Trabajo Fin de Máster

Reposición y Redistribución de Inventarios: Gestión y Optimización

Autora: Irene Llana García

Director: Julio González Díaz

Máster en Técnicas Estadísticas

Curso 2016-2017

Propuesta de Trabajo Fin de Máster

Título en galego: Reposición e Redistribución de Inventarios: Xestión e Optimización
Título en español: Reposición y Redistribución de Inventarios: Gestión y Optimización
English title: Inventory Replenishments and Transfers: Management and Optimization
Modalidad: Modalidad A
Autor/a: Irene Llana García, Universidade de Santiago de Compostela
Director/a: Julio González Díaz, Universidade de Santiago de Compostela;
Tutor/a:
Breve resumen del trabajo: Este informe presenta una mayor profundización y validación de un proceso de redistribución de mercancías sobre una red de tiendas que se presentó en otro TFM realizado en septiembre de 2014. Dicho proceso se basa minimizar los costes de envío de paquetes, resolviendo consecutivamente un problema de flujo en redes, uno de redondeo de matrices y uno de empaquetado.
Recomendaciones:
Otras observaciones:

Agradecimientos

En primer lugar quisiera agradecer a Julio González la oportunidad que me ha brindado para continuar con este interesante trabajo, y por su excelente supervisión.

A Jorge Rodríguez por sus consejos sobre el uso de AMPL, por la ayuda proporcionada con las traducciones y por estar siempre dispuesto a colaborar.

A Ángel González por su ayuda con el ajuste de los parámetros de *Gurobi*.

A Patricio Reyes por su incansable curiosidad y encontrar siempre herramientas interesantes.

Al Departamento de Matemática Aplicada y a ITMATI por permitirme usar las licencias de AMPL y *Gurobi* en sus máquinas, Mariscal y Mares.

A C.S.M por el interés mostrado en el proyecto y por facilitarme una base de datos real.

A mis amigos y compañeros, Jon Urrestarazu, Oana Chis y Noemí Esteban por estar siempre presentes.

Gracias.

Índice general

Resumen ix

1. Introducción 1

2. Modelado inicial 5

2.1. Descripción del problema 5

2.2. Conjuntos, parámetros y variables del problema 6

2.2.1. Conjuntos 6

2.2.2. Parámetros 6

2.2.3. Variables 7

2.3. Modelo matemático 8

2.3.1. Planteamiento inicial 8

2.3.2. Subproblema de flujo en redes 10

2.3.3. Subproblema de redondeo de matrices 11

2.3.4. Subproblema de empaquetado 13

2.4. Programación y resultados 14

3. Profundizando en el modelo 17

3.1. Fase 0: Transferring relajado 18

3.2. Función objetivo del Transferring 19

3.3. Redondeo de matrices 21

3.3.1. Planteamiento 21

3.3.2. Implementación 23

3.3.3. Comparación con el modelo antiguo 25

3.4. Resolución exacta del problema de empaquetado 25

4. Modelo Final 29

4.1. Fase 0. Transferring Relajado 29

4.1.1. Modelo 29
4.2. Fase 1. Transferring 30
4.2.1. Modelo 30
4.3. Fase 2. Rounding 31
4.3.1. Modelo 31
4.3.2. Implementación 31
4.4. Fase 3. Packing 32
4.4.1. Modelo 32
4.4.2. Implementación 32
4.5. Algoritmo TRP 33

5. Heurísticas **35**
5.1. Heurísticas tipo 1 35
5.2. Heurísticas tipo 2 36
5.2.1. Heurística 2 a 36
5.2.2. Heurística 2 b 36

6. Interacción con la empresa y adaptación del modelo **39**
6.1. Avance 41
6.2. Adaptación del modelo 43

7. Resultados numéricos **45**
7.1. Soporte informático 45
7.2. Simulaciones 46
7.2.1. Generación de batería de problemas 46
7.2.2. Resultados 47
7.2.3. Comparaciones con las heuristicas 59
7.3. Aplicación a datos reales 60

8. Conclusiones **65**

Bibliografía **67**

Resumen

Resumen en español

Este informe presenta una mayor profundización y validación del modelo de optimización de redistribución de mercancías sobre una red de tiendas, que presentó Pablo Montero Souto en septiembre de 2014 en su trabajo de fin de máster bajo el título "Optimization of Inventory Transfers". Dicho proceso se basaba en minimizar los costes de envío de paquetes entre cada par de tiendas de la red, resolviendo consecutivamente un problema de flujo en redes, un problema de redondeo de matrices y un problema de empaquetado. En este trabajo se realizan mejoras sobre cada una de las fases mencionadas y se proponen además dos heurísticas sencillas para destacar la efectividad del método propuesto. Para la validación del modelo, se prepara una batería de problemas mediante el software R y el modelado del problema se ha hecho con el lenguaje de programación AMPL, siendo *Gurobi* el solver utilizado para la optimización. Finalmente, se muestra la aplicación del método a un caso real para una cadena de moda gallega.

Resumo en galego

Este informe presenta unha maior profundización e validación do modelo de optimización de redistribución de mercadorías sobre unha rede de tendas, que se presentou Pablo Montero Souto en setembro de 2014 no seu traballo de fin de máster baixo o título "Optimización of Inventory Transfers". Dito proceso baseábase na minimización dos custos de envío de paquetes entre cada par de tendas da rede, solventando consecutivamente un problema de fluxo en redes, un problema de redondeo de matrices e un problema de empaquetado. Neste traballo realízanse melloras sobre cada unha das fases mencionadas e propóñense ademais dous heurísticas sinxelas para destacar a efectividade do método proposto. Para a validación do modelo, preparase unha batería de problemas mediante o software R e o modelado do problema fíxose coa linguaxe de programación AMPL, sendo *Gurobi* o solver empregado para a

optimización. Finalmente, mostrase a aplicación do método a un caso real para unha cadea de moda galega.

English abstract

This report presents a further deepening and validation of a stock redistribution optimization model in a network that was presented by Pablo Montero Souto in September 2014 in his master's thesis titled "Optimization of Inventory Transfers". This process was based on the minimization of package shipping costs between every pair of stores in the network, solving consecutively a network flow problem, a matrices rounding problem and a packing problem. In this work, some improvements in all mentioned phases are presented, as well as two simple heuristics to highlight the effectiveness of the proposed method. For the model validation, a set of problems is prepared using the R software and the modelling has been done with the AMPL programming language, being *Gurobi* the solver used for the optimization. Finally, an application of the method to a real case of a Galician fashion chain is shown.

Capítulo 1

Introducción

El problema de redistribución de inventario es un hecho para muchas empresas con varias tiendas en su red. Cuando la demanda de un cliente no se puede satisfacer se producen costes, por lo que es muy importante evitar quedarse sin existencias. Este problema es aún mayor para aquellas empresas que trabajan en el mundo de la moda, pues a lo largo de un año puede haber varias colecciones, por lo que esa redistribución tiene que hacerse necesariamente cada pocos meses.

Para una gran empresa que tiene cientos de ventas a diario en cada una de las tiendas de su red, este problema puede no resultar de tanta importancia porque cada una de las tiendas recibirá prácticamente todos los días nueva mercancía desde el almacén. Además, el cliente es consciente de esta situación y sabe que si un determinado producto no lo tiene el día que va a comprarlo seguramente lo tenga al siguiente.

Sin embargo, este problema puede ser de mayor importancia para empresas más pequeñas en las que el inventario inicial no es tan elevado. Llegará un punto en el que el almacén no pueda abastecer las demandas porque el inventario inicial esté ya distribuido entre las tiendas, y sean estas últimas las que tengan que realizar esta labor en la medida de lo posible.

En febrero de 2014 Pablo Montero Souto decide ponerse en contacto con una empresa de moda gallega para plantearles un método para la redistribución de su stock en su red de tiendas.

En aquel momento, y en la actualidad, cada vez que una tienda tiene una demanda de un producto del que no dispone, es el almacén el que se lo envía. Si el almacén tampoco dispone

de dicho producto, una tienda que lo tenga deberá enviarlo al almacén para posteriormente ser enviado a la tienda que lo demandaba. Puede considerarse como una red en la que todos los nodos están conectados con un único nodo central pero no entre ellos (en el caso de que hubiera varios almacenes todos los demás nodos estarían conectados a esos nodos centrales).

Desde el punto de vista temporal este reabastecimiento no es efectivo, pues la redistribución centralizada en el almacén necesita 48 horas para procesarse (24 horas el envío de la mercancía al almacén y otras 24 el envío desde el almacén a las tiendas), mientras que una redistribución centralizada necesitaría sólo 24 horas. A cambio, la redistribución centralizada puede requerir un menor envío de paquetes, con lo que puede ahorrar en costes de transporte. Las técnicas desarrolladas en este trabajo permiten valorar hasta qué punto un enfoque puede ser superior al otro y también el potencial de un enfoque intermedio, que combine redistribución centralizada (*reabastecimiento*) con redistribución descentralizada (*transbordos*).

Algunos estudios realizados trabajan con modelos específicos para manejo de inventario teniendo en cuenta reabastecimientos y transbordos [1] [7] [5], y también algunos en particular para el mundo de la moda [2] [3] [8] [4] [12] [11].

(a) Reabastecimiento (b) Transbordo

Figura 1.1: Recolococación de Stock: Reabastecimiento y Transbordo

Cabe mencionar que el Capítulo 2 en el que se hablará del modelado inicial que se propuso por Pablo Montero, es en gran parte un resumen de lo que se hizo en el trabajo anterior, y que las imágenes como las de la Figura 1.1 son similares a las que se utilizaron en dicho trabajo. Posteriormente, en el Capítulo 3 se hablará sobre las dificultades que se encontraron a la hora de reprogramar el modelo y se comentarán las mejoras realizadas sobre el antiguo modelado.

En el Capítulo 4 describirá el modelo definitivo que incorpora las mejoras de las se habla

en el Capítulo 3, y será el modelo que se utilice para hacer los test numéricos. En el Capítulo 5 plantearán dos heurísticas para demostrar la eficacia y consistencia del algoritmo propuesto. En el Capítulo 6 se hablará sobre la interacción con la empresa y de la necesidad de adaptar ligeramente el modelo para hacer una ejecución sobre un caso real.

En el Capítulo 7 se presentarán los resultados numéricos de ejecuciones hechas sobre unos datos simulados, tanto con el algoritmo propio como con las heurísticas, que servirán por una parte como validación del modelo y por otra parte darán información al usuario. También se mostrarán los resultados de una ejecución del modelo adaptado con datos reales proporcionados por la empresa.

Finalmente, en el Capítulo 8 se presentarán las conclusiones y futuro trabajo.

Capítulo 2

Modelado inicial

2.1. Descripción del problema

A principio de temporada, cada una de las tiendas de la red recibe una cantidad determinada de cada producto. A lo largo del tiempo, éstas irán efectuando sus ventas y reduciendo su nivel de stock, por lo que necesitarán cubrir de nuevo los productos que hayan vendido. Al principio, estos productos vendrán suministrados desde el almacén central, pues una gran parte del inventario estará concentrado inicialmente allí, pero en algún momento la mayor parte de él estará distribuido por toda la red.

En ocasiones, algunos de los productos que tienen en un lugar determinado una buena aceptación por parte de los compradores, en otros lugares no la tienen. Esto sugiere que quizás sería más beneficioso que los productos que una tienda no vende adecuadamente, estuvieran en otra en la que el producto ha tenido mejor aceptación por tener más posibilidad de venta.

Por otra parte, los productos con los que se está tratando son prendas de ropa y por tanto se manejan tallas. Es probable que un comprador demande un determinado producto del que en su tienda habitual no tengan su talla. Sin embargo, posiblemente en otra tienda sí que la haya, y por tanto la tienda que está en disposición de dicho producto deberá enviárselo a la tienda que lo demanda.

Por tanto, el problema a resolver es el de hacer la redistribución del inventario de la red de tiendas minimizando el coste de envío de paquetes entre ellas. De otra manera, se trata de determinar qué productos debe enviar una tienda a otra y de qué manera hacerlo.

2.2. Conjuntos, parámetros y variables del problema

A continuación se hará la descripción de los conjuntos, parámetros y variables para el modelado del problema.

2.2.1. Conjuntos

- $\mathcal{I} = \{1, \ldots, I\}$: Conjunto de tiendas que forman la red. El almacén central está contenido en este conjunto.
- $\mathcal{R} = \{1, \ldots, R\}$: Conjunto de productos. Se llamará **producto o artículo** a la combinación de referencia más talla. En caso de que se tuviera una referencia con dos tallas, habría dos productos. Puesto que es una nomenclatura que se utilizará a lo largo de todo el informe, es importante diferenciar entre la **referencia** (que es el modelo de la prenda independientemente de la talla) de lo que se llama **producto o artículo** (combinación de referencia y talla).
- $\mathcal{P} = \{1, \ldots, P\}$: Conjunto de tipos de paquetes disponibles para realizar los envíos.

2.2.2. Parámetros

- $\boldsymbol{Peso_r},\ r \in \mathcal{R}$: Peso de cada uno de los productos (valor positivo).
- $\boldsymbol{Stock_{i,r}},\ i \in \mathcal{I},\ r \in \mathcal{R}$: Unidades disponibles del producto r en la tienda o almacén i (valor ≥ 0).
- $\boldsymbol{FixDem_{i,r}},\ i \in \mathcal{I},\ r \in \mathcal{R}$: Demanda fija del producto p de la tienda i. Es demanda **fija** porque la tienda que la tiene ya ha vendido ese producto lo tenga o no lo tenga disponible en ese momento (se entiende que se puede vender un producto no teniéndolo en tienda cuando el encargado o la encargada verifica que hay stock de ese producto en la red y lo encarga para el cliente). En la práctica, las tiendas que se corresponden con los almacenes no tendrán demandas, y por tanto para ellas este parámetro será igual a 0. La demanda fija siempre ha de satisfacerse.
- $\boldsymbol{VarDem_{i,r}},\ i \in \mathcal{I},\ r \in \mathcal{R}$: Demanda variable del producto p de la tienda i. Para definir la demanda **variable** es preciso definir otros parámetros:
  - $\boldsymbol{TienDem_{i,r}},\ i \in \mathcal{I},\ r \in \mathcal{R}$: Demanda que el vendedor o la vendedora de la tienda i estima que va tener del producto r.

