer bucle for que nos permite ir recorriendo cada una de las filas de la matriz.
- El segundo bucle nos permite comparar la fila actual en la que nos encontramos con todos los elementos ya almacenados, y que por tanto, no están repetidos.
- Para comparar si dos filas son iguales, vamos comparando de uno en uno cada elemento con mismo índice de la fila. Si fuesen iguales, se suma 1 a la variable auxiliar `auxIguales`.
- Tras terminar de comparar dos filas se comprueba si el valor de es igual al valor del número de días en el horizonte de trabajo.
- Si no son iguales se guarda esta nueva fila, y en caso contrario, no se hace nada, con el fin de que quede descartada como posibilidad.

Finalmente tras realizar este proceso, tendremos una matriz que cumple con las premisas de no tener posibilidades que impliquen solicitar en días no posibles, no tener días de pedidos consecutivos y no tener posibilidades iguales.

3.5 Matriz con número de pedidos indicados por el usuario

A partir del resultado anterior, tendremos que ver la lógica para obtener la matriz con todas las posibilidades de pedido que el usuario nos indique.

Este proceso nos deberá facilitar una matriz lista con un número de ‘1’s por fila igual al número de pedidos que el usuario quiere realizar dentro de ese horizonte.

A continuación vemos un diagrama de flujo explicativo del proceso:

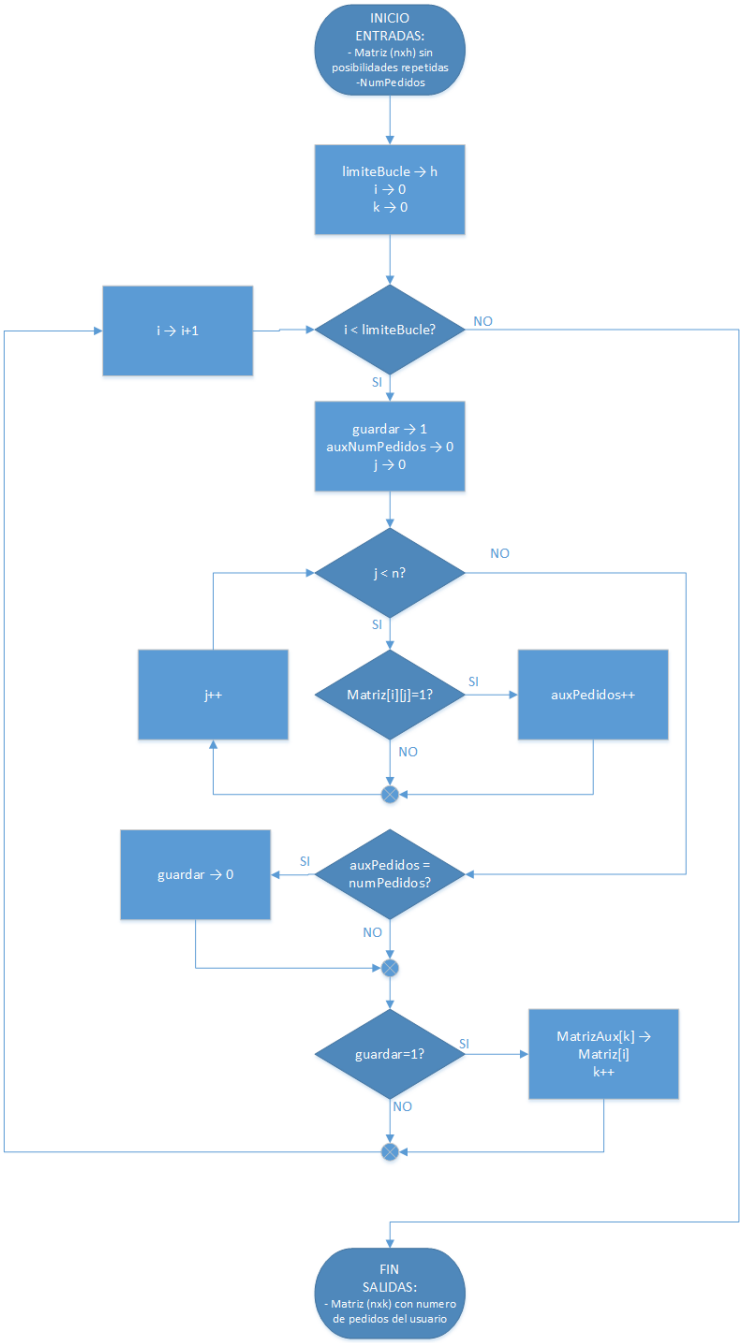

Diagrama de flujo 3-7 Algoritmo para obtener matriz con número de pedidos solicitados por el usuario

Como puede observarse, el proceso es similar al del apartado anterior:

- Realizamos un bucle for de reocorrido de la matriz fila a fila.
- En cada fila vamos recorriendo todos los elementos añadiendo 1 a la variable auxiliar `auxNumPedidos` al encontrarnos con un elemento igual a '1'.
- Al terminar el recorrido de la fila comparamos el número de pedidos solicitados por el usuario con el valor de la variable `auxNumPedidos`.
- Si son iguales se almacena la fila y en caso contrario no se hace nada, quedando descartada.

En el caso de tener una matriz de las vistas en ejemplos anteriores, y si suponemos que el número de pedidos a realizar en el horizonte es igual a 1, obtendremos una matriz como la siguiente:

$$\begin{bmatrix} 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & & & & & & & \cdot \\ \cdot & \cdot & & \cdot & & & & & \cdot \\ \cdot & \cdot & & & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot & & & \cdot \\ \cdot & \cdot & & & & & & & \cdot \\ 0 & 0 & 1 & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 \end{bmatrix}$$

Matrices 3-4 Matriz con número de pedidos solicitados por el usuario

3.6 Matriz con todas las posibilidades

Por último, vamos a proceder a obtener la matriz final con todas las posibilides, tal y como nos plantea el problema a resolver.

A partir de la matriz que se obtiene tras el proceso del apartado anterior, procedemos ahora a incluir las opciones de cantidades de pedido que nos permite el distribuidor de fármacos que está asociado a un medicamento.

Esto en lo práctico, atendiendo a la composición completa del sistema con el cual van a trabajar los farmacéuticos y que se introdujo en el apartado 1.4, el programa recibirá también como parámetros a tener en cuenta a la hora del computo un vector con las diferentes posibilidades de pedido del que dispone el fármaco en cuestión con el que se está trabajando.

En este apartado en concreto, en vez de reducir el número de filas con el que trabajamos en el problema se aumentará.

Una vez hecho esto, pasaremos a obtener la matriz en dos subprocesos.

3.6.1 Obtención de matriz auxiliar de las combinaciones

En este apartado nos limitaremos a obtener una matriz auxiliar que contemple todas las posibles combinaciones a obtener entre las distintas cantidades de un fármaco que se pueden solicitar, en función del número de días en el horizonte en los cuales se van a realizar pedidos.

Para comenzar con esta parte del programa, primero calculamos el número total de combinaciones posibles para cada una de las filas que hemos obtenido en la Matrices 3-4 Matriz con número de pedidos solicitados por el usuario. El resultado a obtener será del número de cantidades de pedidos posibles a realizar, elevado al número de posibles pedidos a realizar dentro del horizonte.

La lógica aplicada para obtener este valor es la de un bucle for para implementar el exponente.

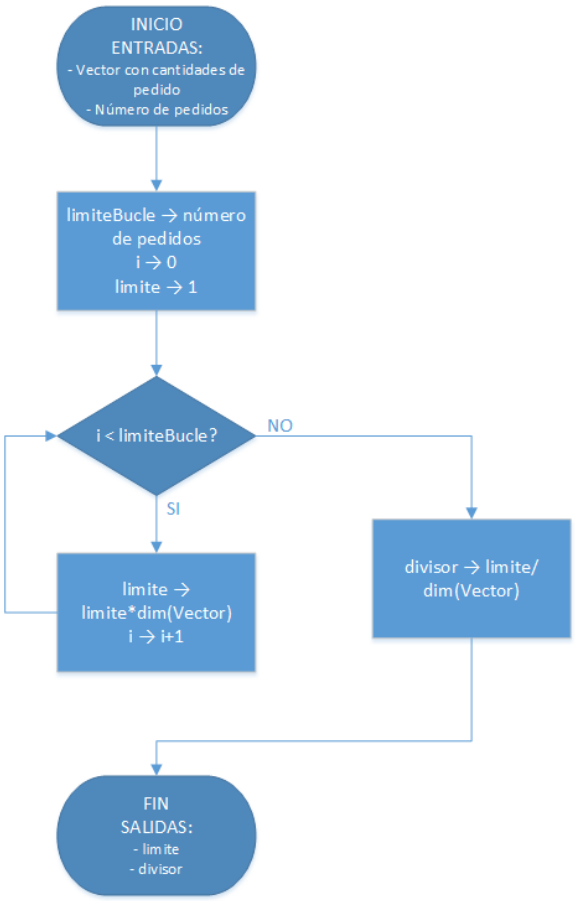

Diagrama de flujo 3-8 Obtenención de variables auxiliares para la matriz de combinaciones

Así pues, tendremos que contemplar este límite como el número de filas de nuestra matriz. El número de columnas estará fijado por la cantidad de diferentes pedidos a poder realizar de ese fármaco.

A la hora de realizar el llene de esta matriz se sigue un proceso como viene en el siguiente diagrama de flujo:

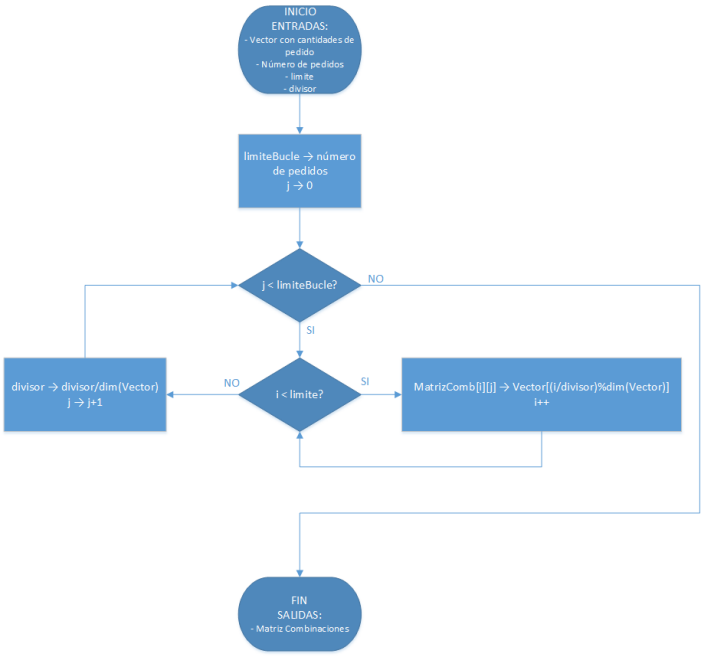

Diagrama de flujo 3-9 Algoritmo para obtener la matriz de combinaciones

Como se puede observar atendiendo al diagrama, el llenado de la matriz lo hacemos avanzando por columnas y luego por filas. O mejor dicho, primero rellenamos el primer elemento de cada fila y luego pasamos a la siguiente columna.

Esto nos permite aumentar el cambio de un tipo de cantidad de pedido al siguiente en función de la columna en la que se encuentre de llenado, gracias a la variable auxiliar `divisor`.

Esta variable se va modificando cada vez que finalizamos con el llene de una columna atendiendo a:

$$\text{divisor} = \text{divisor}/\text{medicine.nTamPedidos};$$

Donde `nTamPedidos` es igual al número de cantidades diferentes de pedidos posible.

Así pues, con esto tendríamos la matriz completa y final a falta de añadir los días en los que no se pide a cada una de las filas.

Para comprender esto de una manera más intuitiva pongamos el ejemplo de que tenemos un vector con las diferentes cantidades de pedido $\{1,5,7\}$ *y el número de días de pedido dentro del horizonte sea igual a* 3.

Esto nos generaría la matriz auxiliar siguiente:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 5 \\ 1 & 1 & 7 \\ 1 & 5 & 1 \\ 1 & 5 & 5 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ 7 & 7 & 5 \\ 7 & 7 & 7 \end{bmatrix}$$

Matrices 3-5 Matriz con todas las combinaciones posibles de los días de pedido

---

3.6.2 Matriz final con todas las posibles soluciones

Una vez obtenida la matriz anterior, ya sólo nos resta aplicar a cada una de las filas de la Matrices 3-4 Matriz con número de pedidos solicitados por el usuario, la matriz anterior.

Para ello lo que tenemos que implementar es un proceso para combinar ambas matrices. Esta lógica se ha implementado tal y como se explica en el siguiente diagrama de flujo.

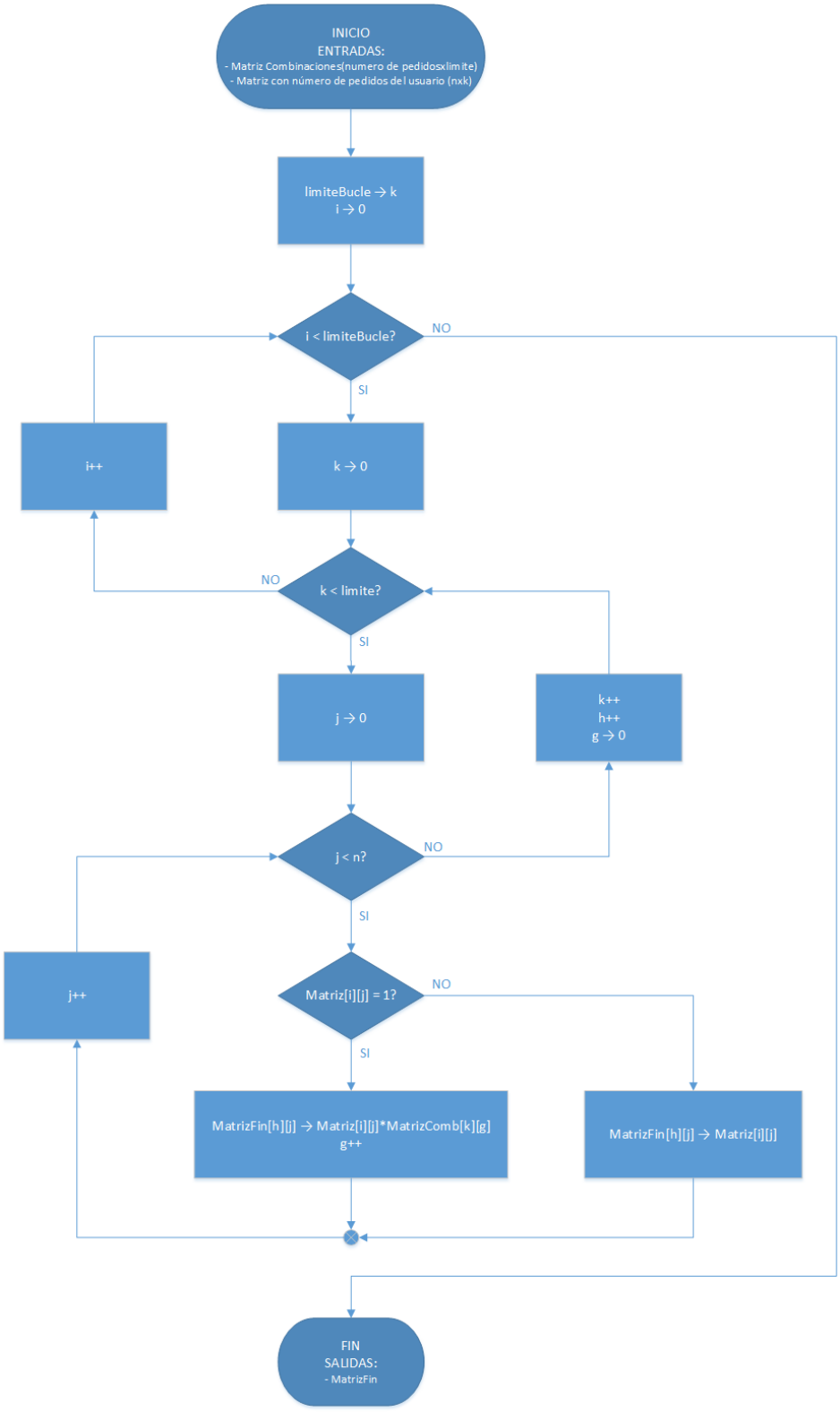

Diagrama de flujo 3-10 Algorimto para obtener la matriz final

Aquí vemos como el proceso seguido ha sido el siguiente:

- Por cada fila de la Matriz 3-4, se acceden a todas las filas de la Matriz 3-5.
- Por cada fila de la Matriz 3-5, se recorren todos los elementos de la fila actual de la Matriz 3-4.
- En cada elemento se comprueba si es un día de pedido, es decir, está a '1', o no lo es, está a '0'.
- En caso de que se encuentre a '1' el valor de este elemento pasará a ser el valor que le corresponda de la Matriz 3-5. En la práctica esto lo realizamos mediante una multiplicación del valor del elemento de 3-5 por el '1'. Se añade 1 a la variable auxiliar que lleva el índice del elemento de la fila de la Matriz de combinaciones en el cual nos encontramos, para que en la siguiente pasada no repita valor.
- En caso de que sea '0', se deja a '0'. Estos serán los días en los cuales no se realizará ningún pedido.

Con todo esto ya tenemos la matriz con todas las posibilidades creada.

Siguiendo con los ejemplos anteriores, si tenemos un solo día de pedido y las tres posibilidades de pedir $\{1,5,7\}$ *obtendremos una matriz tal que así:*

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 5 \\ 0 & 0 & \cdots & 0 & 0 & 7 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 5 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 1 & 0 & 0 \cdots & 0 & 0 & 0 \\ 5 & 0 & 0 \cdots & 0 & 0 & 0 \\ 7 & 0 & 0 \cdots & 0 & 0 & 0 \end{bmatrix}$$

Matrices 3-6 Matriz con todas las posibilidades para nuestro problema

---

Esta Matriz contiene todas las posibles soluciones atendiendo al número de días en el horizonte, el número de días en los cuáles se quiere realizar un pedido dentro de este horizonte, y las distintas posibilidades de pedido que se pueden realizar para un cierto fármaco.

3.7 Conclusiones

En este cápitulo hemos visto el proceso por el cual se obtiene la matriz con todas las posibilidades que nos interesan probar con nuestra función objetivo y así evaluar cual es la opción que optimiza dicha función.

El programa realiza siempre un proceso de descarte de diferentes filas dentro de todas las posibles en el horizonte indicado.

Esto se puede observar en el diagrama de flujo del principio del cápitulo.

Como puede observarse se han hecho para facilitar tanto el cálculo, como para dotar de más algoritmos al sistema, una serie de asunciones para que el programa sea más útil para resolver el problema dado en la realidad:

- En primer lugar, nunca debemos perder la perspectiva de que este sistema se adaptará a la optimización de inventarios, en este caso para hospitales, pero que puede generalizarse para su uso ante cualquier tipo de inventario que se quiera trabajar.
- En este apartado en concreto, no perdiendo la perspectiva antes citada, se trataba de plantear un subproblema del problema real. Esto nos ha permitido plantear un problema de principio a fin extrayéndonos del sistema total.
- Se asume que en un caso real, el usuario real no va a querer realizar pedidos en días consecutivos. Esto es algo lógico, puesto que al trabajar con horizontes de días comprendidos entre 1 y 15 días, se

tratará siempre de minimizar las recepciones a realizar, ya que prima también el factor humano.

- Se ha incluido la posibilidad de descartar fechas de pedido, puesto que nos podremos encontrar con fines de semana, festivos o días en los que no contemos con personal suficiente como para llevar a cabo la recepción de un pedido.

Es con todo esto con lo que comenzamos a trabajar en este apartado del proyecto.

Los resultados tras realizar pruebas con diferentes casos son los deseados, y el tiempo que tarda en realizar este computo de la matriz que necesitamos va en función del horizonte, pero hasta 17 días el tiempo de cálculo total del programa no excede los 3 segundos de computación, llegando a 50 segundos para el caso de 20 días.

Una de las limitaciones importantes a tener en cuenta, es la limitación de la memoria. La cantidad de información a almacenar aumentará más del doble con cada día que aumentemos el rango, pudiendo llegar a desbordar en caso de que el aparato que ejecute el programa no cuente con espacio suficiente de memoria para poder gestionar toda la necesaria simultáneamente.

Igualmente al realizar una gestión dinámica de la memoria y reutilizar la memoria reservada al principio del programa todas las veces que ha sido necesario, se consigue aumentar el rango de horizonte permitido, por la máquina.

Esta limitación en el cómputo, no lo es tal en la realidad. A la hora de realizar la gestión de un inventario, presuponemos que el usuario realizará una consulta al sistema de forma periódica, ya sea una vez a la semana o una vez cada dos semanas. Esto será así puesto que el sistema global, en la parte realizada en otro proyecto, realiza estimaciones del consumo durante un cierto periodo de tiempo.

Las estimaciones se pueden realizar de muchas formas, por ejemplo como media del consumo en un periodo anterior, o por sistemas similares a predictores de Smith.

Pero lo que tienen en común todas estas predicciones es que, conforme nos vamos alejando temporalmente de los datos a priori, se van desviando más de los datos reales futuros, generando un error en muchos casos acumulativos que acaba por no servirnos al alejarnos mucho del momento en el que se realizó la estimación.

Es por ello recomendable que no se trabaje con horizontes superiores a los 15 días, puesto que es un periodo suficiente para haber realizado la estimación del producto sin una gran desviación.

Con todo esto, se da por concluido el cálculo de posibiliades. Ya tenemos todas las opciones posibles, y ahora nos queda tener claro cual es la función objetivo con la que debemos trabajar e ir probando hasta obtener el resultado mínimo de coste, y asocidado a ello, los pedidos que debemos realizar para alcanzarlo.

4 FUNCIÓN DE COSTE

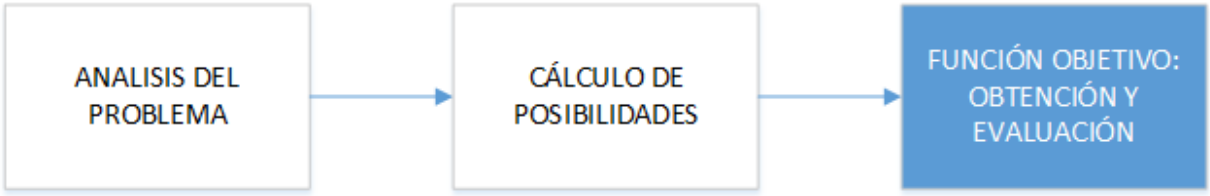

Diagrama de flujo 4-1 Fase de obtención y evaluación de la función objetivo

Una de las partes importantes de este proyecto es poder modelar la función de coste. Esta función tendrá que tener una representación clara de la realidad, teniendo en cuenta todos los factores que pueden influir en el coste total de recibir, almacenar y dispensar un cierto producto. Este apartado del proyecto que se ha realizado también de una forma espcecifica teniendo en cuenta las necesidades del inventario de farmacia hospitalaria, también puede llegar a generalizarse para llevar a cabo la gestión del inventario de cualquier otro producto.

Para llegar a comprender porque decidimos obtener la función que hemos obtenido y más aún, cómo la hemos modelado para su cálculo computacional de esa determinada forma, primero vamos a realizar un análisis de cuales son cada uno de estos factores, en que medida afectan y como podemos modelar su valor de una forma matemática. Todo esto, sin perder nunca de vista cual es el problema real al que nos estamos enfrentando, y cuales son los objetivos que nos hemos marcado para resolverlo.

4.1 Problema de optimización del inventario de farmacias

Los objetivos principales de este problema son:

- Toda la demanda debe satisfacerse
- Se debe reducir el coste total

La principal diferencia de la gestión de inventario en farmacia hospitalaria frente a la gestión de cualquier otro tipo de inventario, es que no podemos dejar a los clientes últimos, es decir a los pacientes, sin medicamentos. Si la gestión fuese de un caso genérico, probablemente el hecho de quedarnos sin stock no supondría más

problema que perder un cliente por no poder ofrecerle el producto cuando ha venido a solicitarlo, pero en este caso no es así. El paciente necesita su medicamento y el hospital siempre ha de tener suficiente producto para no quedarse a cero con el perjuicio que esto supone.

En cambio, la segunda premisa si que es aplicable a cualquier sistema y será el centro de nuestro trabajo de optmización. Igualmente de un caso genérico al caso concreto que nos compete, esto se modelará como una penalización durante el cómputo que será mayor o menor, pero que servirá para descartar la posibilidad de pedido que nos haga quedarnos a cero.

El inventario se podría representar como en la figura siguiente:

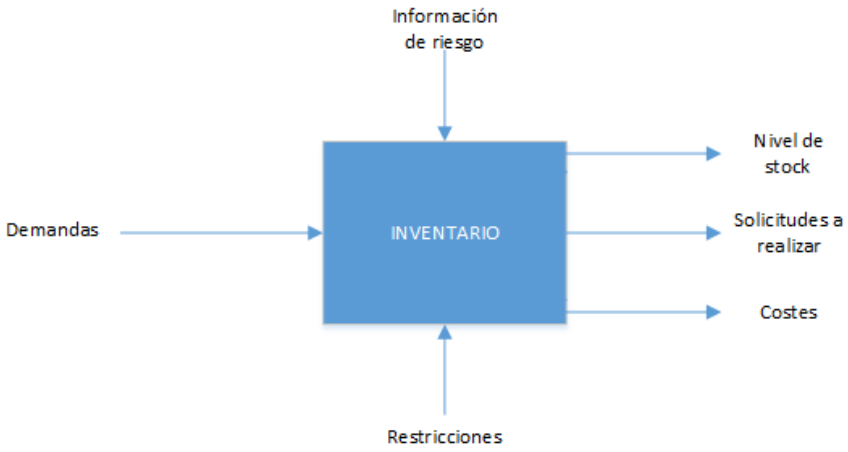

Ilustración 4-1 Modelo inventario

Las entradas al sistema del inventario serán:

- Estimación de las demandas que va a tener.
- Las restricciones que nosotros impongamos al problema. Estas pueden ser que no se permita que el stock sea cero en ningún momento. Es más, para el problema propuesto el farmacéutico puede fijar un límite a través de la interfaz indicando el mínimo stock permitido, así como el máximo posible, para tener en cuenta estos dos límites de cara al cálculo de los pedidos.
- Información de riesgo, que serían perturbaciones del sistema en sí, como por ejemplo los retrasos que puede tener un pedido desde que se entrega hasta que se recibe.