- $\boldsymbol{EmpDem_{i,r}},\ i \in \mathcal{I},\ r \in \mathcal{R}:$ Demanda que el departamento logístico de la empresa estima que la tienda i va tener del producto r.

Así, se puede definir la demanda variable como una ponderación de estos dos parámetros:

$$\boldsymbol{VarDem_{i,r}} = \frac{\alpha_i\ TienDem_{i,r} + \beta_i\ EmpDem_{i,r}}{\alpha_i + \beta_i}, \qquad \forall i \in \mathcal{I},\ \forall r \in \mathcal{R}$$

donde $\alpha_i \in [0,1]$ representa la confianza del departamento de logística en la tienda i en tomar buenas decisiones y $\beta_i \in [0,1]$ la precisión del almacén en estimar las ventas de la tienda i. La demanda variable no es obligatorio satisfacerla (será una restricción "suave" en el problema de optimización).

- $\boldsymbol{\gamma_i},\ i \in I$: Prioridad de la tienda. $\gamma_i \in [0,1]$. Dadas dos tiendas i y j tales que $\gamma_i > \gamma_j$ y que demanden de manera variable un mismo producto y sólo haya disponible uno en la red, la tienda i tendrá preferencia ante la j para recibirlo.

- $\boldsymbol{Cap_p},\ p \in \mathcal{P}:$ Capacidad del paquete p. Tiene que estar en la misma unidad que $Peso_r$ (en principio se pensará en kilogramos, aunque en general podría ser volumen o cualquier otra medida de capacidad).

- $\boldsymbol{Coste_{i,j,p}},\ i,j \in \mathcal{I},\ r \in \mathcal{R},\ j \neq i:$ Coste de enviar el paquete p de la tienda i a la tienda j.

2.2.3. Variables

Teniendo en cuenta que el objetivo del problema es redistribuir el inventario de una red, por una parte cumpliendo con la demanda fija y por otra minimizando el coste de envío, se definen las variables X e Y como sigue:

- $\boldsymbol{X_{i,j,r}},\ i,j \in \mathcal{I},\ r \in \mathcal{R},\ j \neq i:$ Número de unidades del producto r que se envían de la tienda i a la tienda j. Serán variables *enteras* pues los productos son indivisibles.

- $\boldsymbol{Y_{i,j,p}},\ i,j \in \mathcal{I},\ p \in \mathcal{P},\ j \neq i:$ Número de paquetes de tipo p se envían de la tienda i a la tienda j. Serán variables *enteras* pues los paquetes son indivisibles.

2.3. Modelo matemático

2.3.1. Planteamiento inicial

Una vez que ya se han descrito los ingredientes del problema, el modelo que nos gustaría resolver es el siguiente:

$$\text{mín} \quad \sum_{p}\sum_{i}\sum_{j} Coste_{i,j,p}\ \boldsymbol{Y_{i,j,p}} \tag{2.1}$$

$$+ M_1 \sum_{i}\sum_{r} \gamma_i \Big[FixDem_{i,r} + VarDem_{i,r} - Stock_{i,r} - \sum_{j} (\boldsymbol{X_{j,i,r}} - \boldsymbol{X_{i,j,r}}) \Big] \tag{2.2}$$

$$+ M_2 \sum_{i}\sum_{j}\sum_{r} \boldsymbol{X_{i,j,r}} \tag{2.3}$$

s.a.

$$\forall i,\ \forall j, \qquad \sum_{r} Peso_r\ \boldsymbol{X_{i,j,r}}\ \leq\ \sum_{p} Cap_p\ \boldsymbol{Y_{i,j,p}} \tag{2.4}$$

$$\forall i,\ \forall r, \qquad \sum_{i} (\boldsymbol{X_{j,i,r}} - \boldsymbol{X_{i,j,r}}) \geq FixDem_{i,r} - Stock_{i,r} \tag{2.5}$$

$$\forall i,\ \forall r, \qquad \sum_{i} (\boldsymbol{X_{j,i,r}} - \boldsymbol{X_{i,j,r}}) \leq FixDem_{i,r} + VarDem_{i,r} - Stock_{i,r} \tag{2.6}$$

$$\forall i,\ \forall r, \qquad \sum_{j} \boldsymbol{X_{i,j,r}} \leq Stock_{i,r} \tag{2.7}$$

$$\forall i,\ \forall j,\ \forall p, \quad \boldsymbol{Y_{i,j,p}} \in \mathbb{Z}_+ \tag{2.8}$$

$$\forall i,\ \forall j,\ \forall r, \quad \boldsymbol{X_{i,j,r}} \in \mathbb{Z}_+ \tag{2.9}$$

En la **función objetivo** podemos observar tres términos:

1. El término 2.1 representa el coste real de envío de todos los paquetes.
2. El término 2.2 representa la penalización por no satisfacer la demanda variable, donde M_1 representará cómo de agresivo se quiere ser a la hora de satisfacer demanda variable.
3. El término 2.3 representa la penalización por hacer movimientos innecesarios, donde M_2 típicamente tomará un valor muy pequeño, cuya función es desempatar entre soluciones que, teniendo la misma función objetivo, realizan una cantidad distinta de movimientos de productos entre tiendas.

Las **restricciones** sobre el conjunto factible son las siguientes:

1. La ecuación 2.4 asegura que la capacidad total de los paquetes que se envían desde la tienda i a la tienda j es suficiente para transportar el peso de todos los productos que se envían de i a j.

2. La ecuación 2.5 asegura que una vez hecha la redistribución, la tienda i tendrá al menos la demanda fija del producto r.

3. La ecuación 2.6 asegura que una vez hecha la redistribución, la tienda i tendrá como máximo la suma de su demanda fija más su demanda variable del producto r.

4. La ecuación 2.7 asegura que la tienda i no enviará más productos r de los que tiene disponibles.

A la vista la estructura del problema, la idea inicial fue aproximarlo al conocido **problema del transporte**, en el que una mercancía sale de unas fuentes $i \in I$ y tiene unos destinos $j \in J$. Añadiendo una única fuente y un único destino, el problema puede transformarse en un problema de flujo con coste mínimo, cuyo objetivo es determinar la cantidad de flujo que va por cada una de las aristas teniendo en cuenta las restricciones de capacidad de cada una de ellas.

Una particularidad de este tipo de problemas es que, debido a la **unimodularidad** de la matriz de coeficientes, existirán soluciones enteras siempre y cuando las capacidades sean enteras. Y por tanto, se podrá resolver el problema con algoritmos polinómicos sin necesidad de acudir a técnicas de programación entera. Este hecho es de gran ayuda para una eficiente resolución del problema, por lo que lo ideal sería alejarse lo menos posible de él. Para más detalles de los problemas de flujo con coste mínimo consultar [9].

Sin embargo, una restricción fundamental de nuestro caso, la restrición 2.4, impide que la propiedad de unimodularidad se cumpla:

$$\forall i, j \in I \qquad \sum_{r} Peso_r \; \boldsymbol{X_{i,j,r}} \leq \sum_{p} Cap_p \; \boldsymbol{Y_{i,j,p}}.$$

Tal y como se ha comentado, que la capacidad total (la suma de las capacidades) de los paquetes que se envían desde la tienda i a la tienda j, tiene que ser suficiente para transportar todos los elementos que se envían desde i a j. Esta restricción hace que se pierda la unimodularidad, que exige que los coeficientes de las restricciones sean 0, 1 o -1, y en este caso tenemos los pesos de los productos y las capacidades de los paquetes, que no tienen por

qué tomar dichos valores.

Al tener que abordar por tanto el problema con técnicas de programación entera, la siguiente complicación que se encuentra es que la cantidad de variables enteras que se tiene es computacionalmente intratable por técnicas exactas incluso para problemas con un número relativamente pequeño de tiendas y productos. Por tanto, se propone un problema programación lineal entero mixto en el que las variables $\boldsymbol{X_{i,j,r}}$ se relajan (se vuelven continuas $\boldsymbol{X^{Rel}_{i,j,r}}$) y se mantienen las $\boldsymbol{Y_{i,j,p}}$ como únicas variables enteras.

Una vez resuelto el problema, para recuperar la integralidad de las $\boldsymbol{X^{Rel}_{i,j,r}}$ se resuelve un problema de redondeo. De hecho, se resuelve el problema de redondeo múltiples veces variando la función de coste y quedándose con la solución obtenida con menor coste para el problema original.

Finalmente, se resuelve un problema de empaquetado para obtener, para cada producto r que va de la tienda i a la j, en qué paquete se envía de entre los que van de i a j. Así, el problema queda dividido en tres subproblemas que se resolverán consecutivamente:

1. Subproblema de flujo en redes.
2. Subproblema de redondeo de matrices.
3. Subproblema de empaquetado.

2.3.2. Subproblema de flujo en redes

Tal y como se ha comentado en el apartado anterior, debido al alto número de variables enteras, se decide relajar las variables $\boldsymbol{X_{i,j,r}}$. Se decide que sean estas y no las $\boldsymbol{Y_{i,j,p}}$, en primer lugar porque las $\boldsymbol{X_{i,j,r}}$ son mayores en número, y cuanto más variables enteras haya en el problema más tiempo le llevará al solver el problema (ya que crece exponencialmente con el número de variables). Y en segundo lugar porque las $\boldsymbol{Y_{i,j,p}}$ son las que entran más directamente en la función objetivo, y la aproximación del problema poniendo las $\boldsymbol{X_{i,j,r}}$ como continuas es mucho mejor que la aproximación poniendo las $\boldsymbol{Y_{i,j,p}}$ como continuas.

Por la solución de este primer subproblema de optimización permitirá mover entre distintos pares de tiendas cantidades no enteras de productos. En el siguiente subproblema se redondearán estas soluciones, algunas componentes a la alta y otras a la baja. En alguna de

las transacciones, es posible que tras la fase de redondeo la capacidad total de los paquetes que se envía no sea suficiente para poder transportar los productos ahora ya enteros, por lo que habría que añadir paquetes extra en esa transacción.

Como ayuda a la fase de redondeo, se propone introducir sobre el problema de flujo en redes un parámetro de llenado $\delta \in [0, 1]$ que indicará hasta qué porcentaje podrán llenarse los paquetes que se envían en cada transacción. En el caso de que $\delta = 0.85$, por ejemplo, los paquetes podrán llenarse como máximo al 85 %. Así, es posible que el problema de añadir paquetes extra en la fase de redondeo se vea solventado. Se analizará el comportamiento de δ en los resultados numéricos.

Por tanto, el modelo resultante para la fase del problema de flujo en redes a la que denominaremos **"Transferring"** será el mismo que se ha definido en las ecuaciones 2.1-2.9 exceptuando que la restrición 2.4 incorporará el nuevo parámetro δ:

$$\forall i,\ \forall j, \qquad \sum_r Peso_r\ \boldsymbol{X^{Rel}_{i,j,r}}\ \leq\ \sum_p\ \boldsymbol{\delta}\ Cap_p\ \boldsymbol{Y_{i,j,p}}, \tag{2.4bis}$$

y que la $\boldsymbol{X_{i,j,r}}$ será continua (cambiará la restricción 2.9):

$$\forall i,\ \forall j,\ \forall r, \quad \boldsymbol{X^{Rel}_{i,j,r}} \in [0, \infty). \tag{2.9bis}$$

Será precisamente la ecuación 2.4bis la que haga que la unimodularidad no se cumpla.

Cabe destacar que a pesar de que no se esté resolviendo el problema real pues las variables $\boldsymbol{X^{Rel}_{i,j,r}}$ no son enteras, la resolución del subproblema Transferring para $\delta = 1$ nos proporciona una **cota inferior** del óptimo al que se aspira. Y por tanto, en las siguientes fases se podrá medir la distancia con respecto a esa cota y tener una estimación de cuál sería el margen máximo de mejora con respecto a la solución obtenida.

2.3.3. Subproblema de redondeo de matrices

Recuperar la integralidad de las variables $\boldsymbol{X^{Rel}_{i,j,r}}$ es de vital importancia para el caso particular que se está resolviendo. En el caso real que ha motivado este trabajo, el número de unidades que se mueven de cada producto es bastante pequeño, con lo que las cantidades a redondear pueden ser de la forma 0.6 o 0.4. Entonces, el resultado de hacer el redondeo de una forma eficiente puede determinar, por ejemplo, que los envíos entre un par de tiendas pasen todos a valer cero, con el consiguiente ahorro del envío de un paquete entre esas tiendas.

Se resuelve por tanto un problema de redondeo de matrices, en el que para una matriz $\boldsymbol{X^{Rel}} \in \mathbb{R}^{i \times j}$ se calcula la matriz entera $\boldsymbol{X} \in \mathbb{Z}^{i \times j}$ cuyos elementos están redondeados, y a además la suma de cada una de sus filas y cada una de sus columnas también está redondeada con respecto a las de la matriz $\boldsymbol{X^{Rel}}$. En la Tabla 2.1 se puede ver un ejemplo del redondeo de una matriz.

			Suma de filas
	3.1	4.8	7.9
	1.8	3.3	5.1
Suma de columnas	4.9	8.1	13
			Suma **redondeada** de filas
	3	5	8
	2	3	5
Suma **redondeada** de columnas	5	8	13

Tabla 2.1: Ejemplo de redondeo de matrices

La ventaja de plantear el redondeo de esta manera, es que el subproblema se puede escribir como un **problema de flujo en redes** que respeta la propiedad la **unimodularidad**, y que por tanto, a pesar de tener un alto número de variables enteras, se pueden obtener soluciones enteras sin necesidad de acudir a técnicas de programación entera.