Como salidas tendríamos:

- El stock de cada fármaco.
- Las solicitudes de fármacos a realizar.
- Los costes. Es aquí donde vamos a entrar a optimizar una solución para el problema que se presenta.

A la hora de plantear el problema, hemos de tener en cuenta todas estas variables, pero atendiendo aque la solución que buscamos es la que optimice los costes. Podría haber otras soluciones diferentes para optimizar el stock, manteniéndolo dentro de unos márgenes, o realizar el mínimo número de peticiones de fármacos.

A grandes rasgos, aunque lo iremos desglosando más a lo largo del capítulo, nuestro objetivo es cumplir con lo siguiente:

$$\min_{u} J \qquad (4\text{-}1)$$

Donde J será el coste total asociado a un fármaco y u será el vector que contiene las solicitudes de fármacos a realizar dentro del horizonte dado.

Estos costes pueden modelarse de forma matemática según la siguiente ecuación:

$$J = \beta_1 Demandas(u,t) + \beta_2 Gastos(u,t) + \beta_3 Peticiones(u,t), \quad (4\text{-}2)$$

Donde $Demandas$, $Dispensaciones$ y $Peticiones$, serán los costes totales asociados a cada uno de estos procesos, y estarán relacionados por el vector de solicitudes, así como con los costes que puede tener cada uno de ellos, como veremos a continuación.

4.1.1 Demandas

Satisfacer todas las demandas debe ser un objetivo que nunca debemos perder de vista si queremos evitar que los pacientes se queden en un momento dado sin su medicamento. Esto sería un error muy grave y que debe poder evitarse a toda costa.

El problema de la demanda, es que no es conocible con total seguridad a futuro, ya que trabajaremos con estimaciones de la demanda que puede haber.

Esta demanda será entendida como una variable estocástica, a la que también le puede influir no conocer con seguridad cual será el retraso en la entrega de un determinado fármaco.

Para poder solucionar estas incertidumbres, deberemos tener siempre un stock de seguridad, que no debe sobrepasarse bajo ningún concepto.

Para fijar este stock de seguridad tendremos dos posibilidades:

- Por un lado, podemos modelar este parámetro como un parámetro a optimizar. Este debará responder a la siguiente notación matemática:

$$\min_{\delta,u} \sum_{k=0}^{N} C_{os} \Pr(s(t+k) < 0), \quad (4\text{-}3)$$

  Donde:

  - δ será el vector booleano que modela si hay o no pedido
  - u el vector de solicitudes
  - C_{os} penalización por quedarse sin stock del producto
  - s el stock en cada día en el horizonte
  - $\Pr(s(t+k) < 0)$ modela la probabilidad de que el stock sea cero o menor (esto puede darse en el cálculo pero no en la realidad)
  - N será el número de días en el horizonte.

  Siendo esta solución más exacta, nos deja con ciertas incognitas a la hora de no poder modelar de forma certera la probabilidad para cada día y stock.

- Como segunda posibilidad, podemos mantener un cierto límite fijo para este stock de seguridad. Si bien es cierto que no es una solución tan exacta como la anterior, si que nos permite trasladar a código el hecho de tener una penalización en el cálculo, descartando por tanto aquellas soluciones de u que no cumplan esta limitación.

Se ha optado por tanto por tener un parámetro en nuestro programa que estará fijado por el farmacéutico y que será el stock de seguridad. Si alguna solución no cumpliese con dicho límite se descartaría automáticamente.

Este stock de seguridad se ha modelado en el programa como una variable dentro del objeto fármaco, que se denomina `minStock` y el cual se toma de la ristra de números que se encuentran en el fichero de "datos.pha" que se lee durante la ejecución del programa.

Si el stock es menor que `minStock` en algún día de los contemplados dentro del horizonte, se pondrá un valor previo a J como penalización que en el caso concreto hemos puesto:

```
J= med->coste_sin_stock;
```
(4-4)

Donde `J` es el coste, `coste_sin_stock` la penalización impuesta. De este modo ya tendríamos cubierto el caso de lo que sucede en caso de no cumplir con el stock de seguridad del sistema.

No obstante, tampoco se cierra esta posibilidad al completo, puesto que hemos de tener presentes que siempre se debe facilitar una solución al farmacéutico, aunque esta en la vida real implicaría tener que realizar una solicitud del fármaco concreto a algún hospital cercano.

4.1.2 Gastos

Dentro del coste total del producto, tendremos que tener en cuenta los gastos que tiene asociado al mismo. Estos gastos vendrán derivado de la demanda, el almacenamiento y también teniendo presente el coste de oportunidad.

La función matemática que modela este parámetro a optimizar sería la siguiente:

$$\min_{\delta,u} \sum_{k=0}^{N} \delta(t+k)(pu(t+k) + C_{sh}) + (C_s + C_o)s(t+k), \quad (4\text{-}5)$$

Donde:

- δ será el vector booleano que modela si hay o no pedido
- u el vector de solicitudes
- p precio del fármaco
- C_{sh} costes de envío de los fármacos
- C_s costes de almacenamiento del fármaco asociado a la almacenado de un fármaco
- C_o coste de oportunidad asociado a lo almacenado de un determinado fármaco
- s el stock en cada día en el horizonte
- N será el número de días en el horizonte.

Con todo esto ya tenemos una parte importante de nuestra función objetivo planteada. Tendremos que obtener una función que se repita N veces para ir avanzando en cada uno de los días que tendremos que llevar a cabo del cálculo del coste asociado a los gastos.

4.1.3 Peticiones

El número de peticiones de un producto puede ser también un factor a minimizar.

Esto debemos tenerlo en cuenta puesto que cada vez que se realice un pedido de un cierto fármaco, esto tendrá unos costes asociados de recepción y almacenamiento del mismo.

Bien es cierto, que realizar el número mínimo de pedidos, tampoco nos va a asegurar tener el mínimo coste total J, puesto que el coste de oportunidad, aumenta en función de las cantidades de medicamentos que tenemos almacenados sin ser necesarios.

Es por esta cuestión habrá que encontrar un equilibrio entre ambos factores, pero no es sujeto de análisis del presente Proyecto Fin de Carrera entrar a valorar esta cuestión, ya que son los usuarios finales, farmacéuticos, quienes escogerán el número de días en los cuales quieren realizar un pedido.

Por tanto, debemos optimizar también los pedidos, y ver como afectan a nuestra función objetivo:

$$\min_{\delta} \sum_{k=0}^{N} C_{op}\delta(t+k), \qquad (4\text{-}6)$$

Donde:

- δ será el vector booleano que modela si hay o no pedido
- C_{op} coste asociado a realizar un pedido del medicamento
- N será el número de días en el horizonte.

Ya tendríamos modelado pues el coste total asociado a los pedidos que se realizarán en los N días fijados por el farmacéutico

4.2 Función objetivo

Con toda la información recogida de los apartados anteriores, es ahora el momento de obtener la función objetivo completa con la que vamos a trabajar, y la cual tendremos que codificar en nuestro programa.

Esta función objetivo tendrá que tener partes de las funciones de optimización que hemos visto tanto en el apartado de Gastos, como en el apartado de Peticiones ya que para el apartado de Demandas vamos a trabajar con un valor estático que nos servirá para descartar cada una de las opciones que superen dicho límite.

Así pues vamos a tener una función objetivo completa del coste que será de la siguiente manera:

$$\min_{\delta,u} \sum_{k=0}^{N} \delta(t+k)(pu(t+k) + C_{sh}) + (C_s + C_o)s(t+k) + C_{op}\delta(t+k) \qquad (4\text{-}7)$$

En la cual el valor del stock s, en ningún instante de tiempo, puede ser menor que `minStock` definido por el farmacéutico.

4.3 Desarrollo del programa

Con toda la información recogida en la ecuación (4-7) procederemos a pasar toda la información a Lenguaje C para incluirla en nuestro programa.

La evaluación y el desarrollo de la función objetivo del programa, se encuentra al completo en un archivo denominado *evalua.c*, en una función que tiene el mismo nombre, *evalua*.

4.3.1 Función de stock

Por un lado, tendremos que ser capaces de realizar un seguimiento del stock en función del día dentro de los días de horizonte en el que nos encontremos. Para ello será necesario ver el inventario con sus entradas y salidas.

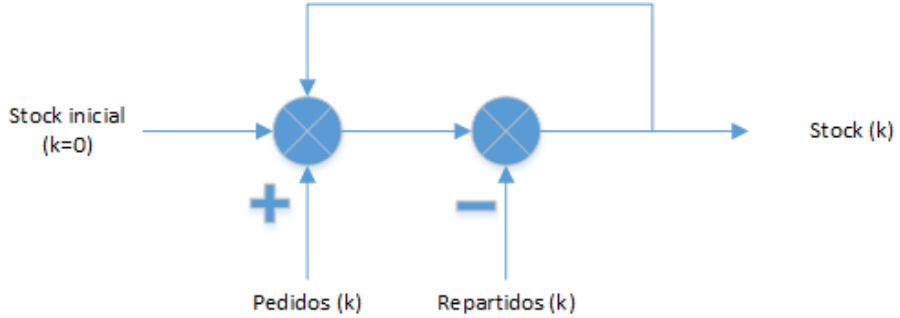

Ilustración 4-2 Modelo del stock del inventario

Todo esto lo podemos modelar de la siguiente manera:

```
stock[k]=stock[k-1]+pedidos[k-retraso]-med->repartidos[k];
```
(4-8)

Cómo puede verse, el sistema vendrá caracterizado por su situación en el día anterior, las entradas serán los fármacos que llegan hoy, que serán los pedidos realizados con los días de retraso correspondientes, y las salidas, las estimaciones proporcionadas por otras partes del sistema global, que serán los fármacos repartidos.

Esta será la primera parte de la función de evaluación. La segunda será la del cálculo del coste con nuestra función objetivo.

4.3.2 Función de coste

Para hallar dicha función, sólo tenemos que ser capaces de ordenar y obtener todos los parámetros que hemos obtenido en la función (4-7). Esto se verá implementado siguiendo la siguiente línea de código:

```
J = J+precio_med*pedidos[k]+(precio_alm+coste_oportunidad)*stock[k]+(coste_pedido+coste_recogida)*orders[k];
```
(4-9)

Se ve necesario aquí realizar una aclaración de las correspondencias entre la ecuación y la fórmula del cálculo

Lenguaje C	$Expresión\ matemática$
`J`	J
`precio_med*pedidos[k]`	$pu(t+k)$
`(precio_alm+coste_oportunidad)*stock[k]`	$(C_s + C_o)s(t+k)$
`(coste_pedido+coste_recogida)*orders[k]`	$C_{op}\delta(t+k)$

Para precisar:

- El coste inicial de `J` se utiliza para tener en cuenta una penalización a la hora de que se exceda el mínimo stock permitido para un cierto fármaco.
- Las variables `precio_med`, `precio_alm`, `coste_oportunidad`, `coste_pedido` y `coste_recogida`, están todas asociadas al objeto `med` que se rellena con toda la información del fármaco durante la lectura del fichero.
- El vector `pedidos` se corresponde con cada una de las posibilididades que nos ofrece la Matrices 3-6 Matriz con todas las posibilidades para nuestro problema hallada en el Cálculo de posibilidades.
- El vector `stock` es el hallado por la función (4-8), y aún calcúlandose en línea, sirve para llevar el coste asociado a tener fármacos en stocks, tanto por el coste de almacenamiento, como por el coste de oportunidad.
- El vector `orders` se halla directamente en la función evalúa y su único fin es discriminar entre los días que hay o no pedido para poder llevar de forma correcta el coste asociado a realizar un pedido de un determinado fármaco.

4.3.3 Proceso de cálculo

Teniendo claras las expresiones realizadas en Lenguaje C para nuestro programa, tendremos que plantearnos ahora el proceso que debe seguir la ejecución del programa para llevar a cabo el cálculo del coste, así como la obtención del vector con la evolución del stock de un determinado fármaco.

Para esto tendremos que realizar un bucle con un número de iteraciones igual al horizonte de cálculo para cada una de ambas funciones.

Finalmente se ha optado por seguir la lógica expresada en el siguiente diagrama de flujo:

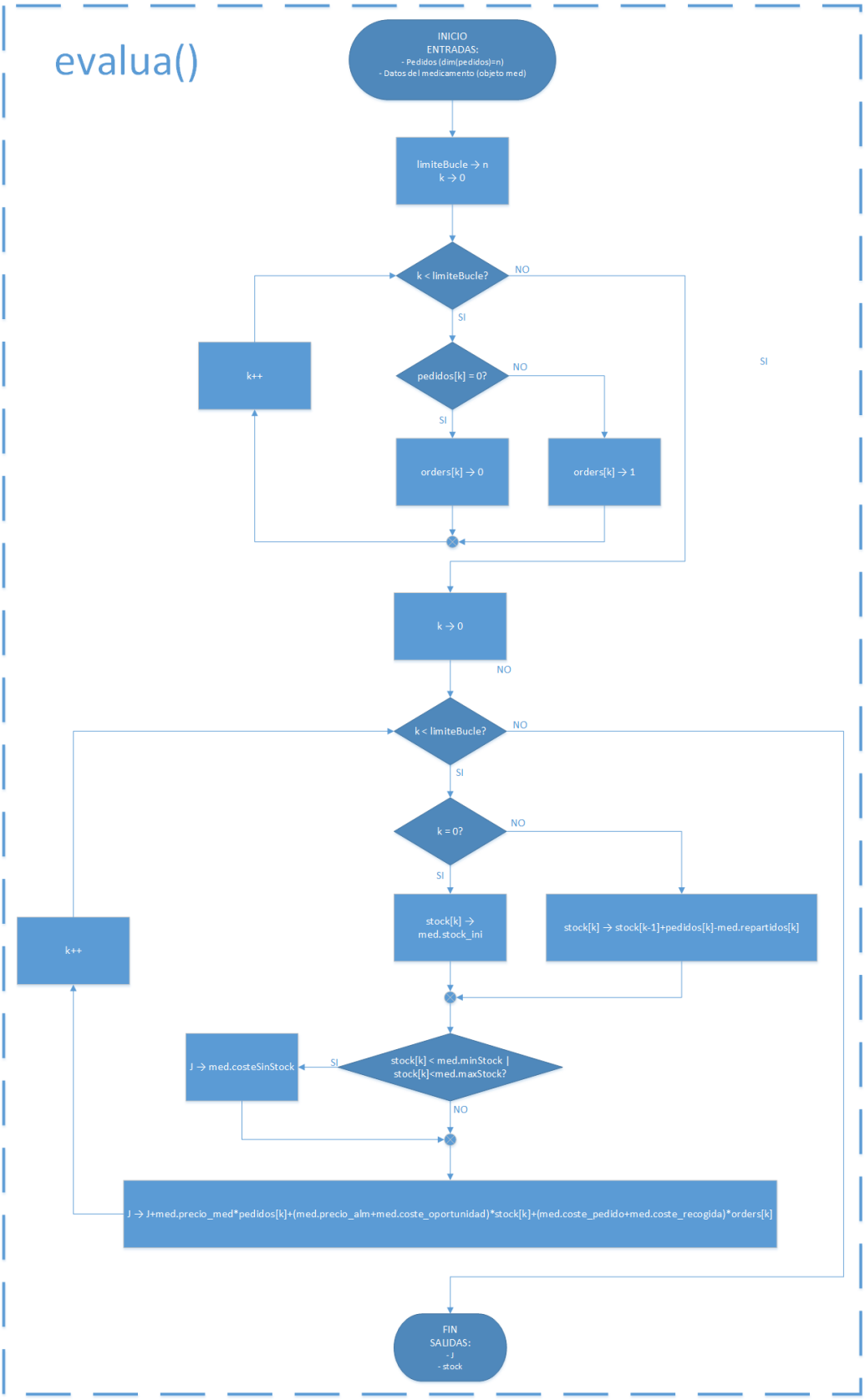

Diagrama de flujo 4-2 Algoritmo para la evaluación de la función objetivo

Se pueden detallar los siguientes pasos tal y como se muestran el diagrama:

- En primer lugar se genera el vector `orders` a partir del vector `pedidos` correspondiente. Esto se realiza comprobando si el valor del vector `pedidos` es cero. En ese caso el valor del vector será `0`, en caso contrario `1`.
- A continuación se entra en el bucle de repetición igual al número de días en el horizonte de cálculo.
- Primero se comprueba si se trata del primer día de cómputo o no. En caso de que lo sea, el valor de `stock[0]` será el del stock inicial, que está almacenado en la variable `stock` del objeto `med` que contiene el stock actual del producto. En caso del resto de días, se opera según (4-8) para obtener el valor del stock en ese día.
- Una vez hallado el valor del stock para ese fármaco en el día correspondiente, se comprueba si se encuentra por debajo del valor del mínimo permitido para ese producto. Si fuese así se asigna un valor a J suficientemente alto para que se descarte dicha posibilidad, y se termina con el bucle.
- Si no se ha penalizado, se realiza el compunto del coste que es acumulativo al coste de los días anteriores, según la función (4-9). Este coste no será el coste exacto de esa posibilidad de pedido, puesto que está sometido a errores en cuanto a la estimación de la demanda del producto. Pero si nos ofrecerá una aproximación válida para saber que posibilidad puede ser más costosa que otra.

Con todo este proceso obtenemos:

- Por un lado el valor del coste total J que se devuelve a la función principal como resultado de la llamada a la función evalua.
- Por otro lado el vector de `stock` que en caso de ser un resultado mínimo, se almacenará en la función principal para ser mostrado por pantalla al finalizar la ejecución del programa total.

4.4 Obtención del mínimo

Como hemos visto en el apartado anterior, ya tenemos realizado el proceso para calcular los costes y el stock asociados a cada una de las filas de pedidos que hallamos en el apartado Matriz con todas las posibilidades.

Es ahora el momento en el cual, con toda la información disponible en cuanto a costes se refiere, vamos a proceder a hallar el mínimo.

No vamos a poder ofrecer el coste exacto mínimo, ya que, como hemos explicado en apartados anteriores, está sometido a errores derivados del error entre la estimación de la demanda futura y la real.

Nuestro objetivo es ofrecer al farmacéutico el resultado final de la ejecución del programa que serán las fechas en las cuales debe realizar pedidos, y con las cantidades de los pedidos a realizar de un determinado fármaco.

Para ello, dentro del programa principal se ha implementado la siguiente lógica:

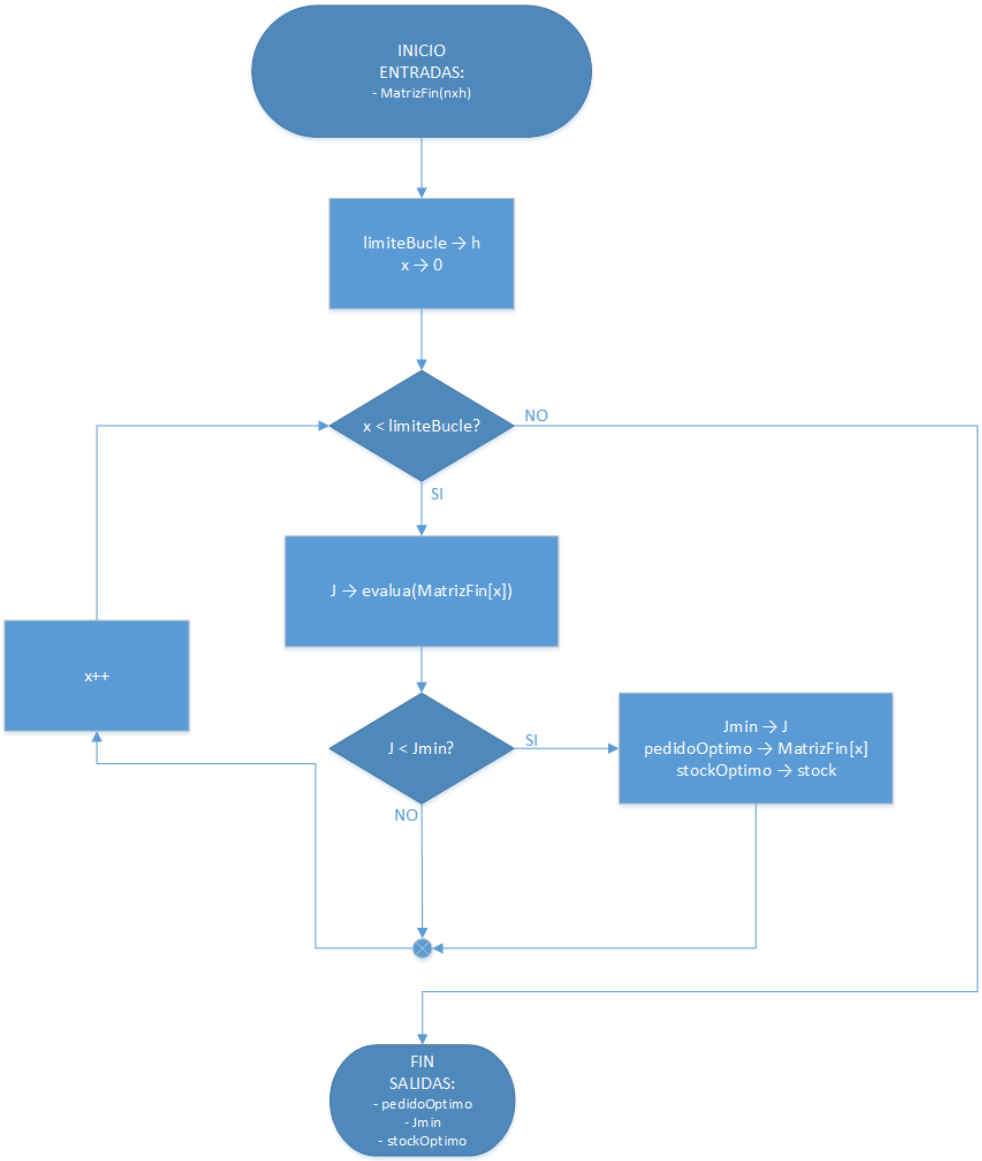

Diagrama de flujo 4-3 Algoritmo para el cálculo de la posibilidad óptima de pedido

Como se puede observar el proceso que seguimos es el siguiente:

- Se inicializan las variables necesarias para la correcta ejecución del código como:
  - `Jmin` almacenará el valor mínimo de coste hallado.
  - `stock` vector auxiliar para pasar los valores del vector de stock hallado en la función *evalua*, a la función principal.
  - `stockOptimo` vector que almacenará el valor del vector de stock relacionado con el coste mínimo hallado.
  - `vectorOptimo` vector que almacenará la fila de la matriz de posibilidades que nos ofrezca el menor coste.
- Realizamos un bucle de ejecución con un número de iteraciones igual al número de filas de la Matrices 3-6 Matriz con todas las posibilidades para nuestro problema. Así garantizamos que vamos a realizar la obtención del coste asociado a cada una de las posibilidades de pedidos.
- En cada iteración se procede a llamar a la función *evalua* de la siguiente forma:

```
J = evalua(matrix[x], TAM, 0, stock, &medicine);
```
(4-10)

- El resultado de la iteración se comprueba con el resultado mínimo almacenado hasta ese momento. Si el resultado obtenido fuese menor, se procede a almacenar los valores ligados a esa fila como valores óptimos, así como renovando el valor de `Jmin`. En caso contrario, no se realiza ninguna operación, descartando por tanto dicha posibilidad.

Con todo esto, obtenemos el objetivo principal del programa, un vector que contiene la cantidad del fármaco que se debe solicitar, cuyo índice nos indica el día en el que debemos solicitarlo en función del índice '0', que es el día actual en el cual nos encontramos.

4.5 Conclusiones

Durante este cápitulo hemos realizado el desarrollo de la función objetivo así como su implementación en el programa.

Es muy importante tener en cuenta que, a pesar de ser la función objetivo, la función para la obtención del coste total de un determinado fármaco, el coste mínimo objetivo no será el coste real, debido al error que se cometerá entre la estimación de la demanda y la demanda real.

El cálculo realizado nos ofrece un resultado óptimo del problema a resolver, que es la posibilidad de pedido que genera menos coste expresado en forma de vector en el cual cada valor de `vectorOptimo` será la cantidad de producto a solicitar, y el índice su relación con el día de hoy, teniendo cómo hoy el índice '0'.

Con esta información, ya sólo nos resta realizar la gestión de las fechas para poder ofrecer el resultado al usuario en el cual se le muestre la fecha y la cantidad de producto a solicitar en dicha fecha.

No se debe perder la perspectiva del funcionamiento que va a tener este programa en la realidad, ya que su funcionamiento va a ser de forma transparente al farmacéutico, puesto que lo ejecutará una capa intermedia que servirá como interfaz. Es por este motivo, por lo que también se ofrecen por salida estándar otros valores que pueden resultar de utilidad para la capa intermedia.

Estos valores son la estimación de la evolución del stock en los días de horizonte solicitados, y la estimación realizada del coste mínimo.

Toda esta información se muestra siempre de la misma forma, para mantener así una consonancia en los datos y que puede ser recogida con relativa facilidad por el programa que ejecute la presente parte del sistema global.

Así pues, tras haber realizado un análisis de diferentes posibilidades de ejecución, el programa no aumenta de forma significativa su tiempo de ejecución. Tras haber completado el cálculo de todas las posibilidades en función de los días de horizonte, días de pedidos y cantidades que se pueden solicitar a la empresa farmacéutica como ya se vió en el Cálculo de posibilidades, añadir el cómputo del coste para cada una de estas posibilidades no añade una carga temporal del cómputo que sea reseñable.

Vemos pues como, teniendo en cuenta todas las variables y parámetros que pueden afectar a nuestro producto, hemos obtenido nuestra función objetivo que modela los costes asociados a la gestión de un fármaco. Tras esto, hemos llegado a la solución deseada, que es el programa que nos permite obtener las peticiones óptimas a realizar de nuestro producto.

5 EJECUCIONES Y RESULTADOS

Teniendo el programa, debemos hacer las comprobaciones pertinentes para comprobar cual es el resultado de la ejecución del mismo y si concuerda con lo esperado a fin de tener un sistema que permita optimizar las solicitudes de fármacos, así como todos los costes asociados al mismo.