Así, el modelo resultante para la fase de redondeo de matrices a la que denominaremos **"Rounding"** es el que sigue:

$$\text{mín} \quad \sum_i \sum_j \sum_r \hat{c}_{i,j,r} \; \boldsymbol{X_{i,j,r}} \tag{2.10}$$

s.a.

$$\forall i, \; \forall j, \; \forall r, \qquad \left\lfloor X^{Rel}_{i,j,r} \right\rfloor \leq \; \boldsymbol{X_{i,j,r}} \leq \left\lfloor X^{Rel}_{i,j,r} \right\rfloor + 1 \tag{2.11}$$

$$\forall j, \; \forall r, \qquad \left\lfloor \sum_i X^{Rel}_{i,j,r} \right\rfloor \leq \; \sum_i \boldsymbol{X_{i,j,r}} \leq \left\lfloor \sum_i X^{Rel}_{i,j,r} \right\rfloor + 1 \tag{2.12}$$

$$\forall i, \; \forall r, \qquad \left\lfloor \sum_j X^{Rel}_{i,j,r} \right\rfloor \leq \; \sum_j \boldsymbol{X_{i,j,r}} \leq \left\lfloor \sum_j X^{Rel}_{i,j,r} \right\rfloor + 1 \tag{2.13}$$

$$\forall i, \; \forall j, \; \forall r, \quad \boldsymbol{X_{i,j,r}} \in \mathbb{Z}_+ \tag{2.14}$$

donde $X^{Rel}_{i,j,r}$ es el resultado de la fase anterior.

Puesto que esta fase es la más importante del algoritmo y es sobre la que más cambios se ha hecho en el nuevo modelado, se explicará con más detalle en el Capítulo 3. Se pueden encontrar más detalles en [6].

2.3.4. Subproblema de empaquetado

Tras resolver los dos subproblemas anteriores, ya está determinada la cantidad de productos p que se envían entre cada par de tiendas i y j ($\boldsymbol{X_{i,j,r}}$) y el número de paquetes de cada tipo necesarios para hacer la transacción ($\boldsymbol{Y_{i,j,p}}$). Sin embargo, aún queda por determinar la configuración de cada uno de los paquetes que se envía; es decir, hay que indicar qué producto va en qué paquete.

La necesidad de resolver también este problema proviene del hecho de que la restricción de capacidad suficiente que se verifica en las etapas anteriores sólo tiene en cuenta la capacidad total de todos los paquetes que se envían:

$$\sum_{r} Peso_r \ \boldsymbol{X_{i,j,r}} \leq \sum_{p} \ Cap_p \ \boldsymbol{Y_{i,j,p}}. \tag{2.15}$$

Esto puede suponer un problema para alguna de las soluciones que se obtengan, pues debido a la integralidad de los productos, es posible que el empaquetado no sea factible: supóngase que se tuviera una solución en la que se enviaran 3 productos cada uno con peso 3kg y se tuvieran paquetes con capacidad de 5kg. De las dos fases anteriores se concluiría que sólo haría falta enviar 2 paquetes pues la restricción 2.15 se verificaría. Sin embargo, es claro que no se puede hacer el empaquetado únicamente con 2 paquetes pues los productos son indivisibles. En esos casos habría que añadir paquetes extra a la transacción.

Así, por cada par de tiendas habrá que resolver dos problemas de empaquetado y, por tanto, en el caso de tener I tiendas se reloverán $I^2 - I$ subproblemas. Este problema es el denominado **Problema de Empaquetamiento** o más comunmente conocido como el **Bin Packing Problem**, [10], cuya formulación matemática es la que se muestra en las Ecuaciones 2.16 - 2.20, donde $\boldsymbol{y_p}$ tomará el valor 1 si se utiliza el paquete p y $\boldsymbol{x_{p,r}}$ tomará el valor 1 si el producto r va en el paquete p.

$$\text{mín} \quad B = \sum_{p=1}^{P} \boldsymbol{y_p} \tag{2.16}$$

s.a.

$$\forall p \in P, \qquad \sum_{r=1}^{R} Peso_r \; \boldsymbol{x_{p,r}} \; \leq \; Cap_p \; \boldsymbol{y_p} \tag{2.17}$$

$$\forall r \in R, \qquad \sum_{p=1}^{P} \boldsymbol{x_{p,r}} = 1 \tag{2.18}$$

$$\forall p \in P, \qquad \boldsymbol{y_p} \in \{0,1\} \tag{2.19}$$

$$\forall p \in P, \forall r \in R, \quad \boldsymbol{x_{p,r}} \in \{0,1\} \tag{2.20}$$

En este caso se decidió resolver el problema con una heurística aproximada en lugar de resolver el problema de empaquetado puro. Esta heurística coloca cada producto escogido arbitrariamente en el primer paquete que tenga espacio disponible. En caso de que no hubiera espacio suficiente en ninguno de los paquetes, se añadiría un nuevo paquete. En el peor de los casos se añadirían menos del doble del óptimo número de paquetes.

En el siguiente capítulo veremos que, aunque el bin packing problem es un problema NP-Completo, el hecho de que se pueda resolver independientemente para cada par de tiendas hace tengamos problemas de tamaño relativamente pequeño y que se puedan resolver de forma exacta sin un gran coste computacional.

2.4. Programación y resultados

Para la programación de este algoritmo se utilizó el software R, con el que se creó un paquete denominado **"transfer"** que resuelve cada una de las fases mencionadas individualmente y de manera consecutiva, y que además representa los resultados gráficamente. Para la resolución de los problemas de optimización se hizo uso de los solvers de programación lineal y entera *Gurobi* y *lpSolve*, y se generaron dos bloques de 25 simulaciones para la validación del algoritmo.

Para comprobar la efectividad del parámetro δ, se realizaron varias ejecuciones del algoritmo alterando los valores de dicho parámetro. En particular para $\delta = \{0.8, 0.85, 0.9, 0.95, 1\}$.

Se comprobó, tal y como se esperaba, que a medida que δ aumentaba el número de paquetes era menor. También se realizaron distintas pruebas alterando los valores de los parámetros de penalización M_1 y M_2 de la fase Transferring.

Una buena configuración de las opciones del solver fue fundamental, pues los parámetros por defecto agotaban la memoria. Entre otros, hubo que ser muy agresivo con el pre-solve del primer subproblema y hubo que limitar el tiempo de búsqueda cuando el modelo era excesivamente grande.

Por otra parte, también se ejecutó el algoritmo para un caso con datos reales. Se comprobó que en comparación con el trabajo manual que entonces se hacía en la empresa, el coste de envíos se reducía en un alto porcentaje. Esto se produce en parte porque, tal y como se ha comentado anteriormente, la empresa no permite que las tiendas se envíen paquetes entre sí, sino que tienen que enviarlo todo al almacén para ser de nuevo redistribuido. Esto hace que los envíos se dupliquen pero le permite a la empresa tener un mayor control sobre el inventario.

En total, como los resultados obtenidos fueron muy prometedores y el problema tenía margen de mejora, se decidió continuar con su validación, pues no había sido todo lo exhausta que se desearía, y a la introducción nuevos elementos que contibuyeran a su mejora.

Capítulo 3

Profundizando en el modelo

A la vista de los resultados obtenidos por Pablo Montero con el modelo propuesto y la potencialidad del propio problema, se decide continuar con su validación y mejora. Para ello, en primer lugar se reprograma completamente el todo el proceso de decisión utilizando el lenguaje de programación AMPL que está especialmente enfocado a la optimización.

Una vez hecha la reprogramación del algoritmo, se procedió a su validación. Tras observar los resultados, se decidió realizar algunos cambios sobre determinadas partes y añadir contenido al proceso. A continuación se mencionan algunos de los mayores cambios realizados sobre los modelos de partida, que se describirán más en detalle en los siguientes apartados.

- Incorporación de una fase relajada del Transferring.
- Reajuste del término de penalización de la demanda variable en la función objetivo de la fase de Transferring.
- Reajuste de la fase de redondeo de matrices.
- Asegurar la satisfacción de la demanda fija con la incorporación de una heurística tras el redondeo.
- Resolución exacta de la fase de empaquetado.

3.1. Fase 0: Transferring relajado

Una de las primeras observaciones que se pudieron hacer tras la reprogramación del problema, es que el cuello de botella del algoritmo se encuentra en la fase de Transferring. En ejemplos relativamente pequeños, prácticamente la totalidad del tiempo de ejecución se invertía en encontrar la solución óptima de la primera fase, y esto ponía en duda la posibilidad de encontrar una buena solución para un problema de mayor tamaño en un tiempo razonable.

Idealmente, lo que se desearía es poder lanzar la versión totalmente relajada del problema Transferring (con las variables $\boldsymbol{Y_{i,j,p}}$ continuas), resolver el problema de flujo con coste mínimo (PFCM) para el redondeo de paquetes y después resolver el PFCM para el redondeo de productos, y finalmente resolver el problema de empaquetado. Por desgracia, el redondeo de paquetes debería tener en cuenta los productos, y esto no es posible hacerlo sin romper la unimodularidad.

Se propone por tanto una solución intermedia, que trata de añadirle al algoritmo una fase inicial que dependerá de un parámetro de ancho de ventana al que se denotará por v:

Etapa 0: Transferring relajado. Se resuelve el problema del Transferring relajando tanto los artículos como los paquetes para que sean variables continuas. Este proceso debería ser muy rápido y nos dará una primera cota inferior para la mejor solución que se puede obtener que es independiente del valor de v. Evidentemente, no se aspira a estar cerca de esa solución, pues la posibilidad de enviar fracciones de paquetes reduciría mucho el coste. El valor del parámetro de ancho de ventana v, que vendrá dado como dato, implicará que el proceso se ejecute de una manera u otra:

- Si $v < 0$, se pasará a la ETAPA 1 (Transferring).
- Si $v \geq 0$, entonces se cambiarán las cotas del problema original como sigue: Dadas un par de tiendas i y j, y $\boldsymbol{Y^{Rel}_{i,j,p}} = Valor$ el número de paquetes que envía i a j en el problema **relajado**, entonces,

$$\text{máx}(0\ ,\ \lfloor Valor \rfloor - v)\ \leq\ \boldsymbol{Y_{i,j,p}}\ \leq\ \lfloor Valor \rfloor + 1 + v, \tag{3.1}$$

  siendo $\boldsymbol{Y_{i,j,p}}$ el número de paquetes que envía i a j tras el Transferring.

  En primer lugar, en la parte izquierda de la Ecuación 3.1 se pone el máximo entre 0 y la parte entera del valor obtenido en la fase 0 menos v, pues es posible que v

supere el número de paquetes enviados y la variable $\boldsymbol{Y_{i,j,p}}$ tiene que ser mayor o igual que 0. Por otra parte, en la parte derecha de la ecuación se pone $\lfloor Valor \rfloor + 1$ en lugar de $\lceil Valor \rceil$ pues en caso de que $Valor$ fuese entero, la parte entera y el techo coincidirían.

Se pasa a la ETAPA 1.

Etapa 1: Transferring. Se resuelve el Transferring como hasta ahora, salvo por las cotas hayan podido ser modificadas en la ETAPA 0.

Etapa 2: Rounding. Como hasta ahora.

Etapa 3: Packing. Como hasta ahora.

Así, para el caso $v = 0$, la primera etapa buscará entre las soluciones enteras en las que el número de paquetes se corresponda con algún redondeo del número de paquetes de la solución del problema relajado. Sería prácticamente como utilizar variables binarias, lo cual suele ser bastante más rápido que trabajar con variables enteras. Cabe mencionar que es posible que la solución obtenida sea peor que la que se hubiera obtenido resolviendo el Transferring sin haber cambiado las cotas.

A medida v aumenta, se le da más margen con las cotas y las variables enteras podrán tomar más valores. Por tanto, cada vez se estará más cerca del problema original y se tardará más en resolverlo. La clave está en ver dónde está el equilibrio entre la calidad de la solución obtenida y el tiempo de resolución, pues si realmente hubiera diferencia en los tiempos de ejecución y las diferencias en los costes obtenidos fueran razonables, v puede ser un parámetro de interés para el usuario final.

3.2. Función objetivo del Transferring

Recordamos que la función objetivo de la fase de Transferring es la siguiente:

$$\text{mín} \quad \sum_p \sum_i \sum_j Coste_{i,j,p}\, \boldsymbol{Y_{i,j,p}} \tag{2.1}$$

$$+ M_1 \sum_i \sum_r \gamma_i \Big[FixDem_{i,r} + VarDem_{i,r} - Stock_{i,r} - \sum_j \left(\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}} \right) \Big] \tag{2.2}$$

$$+ M_2 \sum_i \sum_j \sum_r \boldsymbol{X^{Rel}_{i,j,r}}, \tag{2.3}$$

donde, tal y como se ha comentado, el primer término se corresponde con el coste de envío de paquetes, el segundo con la penalización por inclumplimiento de demanda variable, y el tercero con la penalización por movimientos innecesarios.

Tanto la ecuación 2.1 como 2.3 son sencillas de ver y entender, y cabe esperar que en cualquier modelo que se propusiera para resolver este problema deberían estar presentes. Sin embargo, el segundo término es algo más complicado, por lo que se hará un estudio particular de él.

Denotemos por $FixVarStock_{i,r} = FixDem_{i,r} + VarDem_{i,r} - Stock_{i,r},\ \forall i \in \mathcal{I},\ \forall r \in \mathcal{R}$. Así, término de la función objetivo 2.2 se puede reformular como sigue:

$$M_1 \sum_i \sum_r \gamma_i \Big[FixVarStock_{i,r} - \sum_j \Big(\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}} \Big) \Big], \tag{3.2}$$

donde el término $\sum_j \Big(\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}} \Big)$ es el balance entre las unidades que recibe y envía la tienda i del producto r. Tomará un valor positivo cuando reciba más de lo que envía y negativo en el caso contrario.

- Cuando $FixVarStock_{i,r} \geq 0$ se tiene que no hay stock suficiente para satisfacer la demanda fija mas la variable del producto r en la tienda i. La restricción 2.5 garantiza que la demanda fija se va a satisfacer, y el término de penalización M_1 va a provocar que se intente enviar a la tienda i productos hasta llegar a satisfacer la demanda variable por completo (porque así el término se anularía), pero no más porque lo prohíbe la restricción 2.6.

- Cuando $FixVarStock_{i,r} \leq 0$, se tiene que sí hay stock suficiente para satisfacer la demanda fija mas la variable del producto r en la tienda i. El término de penalización M_1 va a intentar que el término se anule y por tanto se incentivará el envío de productos desde esta tienda (al contrario de lo que sucedía en el caso anterior).

El problema que se encontró fue que para $FixVarStock \leq 0$, el término M_1 no sólo provocaba que el término se anulara, sino que fuera negativo. Es decir, se dejaba de cubrir la demanda variable de la tienda para se enviaran tantos productos de más hasta no violar la restricción de satisfacción de demanda fija 2.5.

Se decició por tanto, que el término que debería ir en lugar de el 2.2 es

$$M_1 \sum_i \sum_r \gamma_i \ \text{máx}\left(0,\ FixVarStock_{i,r} - \sum_j \left(\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}}\right)\right), \tag{3.3}$$

puesto que si una tienda tiene stock suficiente para cubrir su demanda fija y variable, a lo sumo debería enviar su diferencia $(Stock - (FixDem + VarDem))$ para seguir satisfaciendo ambas demandas.

3.3. Redondeo de matrices

3.3.1. Planteamiento

Tal y como se ha comentado en el segundo capítulo, el subproblema de redondeo de matrices o Rounding, se puede modelar como un problema de flujo con coste mínimo cuyo grafo asociado:

1. Tendrá un nodo fuente y un nodo destino añadidos. Se corresponderían con los nodos F y D de la Figura 3.1.

2. Tendrá dos nodos intermedios por cada tienda, pues uno de ellos simbolizaría la tienda como origen (nodos I1, I2 e I3) y el otro como destino (nodos J1, J2 y J3).

3. El nodo fuente estará conectado con todos los nodos origen. Dichas aristas simbolizarán el número total de productos que cada tienda envía en total. Esta cantidad tiene que ser entera tras el redondeo. En la Figura 3.1 están representadas con el número 1.

4. Cada par de nodos centrales se conectarán a través de tantas aristas como tipos de productos se envíen. Evidentemente, entre el nodo origen que simboliza la tienda i y el nodo destino que simbolice la misma tienda i no debe haber ninguna arista que los conecte. De nuevo, cada una de estas cantidades tiene que ser entera. En la Figura 3.1 serán las aristas en negro y rojo que están representadas con el número 2.

5. Los nodos centrales destino se conectarán con el destino final ficticio a través de aristas que simbolizarán el número total de productos que ha recibido cada una de las tiendas. Esta cantidad también deberá ser entera. Están representadas con el número 4 en la Figura 3.1.

6. El nodo fuente y el nodo destino ficticios también están conectados representando que todo lo que se envía debe ser recibido. Esta cantidad debe ser entera y se puede ver representada con el 5 en la Figura 3.1.

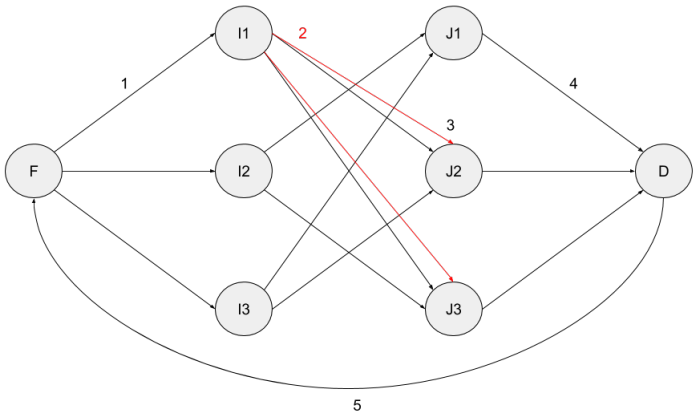


Figura 3.1: Grafo asociado al problema de redondeo de matrices

Por tanto, denotando por $X^{Rel}_{i,j,r}$ al número de productos p que se envían en la tienda i a la j tras resolver el Transferring (que será ahora un parámetro), y por $\boldsymbol{X_{i,j,r}}$ a dicho valor redondeado, la formulación del subproblema de redondeo como problema de flujo con coste mínimo se puede ver descrito en las Ecuaciones 3.4 - 3.10.

$$\text{mín} \quad \sum_i \sum_j \sum_r \hat{c}_{i,j,r} \; \boldsymbol{X_{i,j,r}} \tag{3.4}$$

s.a.

$$\forall i,\ \forall j,\ \forall r, \qquad \left\lfloor X^{Rel}_{i,j,r} \right\rfloor \leq \boldsymbol{X_{i,j,r}} \leq \left\lceil X^{Rel}_{i,j,r} \right\rceil \tag{3.5}$$

$$\forall i,\ \forall r, \qquad \left\lfloor \sum_j X^{Rel}_{i,j,r} \right\rfloor \leq \sum_j \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_j X^{Rel}_{i,j,r} \right\rceil \tag{3.6}$$

$$\forall j,\ \forall r, \qquad \left\lfloor \sum_i X^{Rel}_{i,j,r} \right\rfloor \leq \sum_i \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_i X^{Rel}_{i,j,r} \right\rceil \tag{3.7}$$

$$\forall r, \qquad \left\lfloor \sum_i \sum_j X^{Rel}_{i,j,r} \right\rfloor \leq \sum_i \sum_j \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_i \sum_j X^{Rel}_{i,j,r} \right\rceil \tag{3.8}$$

$$\left\lfloor \sum_i \sum_j \sum_r X^{Rel}_{i,j,r} \right\rfloor \leq \sum_i \sum_j \sum_r \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_i \sum_j \sum_r X^{Rel}_{i,j,r} \right\rceil \tag{3.9}$$

$$\forall i,\ \forall j,\ \forall r, \qquad \boldsymbol{X_{i,j,r}} \in \mathbb{Z}_+ \tag{3.10}$$

- La Ecuación 3.5 dice que el número de unidades enviadas de cada producto entre cada par de tiendas tiene que ser un redondeo del número obtenido en el Transferring.
- La Ecuación 3.6 dice que el número de productos de tipo r enviados desde cada tienda i debe ser un redondeo del número obtenido en el Transferring.
- La Ecuación 3.7 dice que el número de productos de tipo r enviados a cada tienda j debe ser un redondeo del número obtenido en el Transferring.
- La Ecuación 3.8 dice que el número total de unidades de cada producto r que se mueven por la red debe ser un redondeo del número obtenido en el Transferring.
- La Ecuación 3.9 dice que el número total de unidades que se mueven por la red debe ser el redondeo del número obtenido en el Transferring.
- La Ecuación 3.10 dice que el número total de productos que se mueven por la red debe ser un redondeo del número obtenido en el Transferring.

Tanto en la función objetivo 3.4 como en la 2.10 se pretende minimizar el número productos X enviados multiplicados por un coste $\hat{c}_{i,j,r}$. Este coste determina cómo debe realizarse el redondeo. En el trabajo anterior se hicieron varias pruebas con distintas definiciones de este coste, pero finalmente la que mejores resulto fue la que sigue:

$$\hat{c}_{i,j,r} \sim U(0, P_{i,j} - W_{i,j}), \tag{3.11}$$

donde U representa una distribución uniforme y $P_{i,j} - W_{i,j}$ representa el espacio total disponible entre los paquetes que se envían de i a j. Por tanto ahora el subproblema se trataría de maximizar 3.4, pues se resuelve el Rounding tomando el mayor de los $X_{i,j,r}$ para aquellos pares de tiendas que tengan más espacio libre, de manera que la capacidad utilizada para empaquetar los productos quede lejos del límite.

La ventaja de plantear el subproblema de esta manera es que se cumple la unimodularidad, y por tanto, a pesar de ser un problema con variables enteras, se puede conseguir una solución óptima factible (entera) del problema sin necesidad acudir a ténicas de programación entera, y por tanto el tiempo de ejecución será mucho menor.

3.3.2. Implementación

Como ya se ha dicho, es posible que la solución del Rounding no cumpla la restricción de factibilidad 2.4, puesto que puede suceder que el espacio que se había reservado para trans-

portar los productos en una determinada transacción no sea suficiente tras el redondeo. En ese caso habría que añadir paquetes extra.

Por ello, se resuelve el subproblema de redondeo iterativamente un número determinado (parámetro *nIters*) de veces (en cada una de estas ejecuciones los costes $\hat{c}_{i,j,r}$ se habrán generado de manera aleatoria), y para cada una de estas soluciones se calcula el el número de paquetes extra añadidos. La mejor solución será aquella que menos paquetes extra añada. El proceso se detendrá en cuanto se encuentre una solución en la que no se añada ningún paquete extra, o cuando el número de paquetes extra que se añadan sea inferior a un determinado porcentaje ($minPerc$) del número total de paquetes que se envían.

Sin embargo, se comprobó empíricamente que no sólo fallaría la factibilidad en la restricción 2.4, sino que la restricción 2.5 también fallaría (en un pequeño porcentaje de las veces). Es decir, la demanda fija no se satisfaría para alguna de las tiendas de un determinado producto. Esta posibilidad no se había contemplado hasta el momento, por lo que se decide hacer un postprocesado de la solución para recuperar la factibilidad.

Además, se comprobó que en muchos de los casos en los que fallaba la factibilidad era porque en la fase del Transferring, una tienda que demandaba de manera fija un producto y lo tenía en stock, enviaba una fracción de él a otra tienda y recibía esa misma fracción desde otra tienda. Tras el redondeo, las tiendas emisoras y receptoras salían ganando y la tienda en cuestión se quedaba sin su producto.

Por tanto, antes de comenzar con el posprocesado, se introdujo una nueva restricción en el subproblema de Transferring, que dice que en caso de que se tenga en stock un artículo que se demanda de manera fija, ese producto no debe ser enviado:

$$\forall i,\ \forall r:\ Stock_{i,r} - FixDem_{i,r} \geq 0, \qquad \sum_{j} \boldsymbol{X}^{\boldsymbol{Rel}}_{\boldsymbol{i,j,r}} \ \leq\ Stock_{i,r} - FixDem_{i,r}. \tag{3.12}$$

Con la introducción de esta nueva restricción se redujeron el número de veces en el que se violaba la satisfacción de la demanda fija, pero no se arregló el problema por completo. Por tanto, se decidió hacer un posprocesado.

Para ello, supóngase que la demanda fija que no ha sido satisfecha es la $FixDem_{i,r}$. Cabe mencionar que esta demanda fija sólo habrá sido violada como máximo en una unidad a causa del redondeo. En primer lugar, se mirará de entre todos los paquetes que recibe la tienda i

si hay el alguno de ellos espacio suficiente para el artículo r, y si lo hay, se comprobaría si la tienda desde la que se envía el paquete tiene dicho artículo disponible. En caso de que no hubiera espacio suficiente en ninguno de los paquetes, o en los paquetes que lo hubiera la tienda emisora no tuviera el producto disponible, se lanzaría la Heurística 2b que se describirá en el Capítulo 5, que a grandes rasgos, envíará un nuevo paquete desde la tienda que tenga el artículo disponible y que menos coste de envío tenga.

En el Capítulo 4 se puede ver un pseudocódigo de la implementación del Rounding.

3.3.3. Comparación con el modelo antiguo

La tarea de validación del problema de redondeo de matrices ha sido sin duda la más complicada del estudio, pues la detección de limitaciones en su planteamiento ha provenido de sutilezas en los resultados.

En primer lugar se programó el subproblema tal y como se ha propuesto en el Capítulo 2 y se procedió a su ejecución. En unas tablas similares a las que se mostrarán en el Capítulo 7, se detectó que para un mismo problema, el número de paquetes extra que se añadían tras el redondeo era superior para $\delta = 0.95$ que para $\delta = 1$. Esto era sospechoso porque precisamente el parámetro δ se propone para que el número de paquetes extra tras la fase de redondeo se vea reducido.

El motivo de que esto sucediera fue que el número de productos que se envíaban tras el Rounding era mucho mayor que los que enviaban tras el Transferring (un número real), cuando se debería corresponder con un redondeo de dicho número. Esta situación provocó la necesidad de introducir la restricción 3.9. Al fin y al cabo, las restricciones 3.9 y 3.8 lo que hacen es tratar el problema de redondeo como un problema de redondeos anidados, por lo que el problema resultante no deja de ser un problema de flujo con coste mínimo, aunque con un grafo más complejo que el representado en la Figura 3.1.

3.4. Resolución exacta del problema de empaquetado

Recordamos que en el momento en el que se desea resolver el problema de empaquetado ya se han resuelto los dos subproblemas anteriores, y que por tanto se conocen el número de unidades a enviar para cada producto r entre cada par de tiendas i y j, $\boldsymbol{X_{i,j,r}}$, y el número de

paquetes de cada tipo $\boldsymbol{Y_{i,j,p}}$. Así, solamente faltaría por determinar la composición de cada uno de los paquetes. Recordamos también que ese empaquetado puede no ser siempre factible debido a la integralidad de los productos, por lo que en algunos casos podría ser necesario añadir paquetes extra para dichas transacciones.