Para tal fin hemos desarrollado unas simulaciones que nos permitirán comprobar la utilidad del mismo, fijando unos estándares en las mismas, en los cuales siempre se van a comparar horizontes con la misma cantidad de días, y teniendo presentes los pedidos que se realizaron conforme a los que se calcula que se deberían haber realizado.

Utilizando estos datos, necesitamos comprobar cual es el coste que origina. Para tal fin hemos utilizado la función de evaluación que obtuvimos en el capítulo anterior, realizando las llamadas pertinentes en función de las demanda estimada que será la misma para ambos casos, pero con diferentes posibilidades de pedido.

Así iremos comprobando el coste que genera para un mismo medicamento el haber solicitado cuando se hizo según el histórico.

En primer lugar vamos a generar un programa cuya única utilidad será facilitarnos las llamadas a la función *evalua* descrita en el Diagrama de flujo 4-2 Algoritmo para la evaluación de la función objetivo.

A la hora de realizar esta llamada, ya tendremos previamente preparado el fichero con todos los datos necesarios para el cálculo de dicho fármaco en cuestión en el fichero *datos.pha*, el cual nos servirá también para la ejecución del programa completo, del cual podremos obtener nuestras conclusiones al comparar ambos costes, ya que, aunque ninguno vaya a ceñirse a la realidad, por los errores de estimación, si que nos podrán ofrecer una comparativa de los mismos, para ver cual ofrece una solución de menor coste.

5.1 Generar el fichero de pruebas

A la hora de realizer el computo y las pruebas de nuestro programa, en primer lugar tendremos que tener realizado un fichero que contenga todos los datos del fármaco.

Este fichero es *datos.pha* y contiene información referida a los costes asociados a dicho medicamento; cantidades posibles de pedido del mismo; información del stock en el momento de realizar la consulta al programa; y por último, un vector con los datos estimados dentro del horizonte.

Antes de proceder con la comparación, debemos tener un fichero real en cuanto a la estimación de la demanda

se refiere que nos sirva para comparar el método utilizado en el hospital con nuestro programa.

En el hospital, el método utilizado para la estimación de demanda futura se realiza mediante la media aritmética de las demandas de las 52 semanas anteriores (datos de todo un año).

Para poder recrear la estimación del consumo, tenemos los datos entre desde principios del 2012 hasta el final del 2013, en concreto de dos fármacos *Medicamento 1* y *Medicamento* 2 de los cuales disponemos de información de las repeciones, salidas para uso o salidas por orden de movimiento. Para fines estrictamentos númericos, las salidas para uso y salidas por orden de movimiento serán consideradas salidas del stock, y será indiferente a la hora de realizar las estimaciones de demanda del stock que se contemplan, dado que el sistema que nosotros contemplamos es el mostrado en la Ilustración 4-1 Modelo inventario:

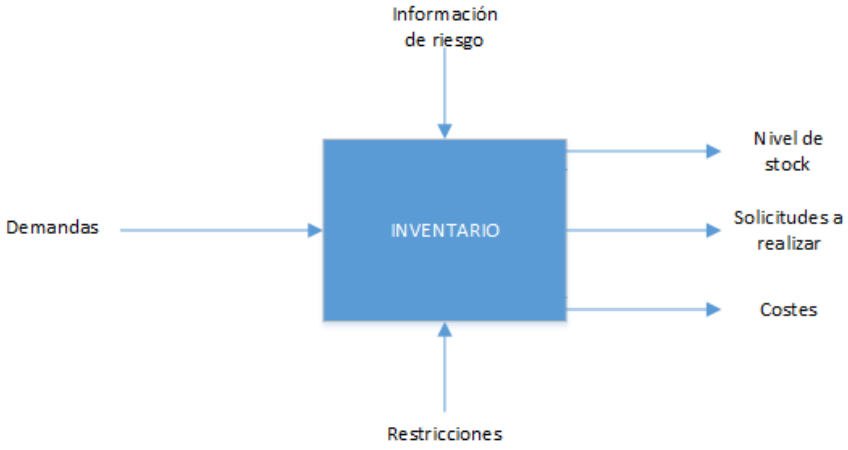

Ilustración 4-1 Modelo inventario

Disponiendo de todos estos datos, así como un dato del stock en una fecha concreta, podemos reproducir el flujo del stock en el tiempo, disponiendo así del dato de stock en el momento en el que nos interese.

También con fines de la simulación, se ha creado un programa en matlab que realiza la media de las 52 semanas previas a las semanas de estudio, por lo que tendremos la estimación con la cual se ha trabajado en el hospital disponible, gracias a todos estos datos.

El resto de datos no están disponibles pero se han realizado aproximaciones al mismo. Se detallan a continuación los cálculos y los valores que se han asignado al $Medicamento\ 1$, teniendo que para el *Medicamento 2* se procederá de igual forma:

- Coste del medicamento es un dato ofrecido en los ficheros, que será de $620.38€$.
- Coste de oportunidad sería el dinero que se obtendría de haber realizado una inversión con el stock almacenado. Lo estimamos en torno a un 1% el valor que se obtendría de los intereses, por lo que será $6.20€$.
- El precio de almacenamiento del medicamento es un dato del que no disponemos pero que no va a variar de forma considerable el coste, teniendo presente que el coste no será elevado en comparación con el del medicamento y coste de oportunidad. Lo fijamos en $0.25€$ por unidad.
- El coste de realizar el pedido se estima en función de una jornada laboral de $8\ horas$ y proponiendo que se tarda en realizar un pedido $30\ min$, siendo quien lo realiza un técnico de la farmacia que tenga un salario de $2000€$ mensuales. Escogemos como media de días en los cuales se trabaja en un mes 22. En total por cada pedido estimamos $5.68€$.
- El coste de recepción de pedido lo estimamos de la misma manera que el anterior, asociado también al salario del farmacéutico. Por tanto por cada recepción $5.68€$.
- El coste sin stock lo estimaremos por un lado como el coste por parte del farmacéutico de realizar un pedido y por otro el coste de enviar un mensajero entre hospitales. Si ponemos que son dos hospitales

que se encuentran en ciudades diferentes, aunque próximas, pongamos que se tardan $2horas$ en la realización. Por tanto se estima como el salario de $2horas$ y $30min$ por parte de personal del hospital un total de $28.40€$.

- Los datos de estimación los obtenemos a través de las funciones en matlab que realizan la media del consumo de las $52\ semanas$ anteriores a la semana en cuestión que se esté proyectando. El problema de dicha estimación es que no nos sirve en cuanto a consumo diario, pero atendiendo a los registros de retirada del producto se puede proponer que sean dos retiradas semanales. Para tal fin, se realiza una estimación de $14\ días$.
- El stock mínimo lo fijamos en función de los regitros que hemos estado revisando, tomando el valor mínimo de stock en el periodo de análisis. Se obteniene por tanto 10 unidades de stock mínimo.
- En cuanto al stock máximo, al no contar con los datos del hospital, lo obtenemos a partir del máximo almacenado en ese periodo. Se queda pues $50\ unidades$ máximas.
- Las posibilidades de pedidos, también es un dato que se obtiene de ambos ficheros, puesto que analizamos todos los pedidos realizados en los dos años de registro. Las posibilidades son $20\ y\ 25$.

Con todo procedemos a recrear nuestro fichero *datos.pha*. Este fichero deberá variar en función del periodo de análisis de los datos al cual nos estemos refieriendo, que para el caso del *Medicamento 1* serán dos, así como para el *Medicamento 2*.

5.2 Pruebas y análisis

A continuación se van a realizar una serie de pruebas para ambos medicamentos, en los periodos en los cuales se registraron recepciones de pedidos por parte del hospital, para comparar así si reduce costes o no el programa realizado.

5.2.1 Medicamento 1. Primer pedido

Situamos en el horizonte de días de pedidos, el día en el cual el hospital realizó un pedido, en concreto, en los $10\ días$ previos a haber realizado el pedido, con $4\ días$ añadidos tras esta petición, teniendo así el horizonte de $2\ semanas$ con el cual vamos a trabajar.

Se realiza en primer lugar la ejecución del programa que nos estimará los costes de haber realizado el pedido de $25\ unidades$ tal y como se realizó el $26/11/2015$, que realiza el seguimiento del stock para los $14\ días$ siguientes a partir del $16/11/2015$, para comprobar si se realizó el pedido de forma acorde a lo óptimo.

La ejecución del programa nos devuelve un coste aproximado de la gestión del stock:

```
J=17591.31;
```
(5-1)

A la hora de realizar la comprobación lanzamos una ejecución del programa con los parámetros de entrada por línea de comandos, 14 y 1 puesto que la solución dada por la farmacia fue de realizar un solo pedido dentro de ese horizonte, comprobaremos si es válido este supuesto. Incluimos la línea de comandos de ejecución del programa:

```
./OFH 14 1
```
(5-2)

El resultado que nos devuelve el programa de optimización de fármaco para hospitales es:

```
J=13812.16;
```
(5-3)

Además, nos indica que el día óptimo para realizar el pedido es el último de los posibles, disminuyendo así los costes derivados del coste de oportunidad lo máximo posible.También hay que tener presente, que mientras el hospital solicitó 25, el programa nos ofrece la solución con $20\ unidades$ del fármaco.

No obstante, tras realizar otra serie de pruebas se acaba llegando a que, para este caso de pruebas, la mejor

opción es la de no realizar ningún pedido del producto dentro de este horizonte dado, obteniendo un coste aproximado de:

```
J=1264.20;
```
(5-4)

Este resultado nos indica el hecho de que hay que evitar realizar pedidos, pues son días de repunte en cuanto a costes, sobretodo por el coste asociado a los productos en sí, al ser fármacos para tratamientos especiales los cuales tienen unos costes muy elevados como se ha indicado previamente.

Antes de finalizar la simulación, hay que comprobar con los datos de stock reales de ese periodo si hubiese sido válida nuestra solución.

Realizando estas simulaciones en las cuales se han sustituido la estimación por los datos reales de demanda de producto, se obtienen los resultados que siguen:

- Para el caso en el que realizamos la misma petición que realizó el servicio farmacéutico, tenemos un coste de $J = 17636.46$.
- Para el caso que hemos obtenido usando el algoritmo del programa, el coste se reduce a $J = 13857.31$.
- Por último, en el caso de no haber realizado pedido alguno, tal y como habíamos visto que optimizaba aún más los costes $J = 13874.35$.

De esto concluimos que la mejor opción era la de solicitar el último día dentro del horizonte.

La variación entre el caso en el cual hacemos la consulta con un día de pedido y con cero días de pedido, es que finalmente tenemos que realizar un pedido adicional, con el sobrecoste que supone el hecho de realizarlo de urgencia a otro hospital.

Así pues, en este caso de análisis hemos comprobado como afectan las decisiones que tomamos sobre la estimación a un caso real. También hemos visto que ajustar el stock tanto nos puede conducir en un error en un caso real, en el cual nos quedaríamos por debajo del stock de seguridad previsto.

5.2.2 Medicamento 1. Segundo pedido

Procedemos ahora a analizar un segundo caso de pedido en el cual tendremos las mismas condiciones que antes, puesto que los resultados obtenidos de la estimación tal y como se realiza en el hospital nos devuelve unos resultados indenticos al caso anterior, y el medicamento con el que vamos a trabajar seguirá siendo el mismo.

Lo que sí va a variar será el stock inicial a la hora de realizar nuestros cálculos.

En este caso obtenemos los siguientes resultados:

- Para el caso con el pedido realizado por el servicio hospitalario, el coste ascendería a $J = 19068.36$.
- Con la simulación lanzada para un pedido, $J = 15289.21$.
- Con la simulación lanzada para cero pedidos, $J = 2741.25$.

A la vista de esto, nos decantaríamos nuevamente por la opción de no realizar ningún pedido en los días de horizonte.

Ahora, al igual que en el caso anterior, procederemos a realizar también las simulaciones con los datos de demanda real que tuvo el producto durante ese periodo de tiempo.

Mientras que el hospital solició 25, nuestro programa nos obtiene que se deben solicitar 20.

Los resultados que se obtienen de las estimaciones de costes serán los siguientes:

- La operación que realizó el servicio hospitalario acarreo un coste de $J = 18526.56$.
- En caso de haber realizado un pedido el coste de la gestión del inventario hubiese sido de $J = 14747.41$.

- Finalmente, si hubiésemos hecho lo que nos había indicado el programa obtendríamos $J = 2199.44$.

En esta ocasión, el resultado dado por la computadora es el que definitivamente hubiese disminuido los costes.

Es menester señalar que en esta simulación, los costes disminuyen conforme aumentan la demanda. Esto se debe principalmente al coste de oportunidad y al coste de almacenamiento, así como no haber ningún caso de expcecionalidad en las ejecuciones ni de la estimación, ni con la demanda que finalmente hubo, lo cual no genera sobrecostos.

En cuanto a los casos analizados para el *Medicamento 1* hay que señalar que las ventanas de periodo, así como los horizontes en los cuales hemos trabajado, son los que se vienen trabajando en un hospital de las dimensiones del Reina Sofía o el San Juan de Dios, en el que el primero tiene hasta tres almacenes de fármacos que son la fuente de todos los servicios del hospital.

Por tanto, los costes mínimos de seguridad con los que se trabajarán serán más altos de los aquí expuestos, dado que el objetivo no es abaratar costes sin más, sino con el requisito indispensable de que nunca se debe quedar sin fármacos un paciente.