Para el caso del subproblema de empaquetado, en lugar de reproducir lo que se hizo en el trabajo anterior, se decide desde un principio que el problema se debe resolver de manera exacta. Se decide así, pues a pesar de ser una problema con variables binarias (a resolver con técnicas de programación entera), el tamaño total del mismo se ve reducido al resolverse un problema independiente para cada par de tiendas. Por tanto, para cada par i y j se desea resolver exactamente el problema que se propuso en las Ecuaciones 2.16 - 2.20, a excepción del objetivo, en el que se incluirá el coste de enviar cada paquete en dicha transacción:

$$\text{mín} \quad \sum_{p=1}^{P} Coste_{i,j,p} \; \boldsymbol{y_p}$$

La introducción del coste en la función objetivo se debe a que puede haber distintos tipos de paquetes con distintas tarifas de envío.

Así, para cada par de tiendas i y j se procederá de la siguiente manera:

- Se define el conjunto P como el conjunto de paquetes que se han enviado de la tienda i a la tienda j más una cantidad de paquetes determinada por el usuario a la que se denominará $ExtraPacks_{i,j,p}$.

  El parámetro $ExtraPacks_{i,j,p}$ servirá para evitar en gran parte los posibles problemas de factibilidad que podrían surgir en la fase de empaquetado. Esto se debe a que si de las fases anteriores se ha determinado de manera óptima sin tener en cuenta la integralidad de los productos que en número de paquetes a enviar es n, es probable que añadiendo pocos paquetes más el problema sea factible. En particular porque el tamaño de los productos en comparación con la capacidad de los paquetes es muy pequeño.

  En cualquier caso, en el peor de los casos el número máximo de paquetes a enviar, tal y como se ha comentado, es menor que el doble de los paquetes que se han decidido enviar en las fase de redondeo.

  Asi, si por ejemplo, se hubieran enviado 3 paquetes en la fase de redondeo y el parámetro $ExtraPacks = 2$, el conjunto P sería $P = \{p_1, p_2, p_3, p_4, p_5\}$.

- Se define el conjunto R como el conjunto de productos que se envían de i a j. Si por ejemplo, se enviaran 30 productos, el conjunto R sería $R = \{r_1, \ldots, r_{30}\}$.

- Se resuelve el problema de empaquetado 2.16 - 2.20.
- Se comprueba la factibilidad del problema:
  - Si el problema es factible y no se han utilizado para el envío todos los paquetes del conjunto P, es óptimo y el problema se ha resuelto.
  - Si el problema es factible, pero además se ha resuelto utilizando la totalidad de paquetes que se había permitido utilizar, se resuelve de nuevo el problema aumentando el valor del parámetro $ExtraPacks$.
  - Si el problema es infactible, se resuelve de nuevo aumentando el valor del parámetro $ExtraPacks$.

Uno de los efectos que se observa tras la resolución del Packing es que, además de asegurar empaquetados factibles, en algunas ocasiones también se reduce el coste con respecto a la etapa anterior. Este efecto se debe a que en la fase del redondeo, cuando dichos redondeos han conllevado el añadir paquetes extra, estos paquetes se han añadido de forma miope, sin optimizar. Sin embargo, es posible que donde se están enviando dos paquetes pequeños después del redondeo resulte más barato enviar simplemente un paquete grande. Esto se corregirá en la fase de empaquetado, que por tanto puede acabar reduciendo costes con respecto al redondeo.

Capítulo 4

Modelo Final

Después de haber analizado por separado cada uno de los puntos que se mencionaban en la introducción del Capítulo 3, y tras haber comentado las correcciones y aportaciones sobre cada uno de ellos, a modo de resumen, se describirán a continuación los modelos finales a resolver y los procedimientos a seguir en la ejecución de cada una de las fases: Transferring Relajado, Transferring, Rounding y Packing.

4.1. Fase 0. Transferring Relajado

4.1.1. Modelo

$$\text{mín} \quad \sum_p \sum_i \sum_j Coste_{i,j,p}\ \boldsymbol{Y_{i,j,p}^{Rel}} \tag{4.1}$$

$$+ M_1 \sum_i \sum_r \gamma_i\ \text{máx}\left(0,\ FixVarStock_{i,r} - \sum_j \left(\boldsymbol{X_{j,i,r}^{Rel}} - \boldsymbol{X_{i,j,r}^{Rel}}\right)\right) \tag{4.2}$$

$$+ M_2 \sum_i \sum_j \sum_r \boldsymbol{X_{i,j,r}^{Rel}} \tag{4.3}$$

s.a.

$$\forall i,\ \forall j, \qquad \sum_r Peso_r\ \boldsymbol{X_{i,j,r}^{Rel}}\ \leq\ \sum_p\ \delta\ Cap_p\ \boldsymbol{Y_{i,j,p}^{Rel}} \tag{4.4}$$

$$\forall i,\ \forall r, \qquad \sum_i (\boldsymbol{X_{j,i,r}^{Rel}} - \boldsymbol{X_{i,j,r}^{Rel}}) \geq FixDem_{i,r} - Stock_{i,r} \tag{4.5}$$

$$\forall i,\ \forall r, \qquad \sum_i (\boldsymbol{X_{j,i,r}^{Rel}} - \boldsymbol{X_{i,j,r}^{Rel}}) \leq FixVarStock_{i,r} \tag{4.6}$$

$$\forall i,\ \forall r, \qquad \sum_j \boldsymbol{X_{i,j,r}^{Rel}} \leq Stock_{i,r} \tag{4.7}$$

$$\forall i,\ \forall r:\ Stock_{i,r} - FixDem_{i,r} \geq 0, \qquad \sum_j \boldsymbol{X^{Rel}_{i,j,r}} \leq Stock_{i,r} - FixDem_{i,r} \tag{4.8}$$

$$\forall i,\ \forall j,\ \forall p, \quad \boldsymbol{Y^{Rel}_{i,j,p}} \in [0,\infty) \tag{4.9}$$

$$\forall i,\ \forall j,\ \forall r, \quad \boldsymbol{X^{Rel}_{i,j,r}} \in [0,\infty) \tag{4.10}$$

Su implementación se omitirá pues es trivial. Simplemente se ejecuta con el solver lineal correspondiente.

4.2. Fase 1. Transferring

4.2.1. Modelo

$$\text{mín} \quad \sum_p \sum_i \sum_j Coste_{i,j,p}\ \boldsymbol{Y_{i,j,p}} \tag{4.11}$$

$$+ M_1 \sum_i \sum_r \gamma_i\ \text{máx}\left(0,\ FixVarStock_{i,r} - \sum_j \left(\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}}\right)\right) \tag{4.12}$$

$$+ M_2 \sum_i \sum_j \sum_r \boldsymbol{X^{Rel}_{i,j,r}} \tag{4.13}$$

s.a.

$$\forall i,\ \forall j, \qquad \sum_r Peso_r\ \boldsymbol{X^{Rel}_{i,j,r}}\ \leq\ \sum_p\ \delta\ Cap_p\ \boldsymbol{Y_{i,j,p}} \tag{4.14}$$

$$\forall i,\ \forall r, \qquad \sum_i (\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}}) \geq FixDem_{i,r} - Stock_{i,r} \tag{4.15}$$

$$\forall i,\ \forall r, \qquad \sum_i (\boldsymbol{X^{Rel}_{j,i,r}} - \boldsymbol{X^{Rel}_{i,j,r}}) \leq FixVarStock_{i,r} \tag{4.16}$$

$$\forall i,\ \forall r, \qquad \sum_j \boldsymbol{X^{Rel}_{i,j,r}} \leq Stock_{i,r} \tag{4.17}$$

$$\forall i,\ \forall r:\ Stock_{i,r} - FixDem_{i,r} \geq 0, \qquad \sum_j \boldsymbol{X^{Rel}_{i,j,r}} \leq Stock_{i,r} - FixDem_{i,r} \tag{4.18}$$

$$\forall i,\ \forall j,\ \forall p, \quad \boldsymbol{Y_{i,j,p}} \in \mathbb{Z}, \qquad \text{máx}(0, \lfloor Y^{Rel}_{i,j,p} \rfloor - v) \leq\ \boldsymbol{Y_{i,j,p}} \leq\ \lfloor Y^{Rel}_{i,j,p} \rfloor + 1 + v \tag{4.19}$$

$$\forall i,\ \forall j,\ \forall r, \quad \boldsymbol{X^{Rel}_{i,j,r}} \in [0,\infty) \tag{4.20}$$

Nótese que las variables $\boldsymbol{Y_{i,j,p}}$ ahora son enteras y están acotadas por el parámetro ancho de ventana v y la solución del Transferring relajado. De nuevo, se omitirá la implementación pues simplemente se resuelve el modelo con el solver.

4.3. Fase 2. Rounding

4.3.1. Modelo

$$\text{máx} \quad \sum_i \sum_j \sum_r \hat{c}_{i,j,r} \; \boldsymbol{X_{i,j,r}} \tag{4.21}$$

s.a.

$$\forall i,\ \forall j,\ \forall r, \qquad \left\lfloor X^{Rel}_{i,j,r} \right\rfloor \leq \; \boldsymbol{X_{i,j,r}} \leq \left\lceil X^{Rel}_{i,j,r} \right\rceil \tag{4.22}$$

$$\forall j,\ \forall r, \qquad \left\lfloor \sum_i X^{Rel}_{i,j,r} \right\rfloor \leq \; \sum_i \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_i X^{Rel}_{i,j,r} \right\rceil \tag{4.23}$$

$$\forall i,\ \forall r, \qquad \left\lfloor \sum_j X^{Rel}_{i,j,r} \right\rfloor \leq \; \sum_j \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_j X^{Rel}_{i,j,r} \right\rceil \tag{4.24}$$

$$\forall r, \qquad \left\lfloor \sum_i \sum_j X^{Rel}_{i,j,r} \right\rfloor \leq \; \sum_i \sum_j \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_i \sum_j X^{Rel}_{i,j,r} \right\rceil \tag{4.25}$$

$$\left\lfloor \sum_i \sum_j \sum_r X^{Rel}_{i,j,r} \right\rfloor \leq \; \sum_i \sum_j \sum_r \boldsymbol{X_{i,j,r}} \leq \left\lceil \sum_i \sum_j \sum_r X^{Rel}_{i,j,r} \right\rceil \tag{4.26}$$

$$\forall i,\ \forall j,\ \forall r, \qquad \boldsymbol{X_{i,j,r}} \in \mathbb{Z}_+ \tag{4.27}$$

4.3.2. Implementación

La implementación del Rounding puede verse en el Algoritmo 2.

4.4. Fase 3. Packing

4.4.1. Modelo

$$\text{mín} \qquad \sum_{p=1}^{P} Coste_{i,j,p}\ \boldsymbol{y_p} \tag{4.28}$$

s.a.

$$\forall p \in P, \qquad \sum_{r=1}^{R} Peso_r\ \boldsymbol{x_{p,r}}\ \leq\ Cap_p\ \boldsymbol{y_p} \tag{4.29}$$

$$\forall r \in R, \qquad \sum_{p=1}^{P} \boldsymbol{x_{p,r}} = 1 \tag{4.30}$$

$$\forall p \in P, \qquad \boldsymbol{y_p} \in \{0,1\} \tag{4.31}$$

$$\forall p \in P, \forall r \in R, \quad \boldsymbol{x_{p,r}} \in \{0,1\} \tag{4.32}$$

4.4.2. Implementación

for *Todos los pares de tiendas (i,j)* **do**
  Se definen P de todos los paquetes que se envían $P = 1, \ldots, Y_{i,j} + ExtraBinPacks$;
  Resolver Packing;
  Se inicializa el coeficiente de ExtraBinPacks, Multiplicador1 = 2;
  while *Se utilicen todos los paquetes de P* **do**
    $P = 1, \ldots, Y_{i,j} + Multiplicador1 \cdot ExtraBinPacks$;
    Resolver Packing;
    $Multiplicador1 = Multiplicador1 + 1$;
  end
  Se inicializa el coeficiente de ExtraBinPacks, Multiplicador2 = 2;
  while *Problema sea infactible* **do**
    $P = 1, \ldots, Y_{i,j} + Multiplicador2 \cdot ExtraBinPacks$;
    Resolver Packing;
    $Multiplicador2 = Multiplicador2 + 1$;
  end
end

Algoritmo 1: Proceso de ejecución del Packing

4.5. Algoritmo TRP

Por comodidad, a la hora de referirnos a la ejecución consecutiva de las fases del algoritmo, nos referiremos a ejecución del **Algoritmo TRP**.

MejorSolucion = ∞;
Establecer el porcentaje de paquetes extra máximo para la optimalidad (minPerc);
repeat
  Generar aleatoriamente $\hat{c}_{i,j,r}$ *según el procedimiento descrito en la Ecuación 3.11*;
  Resolver Rounding, obteniendo $\boldsymbol{X}$, $\boldsymbol{Y}$ *y los paquetes extra añadidos, ExtraPacks*;
  CosteSolucionActual = ExtraPacks;
  if *El número de paques extra añadidos es inferior a minPerc* **then**
    MejorSolución = CosteSolucionActual;
    Guardar la solución de X e Y;
    break (se sale del bucle **repeat**);
  end
  if *CosteActual* $\leq$ *MejorSolución* **then**
    MejorSolución = CosteActual;
    Guardar la solución de $\boldsymbol{X}$ *e* $\boldsymbol{Y}$;
  end
until *No se supere el número máximo de iteraciones nIters*;
if *Se viola la restricción de demanda fija* **then**
  for *Todos los productos r* **do**
    for *Todas las tiendas i* **do**
      for *todas las tiendas j:* $j \neq i$ **do**
        if *Hay espacio suficiente en el paquete* $Y_{j,i,b}$ **then**
          if *La tienda j tiene el producto r disponible* **then**
            Enviar el producto r de j a i;
            Actualizar el valor del stock de i;
            Actualizar el valor del stock de j;
            Actualizar la saturación de $Y_{j,i,b}$;
          end
        end
      end
      if *Se sigue violando la demanda fija* **then**
        Ejecutar Heurística 2b;
      end
    end
  end
end

Algoritmo 2: Proceso de ejecución del Rounding

Capítulo 5

Heurísticas

Para poder valorar la calidad del proceso de reditribución de inventario propuesto, se describirán unas heurísticas sencillas para el caso de problemas en los que no hay demanda variable, y posteriormente se comparará con ellos. Para que la comparación sea sobre las mismas funciones objetivo, bastará tomar los parámetros $M_1 = 0$ y $M_2 = 0$ en el TRP. Es decir, las heurísticas nos permitirán valorar la calidad del procedimiento descrito en problemas sin demanda variable.