5.2.3 Medicamento 2. Primer pedido

Para el caso del segundo medicamento los cambios a la hora de plantearnos como realizar las comparaciones entre el caso simulado por lo que realizó la petición del producto el farmacéutico y el caso simulado, así como para comprobar la viabilidad del sistema, son idénticos a los de los casos para el *Medicamento 1*.

Los valores que se adoptan en cuanto a coste de almacenamiento del producto, y los costes asociados al personal son los mismos que en el caso anterior. Las variaciones son en cuanto al coste del producto en sí, el coste de oportunidad, y las cantidades en las cuales se puede solicitar este fármaco:

- El coste de la unidad del producto es $2.98€$.
- El coste de oportunidad lo mantenemos al 1% del producto, quedando pues en $0.02€$.
- Las posibilidades de pedidos del mismo son $480, 600, 720\ y\ 1200$.

Con todo esto procedemos a realizar la estimación de la misma manera que en los apartados anteriores.

Así pues los costes asociados al stock del *Medicamento* 2 serán:

- Según el pedido que realizo el hospital de $J = 4571.57$
- Según los cálculos realizados por el programa con un día de pedido $J = 2168.61$.
- Según los cálculos realizados por el programa sin días de pedido $J = 702.85$.

En este caso, sucede lo mismo que en el apartado 5.2.1, que realmente dentro del horizonte analizado y con la estimación realizada en base a la media de pedidos de las anteriores $52\ semanas$, pues no es necesario realizar un pedido de las características que el programa nos indica que hay que hacer.

Por otro lado, hay que tener presente que por parte del hospital se realizaron dos pedidos $600\ y\ 600$, mientras que el ordenador ofrece una solución pidiendo 480 en un solo pedido.

Ahora procedemos a comprobar los costes conforme a la demanda real de esos $14\ días$ analizados, obteniendo los siguientes:

- Los costes de la gestión realizada por el hospital son $J = 4390.72$.
- Los costes de gestión en caso de haber realizado un pedido son $J = 1987.76$.
- En caso de no haber realizado ningún pedido, tal y como indicaba la computadora $J = 552.00$.

En este caso, nuevamente vuelve a ser la mejor opción no realizar ningún pedido.

5.2.4 Medicamento 2. Segundo pedido

Analizamos ahora un periodo distinto de información para comprobar si el sistema que hemos implementado con nuestro programa supone una mejora frente a la práctica actual.

Todos los datos del fichero se han introducido en consonancia con lo expresado en epígrafe anterior.

Los resultados que se han obtenido de las simulaciones usando la estimación aplicada por parte del hospital son los siguientes:

- Según el cálculo realizado por el personal farmacéutico $J = 8474.78$.
- Según el cálculo realizado por el ordenador con un pedido $J = 1836.46$.
- Según el cálculo realizado por el ordenador sin pedidos $J = 1853.50$.

En este caso, se ha forzado nuevamente la realización de un pedido por los mismos motivos que indicábamos en el epígrafe 5.2.1. Aquí nuevamente se supera el límite de seguridad del stock que se dispone del fármaco *Medicamento 2* por lo que se debe realizar un pedido de urgencia.

En este caso también hay que tener presente que son tres los pedidos que realizó el hospital durante ese periodo, siendo además de $600, 600\ y\ 1200$, frente a un pedido de 480 que es lo que obtenemos en ambos casos con el algoritmo de este Proyecto Fin de Carrera.

Aquí es donde además vemos de forma significativa como al modelar así el caso real, nos puede prevenir de imprevistos como el que aquí ocurriría en caso de no realizar ningún pedido, evitándonos también otros costes además de los propios del medicamento.

Para contrastar estos datos con el caso real, procedemos ahora a cambiar la estimación por la demanda real del producto en aquellas fechas. Los resultados obtenidos son los que se indican a continuación:

- Los costes de gestión fueron aproximadamente de $J = 8307.58$.
- Los costes en caso de haber realizado un pedido del producto tal y como se cálculo serían $J = 1741.26$.
- Los costes en caso de no haber realizado ningún pedido son $J = 1758.30$.

Nuevamente el coste de no realizar pedidos es mayor debido a que no es una opción válida realmente, ya que no previene de los costes debidos.

Vemos que el ahorro estimado y con los parámetros que hemos aproximado para las simulaciones, el ahorro que se da es significativo.

5.3 Aclaraciones

Es necesario aclarar que este cápitulo se basa en una serie de asunciones, como explicamos al principio del apartado 5.2 que nos son de gran utilidad a la hora de comparar la gestión que se ha estado realizando por parte del hospital San Juan de Dios frente a la que se podría haber realizado en caso de utilizar este sistema, referiendome aquí al sistema global.

El análisis en términos económicos de ahorro del sistema no se ha realizado, puesto que este Proyecto Fin de Carrera es sólo una de las partes del sistema global, en la cual no se entra a valorar otros apartados que estarán dentro del mismo.

Es por eso, que hemos realizado todas las pruebas utilizando las estimaciones que se están utilizando en la actualidad en la práctica diaria de los hospitales. Aún así, queda patente que utilizar un sistema de estas características para la optimización del inventario de fármaco hospitalario es muy ventajoso, y el ahorro que supone será aún mayor cuando las estimaciones que se implementen en el sistema final sean más finas y justas analizando y aplicando los métodos del Proyecto "Optimización de Estimación de la Demanda en Servicio Farmacéutico Hospitalario".

Con todo esto, queda para el siguiente cápitulo mostrar las mejoras y perspectivas de futuro, tanto del presente Proyecto Fin de Carrera, como del sistema global, que permitan a nuestros hospitales dotarse de unas herramientas que abaraten de manera notable los costes que presentan en la actualidad.

6 PERSPECTIVAS Y MEJORAS

Habiendo alcanzado el objetivo del proyecto, no es en vanom ni tampoco baladí, platear posibles mejoras del mismo. Todas estas posibilidades entrarían en juego en futuras versiones que se realicen del sistema final, y se tiene en cuenta todo lo desarrollado en este programa. De hecho, en lo práctico se tratraría de programar y desarrollar diferentes módulos que hagan uso de la ejecución de este programa tal y cómo se encuentra finalmente.

Las mejoras pueden ser en cuanto al planteamento de las condiciones impuestas al problema, extensiones u otros usos por los que se pueda sacar provecho al programa en sí.

6.1 Cálculo de coste con farmacéuticas

A la hora de realizar el cómputo del coste de un determinado fármaco, añadimos el coste relacionado con las peticiones. Este coste se debe principalmente a un coste de recursos humanos necesarios para recibir y almacenar los productos que se reciben.

Si tuviésemos en cuenta las distribuidoras farmacéuticas que entregan los fármacos del hospital, podríamos realizar una optimización del coste de petición, puesto que podríamos realizar todos los pedidos a una distribuidora de forma simultánea en el tiempo, reduciendo estos costes de petición hasta en el número de fármacos que se consumen de ella.

Esta mejora supone modificaciones en el programa en cuanto a la evaluación de la función objetivo:

- Tendremos que modificar la forma en la cual se cálcula la función de coste, puesto que ahora deberá haber N_i iteraciones por cada farmacéutica, siendo N_i el número de fármacos que se solicitan a una distribuidora en concreto.
- En cuanto a la gestión de datos, se deberá leer los datos del proveedor de un fichero, por ejemplo "farma.pha", en el cual deben venir el número de fármacos que se solicitan a esa distribuidora, así como un listado con los nombres de los ficheros en los cuales se almacena la información de cada uno de los medicamentos.
- Para realizar la obtención de las cantidades de pedido, habrá que pasar el vector con la posible solución sólo con valores '1' o '0', es decir, se pide o no se pide. Después para cada fármaco habrá que explorar cual es la combinación de peticiones en esos días que genera un menor coste, siendo este el coste que se añada al coste total de la farmacéutica.

- A la hora de mostrar por salida estándar los datos de las peticiones que se van a realizar, deberá quedar claro el orden de los fármacos, así como las cantidades a solicitar de cada uno de ellos, en la fecha determinada.

Las complicaciones que tiene este proceso desde el punto de vista computacional y de la programación es que para cada posibilidad de día de pedido, se tienen que barajar todas las posibles cantidades a pedir de cada uno de los fármacos. Esto va aumentando de forma directamente proporcional el número de posibilidades a barajar, y por tanto la carga computacional del programa, con respecto al programa realizado en este proyecto.

6.2 Cálculo de coste global

Basándonos en el apartado anterior, se podría realizar un cómputo global del coste que tiene un hospital o una farmacia del mismo.

El proceso sería de encerrar en un bucle, con un número de iteraciones igual al número de distribuidoras farmacéuticas, la operación anterior.

Este programa tendría pocos cambios con respecto el anterior apartado, ya que podrían realizarse un número de llamadas al programa desde otro programa ofreciendo cada vez unos datos distintos a través del fichero "farma.pha".

La complicación principal que podría tener este programa, es que habría que clarificar la forma en la que se obtienen los resultados de la ejecución por la salida estándar. Esto habrá de ser así puesto que habría que indicar los días en los cuales se van a realizar pedidos a qué proveedor y las cantidades de cada uno de los fármacos, que se deseasen.

Lo positivo de esta mejora es que, con una sola ejecución, tendríamos ya todos los pedidos a realizar, lo cual podría servir de antesala a un sistema más automatizado.

Habría que tener algún sistema para comunicarnos con los proveedores y realizar los pedidos de forma autónoma quitando esta tarea a los técnicos de la farmacia.

6.3 Cálculo del número de días de pedido

El sistema que hemos implementado en este proyecto recibe como entrada el número de días de horizonte en los que se realizará el cálculo, y el número de días en los cuales se quieren realizar pedidos.

Con este programa, tal y como está, se puede realizar un programa que se encuentre una capa por encima y haga uso de éste para obtener tanto el número óptimo de días de pedido, como obtener finalmente las fechas en las cuales habría que realizar los pedidos, con sus respectivas cantidades para un fármaco concreto.

Se ha llevado a cabo el desarrollo de este programa que se encuentra en el fichero *calcDiasPedidosOpt.c*.

El proceso que lleva a cabo este programa es el que se muestra en el siguiente diagrama:

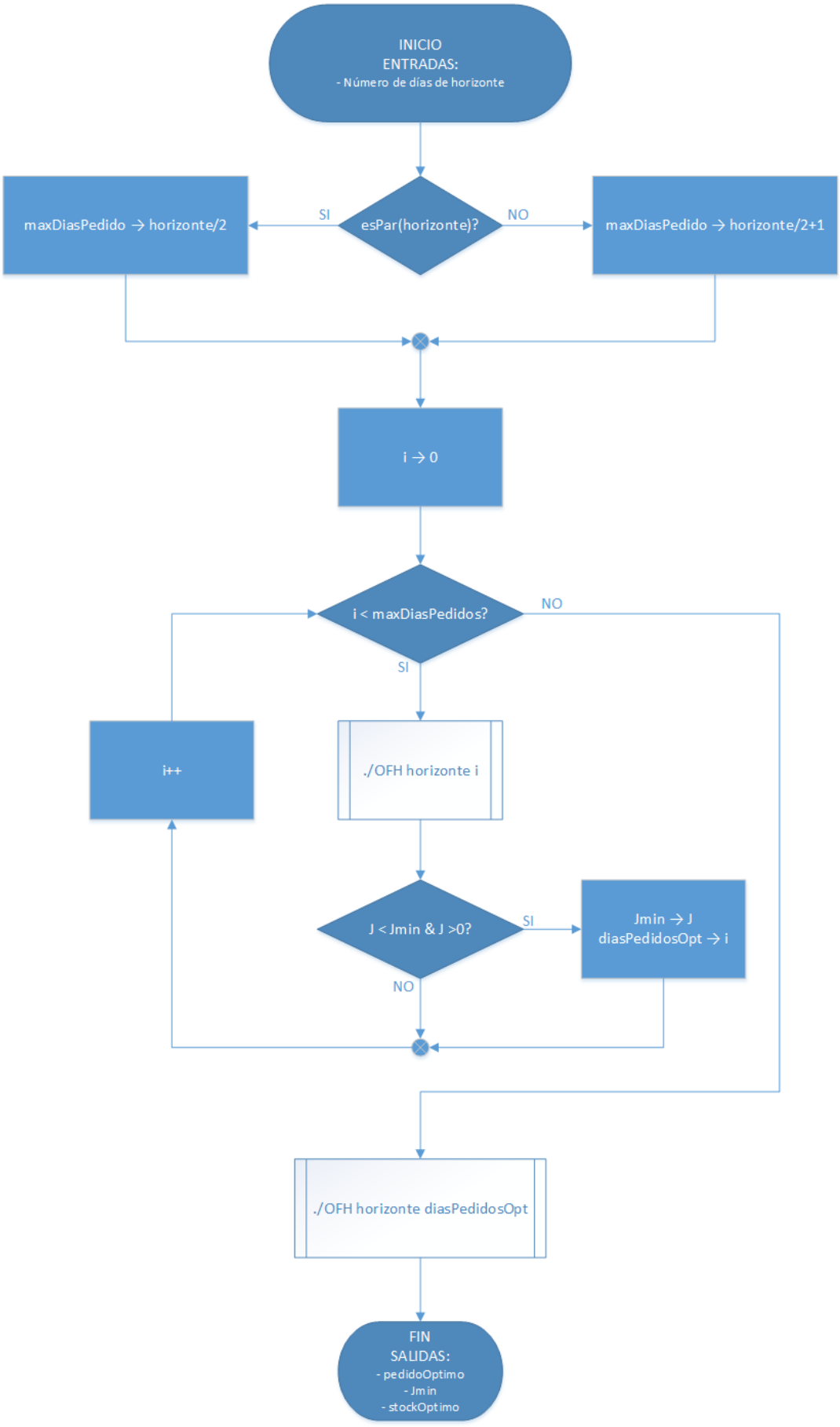

Diagrama de flujo 6-1 Algoritmo para obtener los pedidos óptimos en función de los días de horizonte