5.1. Heurísticas tipo 1

Con esta heurística se pretende mostrar que el hecho de que la fase de Rounding se resuelva a través de un problema de redondeo de matrices en lugar de resolverse redondeando simplemente cada uno de los términos, es porque hacer lo segundo provocaría problemas de factibilidad, pero además porque el Rounding permite tener en cuenta la función objetivo a la hora de hacer uno u otro redondeo. Es decir, el Rounding permite asegurar factiblidad y perseguir criterios de optimalidad al mismo tiempo

En esta heurística, en primer lugar se resolverá la fase de Transferring, después se realizará un redondeo a la alza (heurística 1a), uno a la baja (heurística 1b) o un redondeo puro (heurística 1c), y finalmente, se resuelve el Packing de la misma manera que en el TRP para determinar la composición de cada uno de los paquetes que se han decidido enviar. Los errores de factibilidad que se esperan encontrar para cada una de las heurísticas se mencionan a continuación:

- **Heurística 1a:** Violación del stock disponible.
- **Heurística 1b:** Insatisfacción de la demanda fija.
- **Heurística 1c:** Las dos anteriores.

5.2. Heurísticas tipo 2

Con estas heurísticas se pretende mostrar que se puede conseguir una solución factible fácilmente pero a costa de un alto coste de envío. Estas heurísticas prescinden totalmente de las fases de Transferring y Rounding y construyen una solución factible de modo iterativo. Para determinar la redistribución de los productos, una tienda i_0 recibirá un producto r_0 que demanda (a través de la demanda fija) de una tienda j cualquiera que disponga de dicho producto (y la propia tienda j no la demande). Después, se resuelve la fase del Packing tal y como se propone en el TRP para determinar la composición de los paquetes. Así, se proponen las siguientes variantes:

5.2.1. Heurística 2 a

El artículo r_0 se enviará desde la tienda que disponga de ella que primero encuentra el algoritmo. A pesar de que el conjunto de tiendas no sea ordenado, cada vez que el algoritmo hace un bucle sobre un conjunto hay un orden implícito, y por tanto, la primera tienda que disponga del artículo que la tienda i_0 demande será la que lo envíe. Podemos ver el pseudocódigo en el Algortimo 3.

5.2.2. Heurística 2 b

En este otro caso, el artículo r_0 no se enviará desde la primera tienda que tenga el producto disponible, sino que se enviará desde la tienda cuyo coste de envío del paquete de menor tamaño sea el más bajo, y que además disponga dicho artículo. Es decir, en primer lugar se comprueba para la tienda i_0 cuál de entre todas las tiendas j tiene el menor coste de envío para el paquete más pequeño (más barato), y después se comprobará si esa tienda j_0 tiene el artículo r_0 disponible.

Ahora, el orden sí que será importante y habrá que llevar una lista de las tiendas que han sido chequeadas. Podemos ver su pseudocódigo en el Algortimo 4.

for *Todas las tiendas* i **do**
    $Pendiente_{i,p,s} = FixDem_{i,p,s} - Stock_{i,p,s}$;
    if *Noy hay nada pendiente* **then**
        continue (se pasa a la siguiente tienda del bucle for)
    end
    for *Todas las tiendas* $j \neq i$ **do**
        $Disponible_{j,p,s} = \max(0, Stock_{j,p,s} - FixDem_{j,p,s})$;
        if *No hay suficiente Disponible* **then**
            Enviar lo disponible;
            Actualizar el $Stock_{j,p,s}$;
            Actualizar lo que queda $Pendiente_{i,p,s}$;
        else
            Enviar todo lo pendiente;
            Actualizar el $Stock_{j,p,s}$;
            Actualizar el valor de $Pendiente_{i,p,s} = 0$;
            break (se sale del bucle interno)
        end
    end
end

Algoritmo 3: Heurística 2 a

```
for Todas las tiendas i do
    Pendiente_{i,p,s} = FixDem_{i,p,s} − Stock_{i,p,s};
    if Noy hay nada pendiente then
        continue (se pasa a la siguiente tienda del bucle for)
    end
    repeat
        for Todas las tiendas j ≠ i do
            if El coste de enviar un paquete desde la tienda j a la tienda i es el mínimo
             de todas las tiendas j que aun no se han mirado then
                Disponible_{j,p,s} = máx(0, Stock_{j,p,s} − FixDem_{j,p,s});
                if No hay suficiente Disponible then
                    Enviar lo disponible;
                    Actualizar el Stock_{j,p,s};
                    Actualizar lo que queda Pendiente_{i,p,s};
                else
                    Enviar todo lo pendiente;
                    Actualizar el Stock_{j,p,s};
                    Actualizar el valor de Pendiente_{i,p,s} = 0;
                    break (se sale del bucle interno);
                end
                Actualizar las tiendas j miradas;
            else
                continue
            end
        end
    until No quede ninguna tienda por mirar or Pendiente[i,p,s] sea 0;
end
```

Algoritmo 4: Heurística 2 b

Capítulo 6

Interacción con la empresa y adaptación del modelo

En abril de 2016 se retoma el contacto con la empresa con la que Pablo Montero había tratado inicialmente para desarrollar su trabajo de fin de máster. Se organiza una reunión con la persona responsable en la empresa, en la que se le comunica el deseo por nuestra parte de continuar con el trabajo y los avances hechos hasta el momento. Dicha persona manifestó su interés en seguir colaborando en la medida de lo posible con el proyecto, y dejó en nuestras manos el concertar una nueva reunión cuando el algoritmo estuviera validado y listo para hacer alguna ejecución con datos reales.

En noviembre de 2016 nos volvimos a reunir con la persona responsable para mostrarle los avances realizados. Además, se comentaron algunos aspectos del modelo que podían interesar más a la empresa y dónde se creía que el algoritmo podría serles de más ayuda. Puesto que la idea de descentralizar el proceso completamente (evitar que el almacén fuera el centro de operaciones) podría no ser aceptable, se propusieron las siguientes opciones con las que ejecutar el modelo:

A Redistribución centralizada contra el almacén. Esta opción equivaldría a la manera de operación que actualmente siguen en la empresa, pero se le daría al usuario las pautas de llenado y envío de cada uno de los paquetes. Bastaría con modificar el archivo de datos para evitar los envíos entre tiendas y relajar la restricción 2.7 para el almacén (para que pueda enviar un producto que no tiene en stock pero que sabe que va a recibir). El tiempo de envío sera por tanto de mínimo 48 horas (24 horas de envío de las tiendas al almacén y otras 24 horas de envío del almacén a las tiendas).

B Redistribución 100 % descentralizada. Esta es la opción que se tiene por defecto en el TRP. Sería una opción de 24 horas.

C Una mezcla entre las opciones A y B. Como el proceso de redistribución que actualmente sigue la empresa es el A, las tarifas de envío a almacén y desde el almacén pueden ser más económicas que las de envíos entre cada par de tiendas. Por tanto, el algoritmo intentará encontrar el balance entre las dos opciones.

Puesto que ya se estaba en condiciones de lanzar el TRP con datos reales, también se solicitó información real acerca de la red. En concreto, se solicitó:

- Información acerca del **stock**: Cuántas unidades hay de cada producto en cada tienda en un determinado instante de tiempo, y cuántas unidades hay disponibles en el almacén.
- Información acerca de la **demanda fija**: unidades de cada producto que demanda cada tienda en ese instante.
- Información de la **demanda variable**: unidades de cada producto que una tienda estima que va a vender en ese instante.
- **Coste** de enviar un paquete a almacén de tienda y viceversa. Si hubiera varias compañías, sus correspondientes costes.
- **Coste** de enviar un paquete entre cada par de tiendas. En el caso de que no se tuviera un precio establecido (por no utilizar ese servicio) una estimación a la alta del coste.
- **Capacidades** de cada uno de los paquetes de los casos anteriores y el **peso** de cada una de las referencias (suponiendo que la capacidad de los paquetes se exprese en masa).

En respuesta se obtuvo un archivo excel que incluía información de cada tienda y en la que figuraban:

- Los datos a misma fecha de todos los artículos con sus **compras** por talla. Es decir, el $Stock_{i,r}$ de la tienda.
- Los datos a misma fecha de todos los artículos con sus **ventas** por talla. Es decir, la $FixDem_{i,r}$ de la tienda.
- Situación de Almacén Central a misma fecha.

Además, se nos comunicó el **precio** de enviar al almacén y desde el almacén cualquier tamaño de paquete era el mismo, es decir, tenían una tarifa plana con la empresa de mensajería; y también se nos proporcionó una estimación a la alta del envío de dos tamaños de paquetes entre cada par de tiendas.

Por otra parte, la capacidad de los paquetes se mide en número de unidades en lugar de en peso, por lo que no se nos proporcionó ninguna información acerca del peso de los productos y sí un número máximo de unidades por paquete. Esto no supone ningún problema sobre el modelado, porque bastaría poner que todos los productos tienen peso 1.

Finalmente, el concepto de demanda variable que se ha descrito en los capítulos anteriores es algo complicado de cuantificar para la empresa en la actualidad. Cuando se habla con el responsable en a empresa acerca de este parámetro, él nos sugiere otro con el que ellos trabajan allí: el **avance** de un producto.

6.1. Avance

Se define el **Avance** de un producto r para una determinada tienda i como el cociente entre las ventas y su Stock:

$$Avance_{i,r} = \frac{\sum_{s \in \mathcal{S}} FixDem_{i,r,s}}{\sum_{s \in \mathcal{S}} Stock_{i,r,s}},$$

donde $\mathcal{S}$ es el conjunto de tallas de una determinada referencia. Hasta ahora, se ha evitado el concepto de talla a lo largo de todo el informe pues a través de la referencia y de la talla quedaba definido un **producto**. Como en lo mencionado hasta ahora referencia y talla iban siempre asociados, por comodidad se ha preferido trabajar con el concepto de producto. Sin embargo en este caso se quiere tener un concepto de cuánto o cómo de bien se ha vendido una referencia independientemente de la talla.

Así, el $Avance_{i,r} \in [0, 1]$, tendrá un valor próximo a 1 para una determinada tienda i si el producto ha tenido una buena aceptación por parte de los clientes, y próximo a 0 si no.

Se observa que, a pesar de no disponer como tal del concepto de demanda variable, el avance podría servir como una buena estimación de ella: cabe esperar que aquellas tiendas que tengan un buen avance sobre una determinada referencia sigan vendiendo el artículo, lo que equivale a una demanda futura esperada alta (lo que antes era la demanda variable) y lo contrario sucede con las que tienen un avance bajo.

Una manera de modelar esta situación podría hacerse intentando equilibrar el avance a través de su evolución tras la redistribución: supóngase que una tienda tuviera un avance próximo a 1 sobre una determinada referencia porque hubiera tenido muchas ventas de la misma. Como ha tenido un alto número de ventas, se desearía que esa tienda recibiera más unidades de dicha referencia. Si después de la redistribución esa tienda recibiera más unidades, su avance final habría disminuido, porque para un mismo número de ventas tendría un mayor nivel de stock.

Y análogamente, si una tienda tuviera un mal avance sobre una referencia y se desearía que las enviara, tras la redistribución, para un mismo número de ventas tendría un menor nivel de stock, por lo que su avance aumentaría.

Por tanto, se trataría de intentar acercar todos los avances de todos las referencias y todas las tiendas a un entorno del **avance medio total**, que se denotará por $\overline{Avance}$. Así, si se desea que el avance tras los envíos esté por ejemplo a lo sumo un 5 % por debajo y un 10 % por encima del avance medio total,

$$\frac{\sum_{s\in\mathcal{S}} FixDem_{i,r,s}}{\sum_{s\in\mathcal{S}} Stock_{i,r,s} + Extra1_{i,r,s}} \;\geq\; (1-0.05)\;\overline{Avance} \tag{6.1}$$

$$\frac{\sum_{s\in\mathcal{S}} FixDem_{i,r,s}}{\sum_{s\in\mathcal{S}} Stock_{i,r,s} + Extra2_{i,r,s}} \;\leq\; (1+0.1)\;\overline{Avance} \tag{6.2}$$

donde $Extra1_{i,r,s}$ nos proporcionaría una cota sobre lo mínimo que una tienda i debe recibir de una referencia r y $Extra2_{i,r,s}$ una cota sobre lo máximo.

6.2. Adaptación del modelo

Se decide entonces adaptar el modelo, en particular la parte de Transferring, con el concepto de *Avance* para hacer una ejecución con los datos reales que se poseen. Se define la **demanda variable** (que irá en el objetivo del Transferring) como la cota inferior de lo que una tienda espera recibir de una referencia: $Extra1_{i,r,s}$. Despejando de la ecuación 6.1 se obtiene que

$$VarDem_{i,p,s} = \text{máx}\left(0\ , \frac{FixDem_{i,p,s}}{(1-\alpha)\ \overline{Avance}} - Stock_{i,p,s}\right),$$

siendo α el porcentaje que de separación respecto del avance medio total. Además, se pone el máximo entre 0 y dicho valor, pues es posible que para aquellos productos que tengan un avance muy bajo tome un valor negativo.