Como puede verse en el diagrama de flujo el proceso de cálculo del número de días de pedido se basa por completo en las llamadas al ejecutable del programa desarrollado en este programa:

- En primer lugar se calcula el número de días máximo de pedidos en función del número de días de horizonte. Esto se hace para saber cuantas iteraciones debemos realizar, teniendo en cuenta que se descartan todas las opciones que supone realizar un pedido del fármaco dos días consecutivos.
- Se realiza un bucle con un número de iteraciones igual al número de días máximo en los que se puede realizar pedido en función de las condiciones impuestas.
- En cada iteración se hace una llamada al programa realizado en este proyecto con un número de pedidos que va ligado a la iteración en la que nos encontramos.
- El programa nos devuelve el valor J, que será negativo en caso de haber algún error, o positivo en caso de haber encontrado una solución posible.
- Este valor se compara con el mínimo almacenado. En caso de ser menor, se almacena en una variable `minDiasPedido` el número de días de pedido, y se actualiza el valor de `Jmin` que teníamos hasta ese momento. En caso contrario no se hace nada descartando el resultado obtenido.
- Por último, una vez hallado el número de días de pedido que optimiza la función objetivo, se realiza una última ejecución para sacar por salida estándar los resultados correspondientes, es decir, cuándo y cuánto hay que pedir del fármaco.

Este programa ofrece una solución más avanzada pero en la línea de lo trabajado durante este proyecto, puesto que ya sigue cálculando la mejor posibilidad de pedido, pero recibiendo sólo como parámetro el número de días en el horizonte en el cual se pretende realizar el cálculo.

6.4 Automatización del proceso

Con una visión algo más en perspectiva, el sistema global debería avanzar para automatizar otros procesos ligados a la gestión de inventario de farmacia hospitalaria.

Una de las posibilidades a priori, sería ver como automatizar la gestión del stock en sí, es decir, que el control de las entradas y salidas de fármacos se controlase de forma automática.

Se están trabajando en sistemas que se encarga de reconocer y ordenar cajas que podrían servir para mantener un control de cuatnto entra y sale, a partir de algún tipo de codificación, ya sea código de barras, código QR o por lectura del embalaje de los paquetes.

En cuanto a las salidas, actualmente es el farmacéutico el encargado de llevar un recuento de que productos se están dispensando con sus respectivas cantidades. Se dan situaciones así, en las que si el farmacéutico no registra en el momento la cantidad del producto entregada, nos encontramos ante un error entre el stock que tenemos registrado y el stock real con el cual contamos.

Para solucionar esto, se podría trabajar con un inventario automático al que el farmacéutico le solicite la cantidad del fármaco que necesita dispensar, y así, registrarlo de una forma totalmente automática.

Ligando estas mejoras con lo expuesto en el apartado Cálculo de coste global, podríamos tener un sistema que se comunicase directamente con las distribuidoras de fármacos, y a su vez, llevar una gestión automatizada en lo interno de la farmacia, dónde periódicamente se realizan los cálculos de cuanto hay que solicitar de los productos en el siguiente periodo.

Esto se puede seguir desarrollando, pero siempre va a ir en la línea de ir reduciendo el trabajo que realizan los farmacéuticos del hospital, es decir, la interacción humana, línea que se ha venido siguiendo en lo que a procesos de fabricación industrial se refiere, y que ahora empieza a darse dentro del sector servicios.

7 CONCLUSIONES

El objetivo que nos marcamos a principio del proyecto se ha alcanzado tal y como estaba previsto. Hemos desarrollado un programa que realiza de forma automática el cálculo de los días óptimos de pedido y las cantidades a solicitar. Una de las partes importantes de este proyecto es poder modelar la función de coste. Esta función tendrá que tener una representación clara de la realidad, teniendo en cuenta todos los factores que pueden influir en el coste total de recibir, almacenar y dispensar un cierto producto. Este apartado del proyecto que se ha realizado también de una forma espcecifica teniendo en cuenta las necesidades del inventario de farmacia hospitalaria. Todo esto puede llegar a generalizarse para llevar a cabo la gestión del inventario de cualquier otro producto.

En primer lugar tenemos que ver si son válidas las asunciones que realizamos al principio de este Proyecto Fin de Carrera.

- Por un lado, las entradas que debe tener el programa para funcionar, siendo el número de días de horizonte, así como el número de días de pedido. Esto mantiene el control real de las operaciones a realizar en manos del usuario final, el farmacéutico.
- El no realizar pedidos de un fármaco dos días consecutivos, lo cual nos permite eliminar muchas posibilidades a priori, lo cual facilita el cálculo y descarga de tiempo el proceso en sí.
- También se ha implementado correctamente la opción de indicar mediante la entrada estándar aquellas fechas en las cuales se deba descartar la posibilidad de pedido.

Con todo esto se han generado todas las posibilidades, de una formá sistématica y ordenada en una sola variable que sería la MatrizFin que se indica en el Diagrama de flujo 3-10 Algorimto para obtener la matriz final

Con esta MatrizFin, se van evaluando una a una sus filas en la función objetivo, que nos permite evaluar el coste que supondría trabajar con cada una de las posibilidades.

Con esta función se ha tratado de reproducir de la forma más aproximada posible, todos los costes asociados a la gestión del stock de un cierto fármaco concreto.

Los costes que se han tenido en cuenta han sido fundamentalmente tres:

- Costes asociados al medicamento. Estos serían tanto el valor del medicamento en sí, como el valor de almacenar un determinado medicamento.
- Costes asociados a la manipulación del medicamento. Estos serían tanto los costes de solicitar un

cierto medicamento, como el coste de recogida y almacenamiento.

- Coste de oportunidad. Se modela este coste, puesto que al tener medicamentos almacenados, estamos mantiendo almacenado un capital que se podría estar aprovechando en algún otro gasto o inversión del hospital, así como del sistema sanitario en general.

Con todo esto se ha implementado el programa que, teniendo presente siempre que opción será la que nos devuelva un menor coste, facilita de forma clara al usuario las cantidades que debe solicitar de un cierto fármaco, así como las fechas en las que tiene que realizar dichos pedidos.

El hecho de haber planteado así el proyecto se debe a haber modulado su realización, permitiendo la implementación de cada una de las partes de forma independiente. Esto es importante mencionarlo puesto que se pueden plantear refactorizaciones del programa que agilicen en cierta medida el cálculo del mismo, obteniendo en la práctica un programa similar.

Aún así, teniendo en cuenta que el cuello de botella de un sistema automatizado de estas características se debe al error cometido en la estimación de la demanda a priori, el resultado cubre los casos de uso a los cuales va a estar destinado que serán cálculos sobre un horizonte de entre siete y quince días.

Esto último se ha limitado con un máximo de para evitar errores que no se puedan gestionar, y produzcan violaciones de segmento, algo del todo indeseable para su uso práctico por un farmacéutico.

Por último, destacar que, contando con una misma estimación, este proyecto mejora de forma notable la gestión del inventario de fármacos de nuestros hospitales, como vimos en el Ejecuciones y resultados

El ahorro, será mejor conforme más exacto sea el método que realice la estimación. Puesto que este sistema se fundamenta en obtener el mejor resultado de todos los posibles ante una situación futura, si esta estimación se aleja de la realidad no será viable la solución obtenida, pudiendo suponer un problema real, en el cual se aumenten los costes, o se dejen sin la atención necesaria a pacientes que lo necesiten.

BIBLIOGRAFÍA

[1] Parra Guerrero, F., 2005. "Gestión de stocks". 3ª ed. Editorial ESIC, Madrid

[2] Guerrero Salas, H., 2010. "Inventarios. Manejo y control". Editorial Starbook, Madrid

[3] Maestre Torreblanca, J.M., Isla Tejera, B., Fernández García, M.I., del Prado Llergo, J.R., Álamo Cantarero, T., Fernández Camacho, E., 2011. "Análisis y minimización del riesgo de rotura de stock aplicado a la gestión en farmacia hospitalaria". Farm Hosp. 2012; 36(3):130-134

[4] García Collado, C., Madrid Paredes, A., Jiménez Morales, A., Calleja Hernández, M.Á., 2012. "Mejoras en las consultas de pacientes externos tras la implantación de un robot automático de dispensación". Farm Hosp. 2012; 36(6):525-530

[5] Brewer, A.M., Button, K.J., Hensher, D.A., 2001. "Handbook of Logistics and Supply-Chain Management". Editorial: Pergamon, Hungary

[6] Maestre, J.M., Zafra Cabeza, A., Fernández García, M.I., Isla Tejera, B., del Prado, J.R., Camacho, E.F., 2012. "Control predictivo aplicado a la gestión de stocks en farmacia hospitalaria: un enfoque orientado a la minimización del riesgo". Editorial ELSEVIER

[7] Fernández, M.I., Maestre, J.M., Cuenca, F., "Impacto económico de la aplicación de técnicas de control predictivo basado en modelo a la gestión de un servicio de farmacia". Presentación para el VII Congreso de la SAFH, Universidad de Sevilla, Sevilla.

[8] Alós Almiñana, M., 2004. "Concepción de la gestión económica y su aplicación al desarrollo del SFH I". IV Congreso AAFH – Mendoza 2004

[9] Díaz-Maroto Muñoz, M., 1995. "Gestión de stocks del material sanitario en el servicio de farmacia del hospital general penitenciario (I): clasificación y elaboración de una guía de material sanitario". Farm Hosp. 1995; 19(2):105108

[10] Díaz-Maroto Muñoz, M., 1995. "Gestión de stocks del material sanitario en el servicio de farmacia del hospital general penitenciario (II): informatización y aplicación de la clasificación ABC al análisis del consumo". Farm Hosp. 1995; 19(4):165-168

[11] Fernández Quesada, I., 2006. "Gestión de inventarios. Modelos determinísticos". Universidad de Oviedo, Oviedo

[12] González Casimiro, M.P., 2009. "Análisis de series temporales: modelos ARIMA". Universidad del País Vasco, País Vasco

[13] Muñiz Corral, J., 2010. "Predicción del precio de la electricidad mediante redes neuronales". Universidad Pontificia Comillas, Madrid

[14] Castro Zuluaga, C.A., 2003. "Una estructura para la selección de modelos de gestión de inventarios de artículos individuales cuando la demanda es determinística". Universidad EAFIT, Colombia

[15] López Ramírez, A.J., 2014 “Optimización de estimación de la demanda en servicio farmacéutico hospitalario”. Universidad de Sevilla, Sevilla

[16] Pérez Navarro, A.J., Medina León, A., Alonso Elizondo, P., Ramírez Pérez, N., 2007. “Métodos y técnicas para la previsión de la demanda”. Universidad de Matanzas, Cuba

[17] Ramos, A., 2008. “Optimización de gestión de inventarios (stocks)”. Universidad Pontificia Comillas, Madrid

[18] Toledano Garrido, N., 2006. “Planificación, gestión y control de la producción II: La gestión clásica de inventarios”. Universidad de Huelva, Huelva

ANEXO I. HERRAMIENTAS UTILADAS

Para el desarrollo de este proyecto se ha estrado trbaajando con las siguientes herramientas:

- Máquina virtual:

  Se ha utilizado un sistema operativo Ubuntu versión 12.04, corriendo en un entorno gracias a la aplicación Oracle VM VirtualBox.

  Esto ha supuesto algunas limitaciones en cuanto a las pruebas funcionales del proyecto, puesto que no dispone de toda la potencia del ordenador, pudiendo variar considerablemente la ejecución de este programa en otros dispositivos.

- Editor de textos:

  Al estar trabajando con programación C, no es necesario tener ningún entorno de programación avanzado, tipo Eclipse.

  Se ha optado por el editor de textos Sublime Text 2 (en su versión de prueba), el cual incluye diversas funcionalidades que agilizan la programación, el trabajo con proyectos de varios ficheros y reconocimiento de tipos de lenguajes.

- Repositorio:

  Para mantener siempre una copia de seguridad, así como llevar a cabo el control de versiones del proyecto, se ha trabajado con un repositorio gratuito en GitHub.

  Para poder trabajar de una forma cómoda, se planteó en primer lugar trabajar con SourceTree. El inconveniente de este cliente para repositorios es que no tiene versión operativa para Linux. Finalmente se ha trabajado con SmartGit, cliente que tiene las mismas funcionalidades que SourceTree y algunas añadidas útiles para la depuración del código del repositorio de forma directa.

  A la hora de trabajar con el repositorio se debe ser meticuloso para no cometer errores que no hagan ni práctico ni útil llevar un control de versiones. A fin de evitar esto, se ha seguido la metodología de ramas y trabajo propuesta por GitFlow:

  - Se mantenía una rama master limpía, a la cual sólo se pasaría una versión definitiva y estable del proyecto.
  - Los diferentes desarrollos se han ido incluyendo sobre la rama develop.
  - Cada vez que queremos añadir una nueva funcionalidad al programa que estamos desarrollando se ha abierto una rama feature que se ha cerrado al ser completamente funcional.

- Compilador:

  Al trabajar con Lenguaje C es necesario la utilización de un compilador que nos permita obtener el ejecutable del proyecto. Se ha utilizado el compilador gcc disponible desde cualquier SO Linux y ejecutable a través de la terminal.

  Además, con la implementación del fichero makefile, se simplifica considerablemente la compilación de todos los ficheros que se han ido implementado.

Así pues, todo el proceso de compilación se realiza con una única llamada al comando makefile, en la cual se tendrá que encontrar la terminal en el directorio que contiene todos los ficheros, así como el mismo makefile, y llamar al comando *make*.

Con esto se genera el fichero ejecutable que después será utilizado por la capa de administración del sistema.

ANEXO II. GESTIÓN DE ERRORES

A la hora de la realización de una aplicación de estas carácterísitcas, es siempre positivo que se tenga en cuenta una gestión de los diferentes errores que se pueden dar durante la ejecución del programa.

En función de diferentes posibilidades de ejecución del programa, así como de los resultados que puede dar a posteriori, estos se han tipificado cada uno con un valor entero negativo, que será el valor que se devuelva por parte del programa en caso de una ejecución externa.

Así pues, si llamamos al programa y esperamos que nos devuelva un valor, habrá que tener en cuenta que si es negativo será un error que vendrá tipificado por su código en este anexo, y si es positivo será un valor válido de cálculo que será el del coste mínimo que se ha hallado con el resultado que se muestra por sálida estándar.

Cada uno de estos errores tendrá también un mensaje asociado por sálida estándar que pretende dar una orientación de donde se encuentra dicho error y como subsanarlo.

Los errores que se han tipificado son los siguientes:

- Código de error -1:

  ```
  ERROR1: Numero de argumentos de la funcion incorrectos

  La llamada al programa debe ser "nombre del ejecutable" "numero de dias en el horizonte" "numero de pedidos en el horizonte"
  ```

  Este error sale cuando el número de argumentos introducidos en la línea de comandos al ejecutar el programa es menor del esperado.

- Código de error -2:

  ```
  ERROR2: Valor de numero de dias en el horizonte incorrecto. Introduzca valor numérico
  ```

  Error por introducir caracteres alfabéticos en lugar de númericos para los días de horizonte de cálculo.

- Código de error -3:

  ```
  ERROR3: Valor de numero de pedidos en el horizonte incorrecto. Introduzca valor numérico
  ```

  Error por introducir caracteres alfabéticos en lugar de números para los días pedido solicitados.

- Código de error -4:

  ```
  ERROR4: Numero de dias de pedido no válido.

  Debe ser cómo máximo "horizonte/2+1" para un horizonte impar, o "horizonte/2" para un horizonte par
  ```

  Por las restricciones introducidas en nuestro programa, no es posible que haya dos días de pedidos consecutivos, por lo que esta opción siempre devolverá que no existe ninguna posibilidad.

- Código de error -5:

  ```
  ERROR5: Fechas introducidas incorrectos. Utilizar la siguiente notacion:

  dd/mm/yyyy
  ```

Se trata de un error de notación a la hora de introducir las fechas de los días de no pedidos que imposibilitaría una buena gestión de las mismas para algunos casos.

- Código de error -6:

  ```
  ERROR6: Fecha incorrecta, fuera del horizonte
  ```

  La fecha que se ha introducido se encuentra fuera del horizonte por lo cual, se interpreta como un error que se haya introducido, entendiendo que el usuario ha querido especificar una fecha que si afecta al cálculo.

  NOTA: Es menester recordar que no es necesario introducir ninguna fecha para la ejecución del programa, pero si se ha habilitado dicha posibilidad por lo expuesto en el proyecto.

- Código de error -7:

  ```
  ERROR7: Lectura de fichero no realizada
  ```

  Se ha producido un error durante la lectura del fichero *datos.pha* con toda la información del medicamento, lo cual impide que continue la ejecución del programa.

  Este error se puede deber a la no existencia del fichero en el directorio en el que se encuentre el ejecutable.

ANEXO III. DATOS DE LOS FÁRMACOS

Medicamento 1. Salidas 2012

Fecha	Cantidad	Fecha	Cantidad
10/01/2012	3	20/06/2012	5
21/01/2012	6	20/06/2012	3
01/02/2012	7	26/06/2012	3
01/02/2012	6	27/06/2012	3
07/02/2012	9	07/07/2012	3
10/02/2012	5	17/07/2012	2
16/02/2012	7	19/07/2012	3
20/02/2012	1	26/07/2012	3
24/02/2012	6	07/08/2012	3
14/03/2012	3	09/08/2012	3
16/03/2012	4	16/08/2012	2
20/03/2012	3	21/08/2012	3
29/03/2012	2	28/08/2012	2
03/04/2012	3	18/09/2012	3
04/04/2012	3	02/10/2012	2
09/04/2012	15	04/10/2012	8
14/05/2012	3	09/10/2012	15
06/06/2012	8	30/10/2012	3
19/06/2012	1	30/10/2012	6
19/06/2012	10	14/11/2012	3

Medicamento 1. Salidas 2013

Fecha	Cantidad	Fecha	Cantidad
17/01/2013	8	25/02/2013	6
30/01/2013	3	27/02/2013	5
31/01/2013	3	05/03/2013	1
01/02/2013	6	06/03/2013	3
06/02/2013	4	13/03/2013	3
20/02/2013	3	20/03/2013	9
22/02/2013	1	05/04/2013	2
25/02/2013	6	10/04/2013	8
27/02/2013	5	16/04/2013	1
05/03/2013	1	18/04/2013	3
06/03/2013	3	24/04/2013	8
13/03/2013	3	03/05/2013	8
20/03/2013	9	08/05/2013	1
05/04/2013	2	15/05/2013	10
10/04/2013	8	22/05/2013	3
16/04/2013	1	28/05/2013	3
18/04/2013	3	29/05/2013	5
24/04/2013	8	05/06/2013	2
03/05/2013	8	11/06/2013	3
17/01/2013	8	11/06/2013	2
30/01/2013	3	12/06/2013	12
31/01/2013	3	25/06/2013	3
01/02/2013	6	08/07/2013	5
06/02/2013	4	23/07/2013	8
20/02/2013	3	30/07/2013	2
22/02/2013	1	07/08/2013	3

13/08/2013	2
21/08/2013	3
22/08/2013	2
29/08/2013	3
29/08/2013	4
04/09/2013	11
25/09/2013	7
02/10/2013	2
04/10/2013	6
14/10/2013	2
16/10/2013	12
05/11/2013	11
20/11/2013	4
22/11/2013	2
27/11/2013	4
03/12/2013	3
04/12/2013	3
11/12/2013	3
18/12/2013	4
23/12/2013	1
24/12/2013	3

Medicamento 1. Recepciones 2012

Fecha	Cantidad	Fecha	Cantidad
21/01/2012	25	27/06/2012	25
21/01/2012	25	01/08/2012	25
24/02/2012	25	08/10/2012	25
28/03/2012	25	08/11/2012	25
18/04/2012	20	14/12/2012	25

Medicamento 1. Recepciones 2013

Fecha	Cantidad	Fecha	Cantidad
03/01/2013	25	09/08/2013	25
05/02/2013	25	02/09/2013	25
11/03/2013	25	30/09/2013	25
12/04/2013	25	18/10/2013	25
07/05/2013	25	26/11/2013	25
31/05/2013	25	19/12/2013	25
02/07/2013	25		

Medicamento 2. Salidas 2012

Fecha	Cantidad	Fecha	Cantidad
18/05/2012	120	01/10/2012	360
07/06/2012	120	15/10/2012	120
23/06/2012	120	16/10/2012	240
29/06/2012	120	18/10/2012	120
09/07/2012	240	29/10/2012	120
19/07/2012	120	29/10/2012	240
23/07/2012	240	29/10/2012	240
23/07/2012	120	31/10/2012	120
25/07/2012	120	02/11/2012	120
30/07/2012	120	12/11/2012	120
06/08/2012	120	15/11/2012	240
09/08/2012	120	27/11/2012	120
10/08/2012	120	29/11/2012	120
21/08/2012	120	29/11/2012	120
23/08/2012	120	30/11/2012	60
24/08/2012	240	03/12/2012	120
29/08/2012	120	03/12/2012	120
30/08/2012	120	04/12/2012	120
03/09/2012	120	04/12/2012	240
11/09/2012	120	13/12/2012	240
11/09/2012	120	18/12/2012	120
20/09/2012	120	27/12/2012	120
26/09/2012	120	28/12/2012	120
28/09/2012	120		
01/10/2012	120		

Medicamento 2. Salidas 2013

Fecha	Cantidad	Fecha	Cantidad
08/01/2013	120	16/04/2013	120
10/01/2013	240	17/04/2013	240
10/01/2013	120	23/04/2013	120
11/01/2013	240	25/04/2013	120
31/01/2013	240	02/05/2013	240
01/02/2013	120	06/05/2013	120
02/02/2013	120	16/05/2013	120
05/02/2013	120	17/05/2013	240
12/02/2013	240	20/05/2013	120
19/02/2013	120	22/05/2013	120
19/02/2013	120	26/05/2013	120
27/02/2013	240	28/05/2013	120
07/03/2013	120	29/05/2013	120
12/03/2013	120	06/06/2013	240
13/03/2013	120	07/06/2013	120
15/03/2013	240	13/06/2013	240
20/03/2013	120	13/06/2013	120
20/03/2013	1	17/06/2013	120
22/03/2013	120	24/06/2013	240
22/03/2013	119	24/06/2013	120
05/04/2013	120	25/06/2013	120
05/04/2013	120	27/06/2013	120
09/04/2013	120	04/07/2013	120
15/04/2013	120	08/07/2013	120
15/04/2013	120	09/07/2013	120
15/04/2013	120	16/07/2013	120

16/07/2013	120	16/07/2013	119
16/07/2013	1	19/07/2013	120
16/07/2013	119	22/07/2013	120
19/07/2013	120	23/07/2013	120
22/07/2013	120	26/07/2013	120
23/07/2013	120	01/08/2013	240
26/07/2013	120	02/08/2013	120
01/08/2013	240	06/08/2013	120
02/08/2013	120	08/08/2013	120
06/08/2013	120	14/08/2013	120
08/08/2013	120	27/08/2013	120
14/08/2013	120	27/08/2013	120
27/08/2013	120	28/08/2013	120
27/08/2013	120	29/08/2013	120
28/08/2013	120	30/08/2013	120
29/08/2013	120	04/09/2013	120
30/08/2013	120	10/09/2013	120
04/09/2013	120	12/09/2013	120
10/09/2013	120	17/09/2013	120
12/09/2013	120	18/09/2013	120
17/09/2013	120	19/09/2013	120
18/09/2013	120	23/09/2013	240
19/09/2013	120	23/09/2013	120
23/09/2013	240	24/09/2013	120
23/09/2013	120	30/09/2013	120
24/09/2013	120	02/10/2013	120
30/09/2013	120	14/10/2013	120
02/10/2013	120	15/10/2013	120

17/10/2013	120
18/10/2013	120
21/10/2013	120
23/10/2013	240
25/10/2013	120
12/11/2013	120
14/11/2013	120
14/11/2013	120
14/11/2013	120
14/11/2013	120
19/11/2013	120
21/11/2013	120
21/11/2013	120
21/11/2013	120
21/11/2013	120
25/11/2013	120
26/11/2013	120
26/11/2013	120
28/11/2013	120
28/11/2013	120
29/11/2013	120
29/11/2013	120
04/12/2013	120
05/12/2013	120
05/12/2013	120
05/12/2013	120
12/12/2013	240
12/12/2013	120
12/12/2013	120
16/12/2013	120
17/12/2013	120
17/12/2013	120
20/12/2013	120
21/12/2013	120
26/12/2013	120
27/12/2013	240
27/12/2013	120
27/12/2013	240
27/12/2013	120
30/12/2013	120

Medicamento 2. Recepciones 2012

Fecha	Cantidad	Fecha	Cantidad
10/05/2012	240	06/09/2012	720
11/06/2012	240	09/10/2012	720
11/06/2012	120	23/10/2012	720
05/07/2012	360	06/11/2012	720
17/07/2012	480	02/12/2012	480
26/07/2012	480	11/12/2012	360
08/08/2012	600	11/12/2012	360
28/08/2012	720	21/12/2012	720

Medicamento 2. Recepciones 2013

Fecha	Cantidad	Fecha	Cantidad
14/01/2013	480	12/08/2013	600
06/02/2013	480	02/09/2013	600
21/02/2013	480	18/09/2013	600
14/03/2013	480	02/10/2013	600
21/03/2013	720	16/10/2013	600
17/04/2013	720	25/10/2013	600
02/05/2013	600	17/11/2013	600
20/05/2013	720	25/11/2013	600
30/05/2013	720	29/11/2013	600
13/06/2013	600	12/12/2013	600
28/06/2013	720	13/12/2013	600
22/07/2013	600	19/12/2013	1200
01/08/2013	600		