Por otra parte, se debe añadir como restricción que después las transacciones, un determinado producto no puede superar un porcentaje del avance medio total, y en caso de que quisiera superar ese determinado parámetro, lo haría a un coste alto. Así, despejando de la ecuación 6.2,

$$Extra2_{i,p,s} = \frac{FixDem_{i,p,s} + \epsilon}{(1+\beta)\ \overline{Avance}} - (Stock_{i,p,s} + \epsilon)$$

$$\forall i \in \mathcal{I},\ \forall r \in \mathcal{R},\ \forall s \in \mathcal{S}, \qquad \sum_j (\boldsymbol{X_{i,j,r,s}} - \boldsymbol{X_{j,i,r,s}}) \ \leq\ \text{máx}(0, Extra2_{i,r,s}) + \boldsymbol{H_{i,r,s}} \quad (6.3)$$

donde $H_{i,r,s}$ es una variable de holgura que penaliza en el objetivo en caso de que se quiera superar un más de $\beta\,\%$ del avance medio total. El ϵ es una herramienta para cuando las ventas toman el valor 0. Finalmente, denominaremos a $\alpha = \textit{RelDelay}$ y a $\beta = \textit{RelAhead}$.

Así, el modelo a resolver será el mismo que se expuso en las ecuaciones 2.1-2.9 con la nueva definición de demanda variable, añadiendo la restricción 6.3, añadiendiendo la variables de holgura $\boldsymbol{H_{i,r,s}}$ y su correspondiente penalización en el objetivo. Los subproblemas de Rounding y Packing se resolverán sin hacer ningún cambio.

Capítulo 7

Resultados numéricos

7.1. Soporte informático

Para la realización de este estudio se ha desarrollado un software de optimización en los lenguajes de programación AMPL y python que constan de:

- Varios modelos (archivos .mod) y archivos ejecutables (archivos .run) programados en lenguaje AMPL para la resolución del problema. En total entorno a 2000 líneas de código.
- Varios scripts de R para la generación de baterías de problemas y para la representación de los resultados. Entorno a 1500 líneas de codigo.
- Un automatizador de la lectura del archivo excel de datos reales suministrados por la empresa para su adaptación a los archivos de entrada de AMPL, programado en python. Unas 500 líneas de código.

Todos los archivos que se acaban de mencionar con todas las casuísticas posibles se ejecutan a través de un archivo en python que interactúa con AMPL, los solvers y R, y escribe los archivos de resultados (500 líneas de código).

Por otra parte, para hacer las ejecuciones se han utilizado las máquinas MARES (propiedad del Instituto Tecnológico de Matemática Industrial ITMATI) y MARISCAL (propiedad del Departamento de Matemática Aplicada) con el sistema operativo Linux. Ambas tienen tanto la licencia de AMPL y como la del solver *Gurobi* (en el caso de MARISCAL únicamente en uno de sus nodos), y tienen respectivamente 4 y 48 procesadores.

Gracias a la paralelización que el propio solver *Gurobi* tiene integrada, en particular para el branch and bound que hay que resolver en el problema de programación entera del Transferring, los tiempos de ejecución se reducen considerablemente. Y para un mismo tiempo, la calidad de la solución de problemas grandes es mejor.

Para no poner en riesgo el transcurso de las ejecuciones, pues estas pueden durar varias horas y su interrupción provocaría grandes pérdidas de tiempo, para evitar los fallos en cuanto a conexión de red se, ha hecho uso de la terminal *tmux* que permite seguir corriendo el código sin necesidad de tener la terminal abierta (en *background*). Y así, a pesar de que en caso de pérdida de conexión la terminal se cerrara, nuestro proceso seguiría ejecutándose.

Las representaciones gráficas se han hecho en su mayoría utilizando el software R, pero algunas de ellas se han hecho a través de la librería de python *plotly* disponible para los Jupyter Notebooks. La librería está disponible para R también, pero ya que la salida de datos es en python por comodidad se evita llamar a R de nuevo.

7.2. Simulaciones

7.2.1. Generación de batería de problemas

Para la validación del algoritmo TRP, se generan diferentes baterías de problemas para contrastar los resultados y comprobar que el proceso se desarrolla adecuadamente.

Dicha generación se hace a través de una función en R, que tiene como parámetros de entrada:

- **NIters** = Número de problemas que se desean generar. Todos los problemas tendrán las mismas características generales.
- **NStores** = Número de tiendas de la red.
- **NRef** = Número de referencias.
- **NSize** = Número de tallas.
- **NPack** = Número de tipos de paquetes.
- **MaxItem** = Máximo número de unidades de un producto por tienda. Es decir, unidades máximas a tener en tienda de una misma referencia y una misma talla.

A partir de estos datos, se generarán aleatoriamente, el coste de envío de los paquetes (un numero entero entre 50 y 100), la capacidad de cada uno de los paquetes (enteros y distintos entre 2 y 5), el peso de cada una de las referencias (entre 0 y 1), el stock de cada tienda de cada producto (entre 0 y MaxItem),... En definitiva todos los parámetros que están presentes en la resolución del Transferring.

Así, para cada uno de los problemas se generará un archivo de datos ".dat" que leerá AMPL. Para el caso particular que se va a presentar, se crea una batería de 50 problemas, con 7 tiendas, 7 referencias, 4 tallas, 2 tipos de paquetes y 5 productos de cada como máximo por tienda.

7.2.2. Resultados

Es mucha la información que se puede extraer de las ejeciciones, pues a pesar de que los conjuntos de variables sólo sean $\boldsymbol{Y_{i,j,p}}$, $\boldsymbol{X_{i,j,r}}$ y su correspondiente distribución en los paquetes, hay otros parámetros, como el porcentaje de demanda variable satisfecha o el número de paquetes extra que se añaden en cada una de las fases, que son de gran utilidad tanto para el usuario final como para la validación del TRP.

Por otra parte, cada uno de los 50 archivos de datos generados se ejecutará con distintas configuraciones de los parámetros M_1, δ y v. En particular, se harán ejecuciones con las distintas combinaciones de $M_1 = \{0, 0.1, 1, 10, 100, 1000\}$, $\delta = \{0.85, 0.9, 0.95, 1\}$ y $v = \{-1, 1\}$. El parámetro M_2 estará fijo en todas las iteraciones pues su papel es únicamente el de desempatar entre soluciones y no es un parámetro relevante para el usuario final.

A continuación se presentarán unas tablas con los distintos resultados obtenidos de las simulaciones, y se comentará la utilidad de cada una de ellas. Las tablas son las siguientes:

- Coste total de envío de paquetes y número de paquetes enviados tras cada una de las fases.
- Porcentaje de empeoramiento con respecto al mínimo coste obtenido en la mejor de todas las configuraciones, para cada una de las fases.
- Para las fases 2 y 3, el porcentaje de paquetes añadidos con respecto a los enviados en la primera fase.
- Distancia con respecto a la cota inferior del óptimo.

- Número de veces en las que el coste de envío final es el mínimo para distintas configuraciones de δ y porcentaje de demanda variable satisfecha.

Coste de envío y Paquetes. Tablas 7.1 y 7.2

En primer lugar se describirá la estructura de las tablas 7.1 y 7.2, pues hay mucha información en ellas y a simple vista pueden resultar difíciles de ver.

- En la columna de la izquierda representan las ejecuciones que se han realizado **sin** la fase 0, Transferring Relajado, que en notación del TRP significa que $v = -1$. En la columna de la derecha se representan las ejecuciones resolviendo en primer lugar la fase 0 y después la fase 1 con el parámetro ancho de ventana $v = 1$.

- En la primera fila se encuentran los resultados correspondientes a la fase 1, Transferring, en la segunda fila los resultados para el Rounding, y en la tercera los resultados para el Packing.

- Cada uno de los recuadros está formado por 24 celdas. Cada una de las celdas se corresponde con los resultados de la ejecución del TRP con una determinada configuración de los parámetros M_1 y δ. Así, la celda de la esquina superior izquierda, representa el resultado de la ejecucion del TRP para $M_1 = 0$ y $\delta = 1$.

- El degradado de colores indica en verde claro el valor más bajo de todo el recuadro y en azul oscuro el valor más alto.

Nótese que para estas tablas, y todas las tablas que se mostrarán a continuación, solamente se presentan los resultados para el parámetro ancho de ventana $v = 1$ del Transferring Relajado. Esto se debe a que en las pruebas que se realizaron se observó que para $v = 0$ se perdía calidad en la solución obtenida, y para $v = 2$ se obtenían soluciones similares a las de $v = 1$ y a cambio aumentaba el tiempo de cálculo debido al mayor rango de posibles valores de las variables $\boldsymbol{Y_{i,j,p}}$.

Se decide presentar juntas las Tablas 7.1 y 7.2 puesto que el coste total de envío va de la mano del número total de paquetes que se envía. Más aún, si hay un único tipo de paquete, ambas tablas serán equivalentes.

En cuanto al contenido de los recuadros, para la Tabla 7.1 en particular, en cada una de las celdas se muestra el **coste medio total de envío** de paquetes de las 50 simulaciones, en cada una de las fases y para cada una de las configuraciones de parámetros. Por ejemplo, el

590.68 de la celda superior izquierda del primero de los recuadros representa la media de los costes de envío de las 50 simulaciones tras la fase 1, para $M_1 = 0,\ \delta = 1$ y $v = -1$. El coste medio total de envío de paquetes en particular se abstrae del segundo y tercer término de la función objetivo (el solver lo utilizó, pero nos quedamos con el coste real que le supondría a la empresa la solución obtenida).

Análogamente, para la Tabla 7.2 cada celda representa el número de paquetes medio para cada una de las fases y para cada una de las configuraciones.

Analizando las tablas, se puede observar que a medida que M_1 crece, aumentan el coste y el número de paquetes que se envían. Esto se debe a que el porcentaje de demanda variable que se satisface es mayor (tal y como se verá en la Tabla 7.6). De hecho, como cabe esperar, el coste del Transferring es mayor cuanto menor es el δ pues se permite usar menos espacio en los paquetes. Sin embargo, no queda claro cual es la configuración ideal de δ, pues para cada uno de los M_1 se observan patrones diferentes (este hecho quedará mejor reflejado en la Tabla 7.6). Esto sugiere que cuando se vaya a proporcionar la herramienta al usuario final lo ideal sería darle varias soluciones con distintos valores de δ y M_1 y que él decida cuál es la que mejor se ajusta a sus necesidades.

Un hecho curioso que se puede observar es que en algunos casos, por ejemplo $M_1 = 0$ y $\delta = 0.85$, tanto el coste como el número de paquetes que se envían son menores al final del TRP que tras la resolución del Transferring. Para ilustrar este caso, supongamos que al resolver la primera fase con $\delta = 0.85$, para realizar una determinada transacción hubiera que utilizar 2 paquetes pequeños porque en uno grande al 85 % no hubiera espacio suficiente (pero en el grande al 100 % sí). El Rounding, a pesar de que se deshace de δ, no es capaz de reparar la situación. Sin embargo, cuando se resuelve el Packing de manera exacta, lo que se enviaba en las fases anteriores utilizando dos paquetes pequeños se envía en un paquete grande.

Nótese que esta situación no contradice el hecho de que el Transferring nos proporcione una cota inferior del óptimo, pues dicha cota se obtiene sólo cuando $\delta = 1$.

Por otra parte, como cabe esperar, los valores de la fase de Rounding son siempre superiores a los de la fase de Transferring para todos los casos, pues número de paquetes que se envían nunca se reduce en esta fase.

Por otra parte, como las dimensiones de los problemas resueltos no son muy grandes, apenas se puede notar la diferencia ente $v = -1$ y $v = 1$. Se esperaría, que con $v = 1$ la solución de los subproblemas fuera peor, pues al fin y al cabo no se está resolviendo el problema completo.

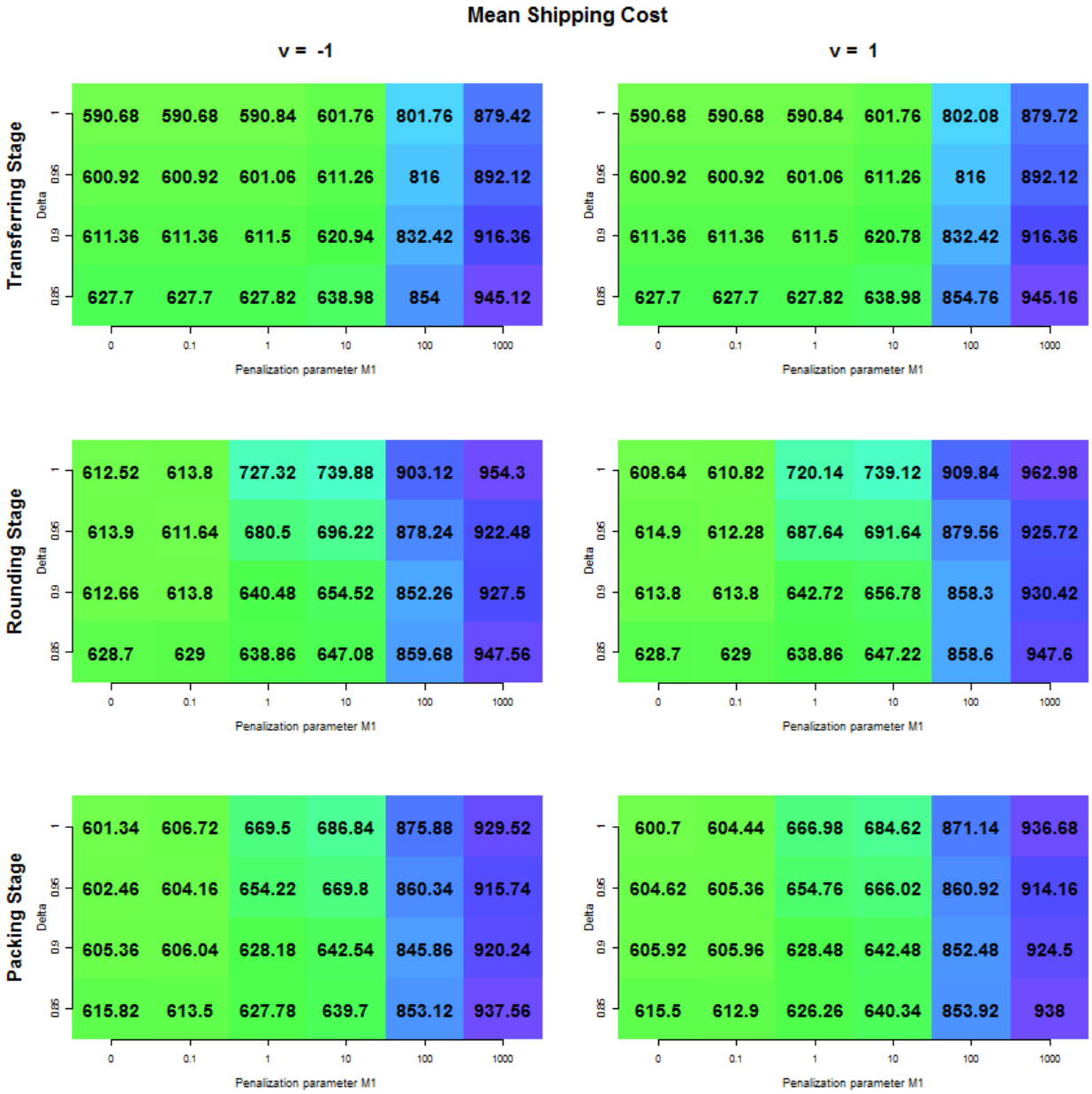


Tabla 7.1: Coste de envío medio del bloque de 50 simulaciones

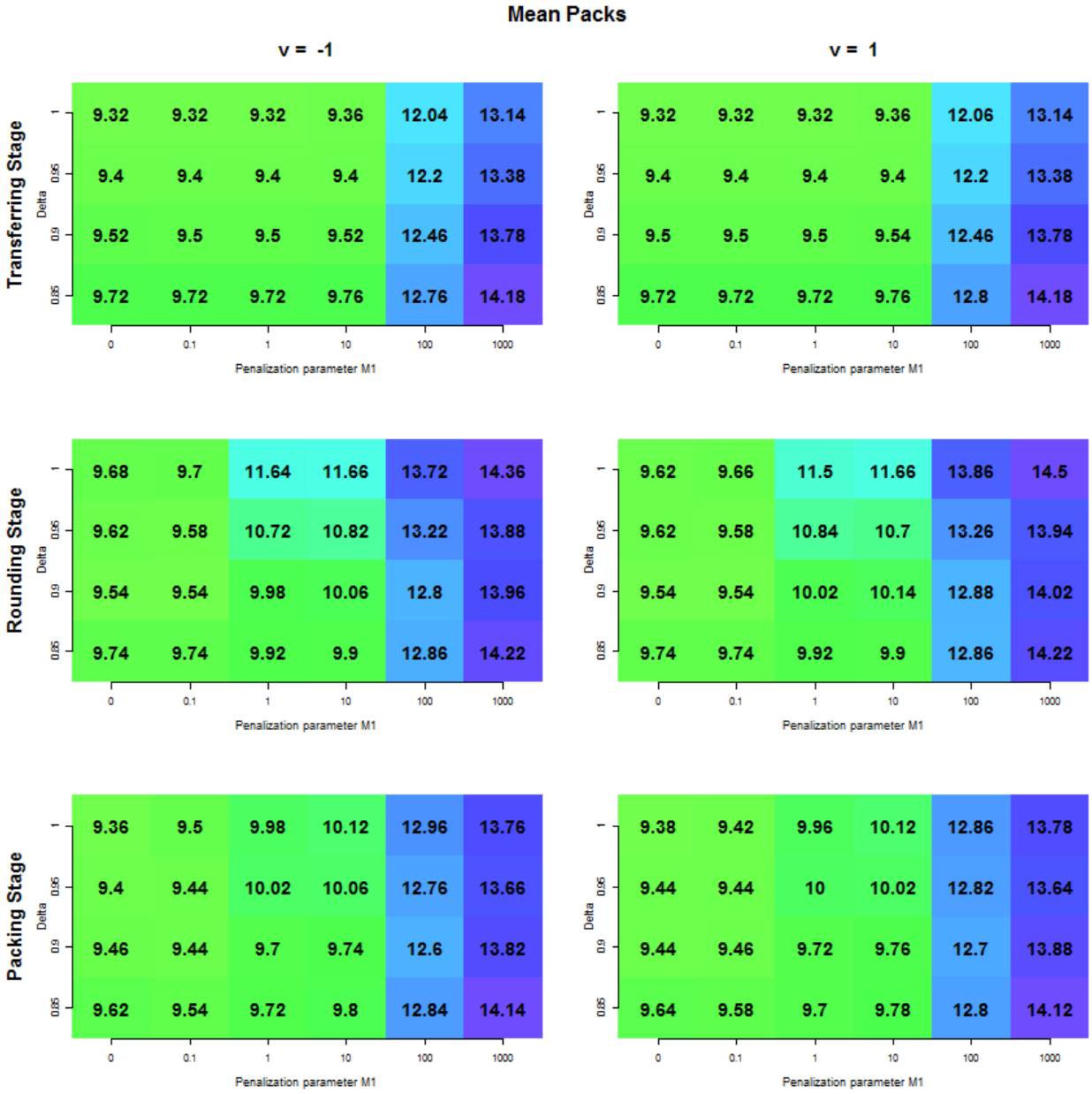


Tabla 7.2: Paquetes enviados

Porcentaje de empeoramiento con respecto al mínimo. Tabla 7.3

La estructura de la tabla es la misma que la de las dos tablas anteriores, pero en este caso el valor de cada celda es diferente. En cada componente se tiene el porcentaje de empeoramiento con respecto al mínimo coste obtenido en la mejor de todas las opciones.

Es decir, dada una simulación y una de las matrices de resultados, se tomará el menor de todos los valores de ella. Después, se le restará ese valor a cada una de las celdas de la matriz, y posteriormente cada una de las celdas se dividirá por él.

Las conclusiones que se sacan de esta tabla son las mismas que se sacan de las tablas anteriores, pues no es más que una transformación de los datos para ver los resultados de manera más clara. Así, al igual que en la Tabla 7.1 para el Transferring se ve que para $delta = 1$ los costes para $M_1 = \{0, 0.1, 1\}$ coinciden, en la Tabla 7.3 se ve de manera más directa que son los tres iguales y que además son los más bajos de toda lo el recuadro. Y que como además toman el valor 0, sigfica que lo han sido en toda las simulaciones.

Que en las otras dos fases no se observe ningún 0 indica que los valores de coste mínimos no siempre se han dado para la misma configuración de δ y M_1.

Porcentaje de paquetes extra con respecto a la fase 1. Tabla 7.4

Tal y como el propio nombre de la tabla indica, en cada celda se reprensenta para cada una de las configuraciones el porcentaje de paquetes extra añadidos con respecto a los enviados en la fase del Transferring para la misma configuración.

Como se ha comentado para la Tabla 7.1, para algunas de las configuraciones el número de paquetes que se envían en la fase del Packing son menores a las que se envían en el Transferring y por eso toman valores negativos.

Tiene sentido que para cada columna de cada uno de los recuadros el máximo se alcance para $\delta = 1$, pues es la que en la primera fase se resuelve con la capacidad total del paquete y por tanto no hay margen de rectificación.

Algo que llama especialmente la atención es que no se observe ninguna tendencia para M_1, pues alcanza su máximo en 1 y no en uno de los extremos como podría esperarse. Este

fenómeno se debe a la satisfacción de la demanda variable. Para $M_1 = \{0, 0.1\}$ la penalización por no satisfacerla es despreciable.

Sin embargo, a partir de $M_1 = 1$, no hacerlo penaliza. Cuando se resuelve el problema con $\delta = 0.85$ se satisfará obligatoriamente la demanda fija y parte de la demanda variable. Cuando se resuelve con $\delta = 1$, se satisface la misma demanda fija, pero se tiene un 15 % más de espacio para satisfacer la demanda variable. Y así, para un mismo coste de envío se envían más productos. Lo que sucede después es que los paquetes estarán bastante saturados porque se tienen más productos, y a la hora de hacer el redondeo se necesitarán más paquetes extra.

Por tanto, lo de que no se observe una tendencia no es totalmente cierto, porque si no se tienen en cuenta las dos primeras columnas, la tendencia es que a medida que M_1 aumente los paquetes extra se reduzcan, pues la penalización por no satisfacer la demanda variable es cada vez mayor y por tanto se intentará satisfacer en un alto porcentaje de demanda.

Distancia con respecto a la cota inferior del óptimo. Tabla 7.5

La resolución del subproblema del Transferring para $\delta = 1$ proporciona una cota inferior al óptimo al que se aspira. Con esta tabla se pretende ver cómo de lejos se encuentra cada una de las soluciones obtenidas en cada una de las fases de ese óptimo.

La estructura de la Tabla 7.5 es la misma que para la Tabla 7.1: Etapa por filas y v por columnas. Evidentemente, para la etapa del Transferring la fila correspondiente a $\delta = 1$ está formada por unos porque esas soluciones son precisamente las cotas inferiores. Para esta misma etapa, a medida que δ disminuye la distancia con respecto al óptimo aumenta, como es de esperar.

El comportamiento que se observa en las etapas de Rounding y Packing, especialmente para $M_1 = \{1, 10\}$, se corresponde con el mismo comportamiento que se veía en la Tabla 7.4, por tanto como se añaden más paquetes, el coste será mayor y por tanto más distancia se tendrá con el óptimo.

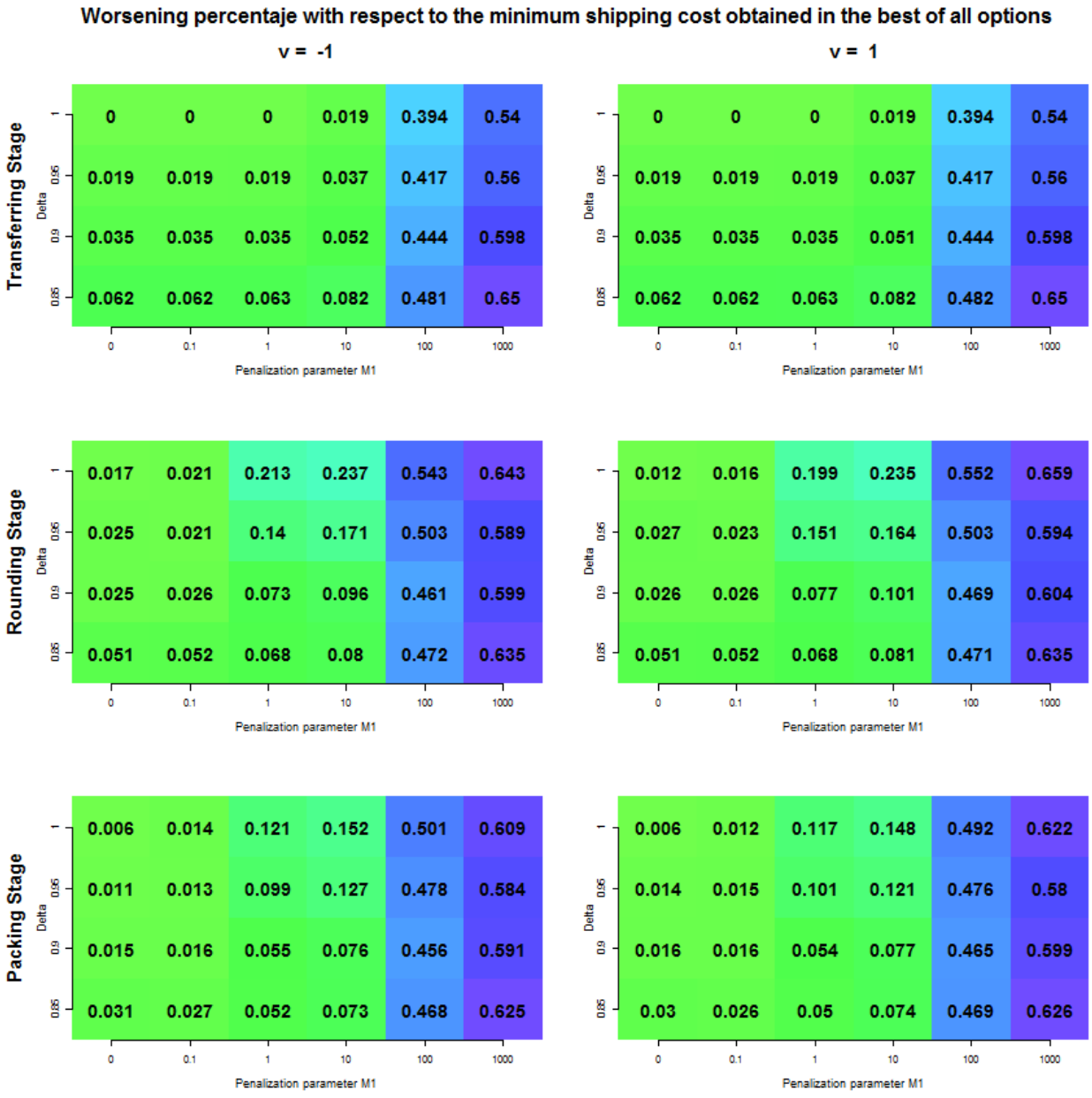


Tabla 7.3: Porcentaje de empeoramiento con respecto al mínimo obtenido en la mejor de todas las configuraciones

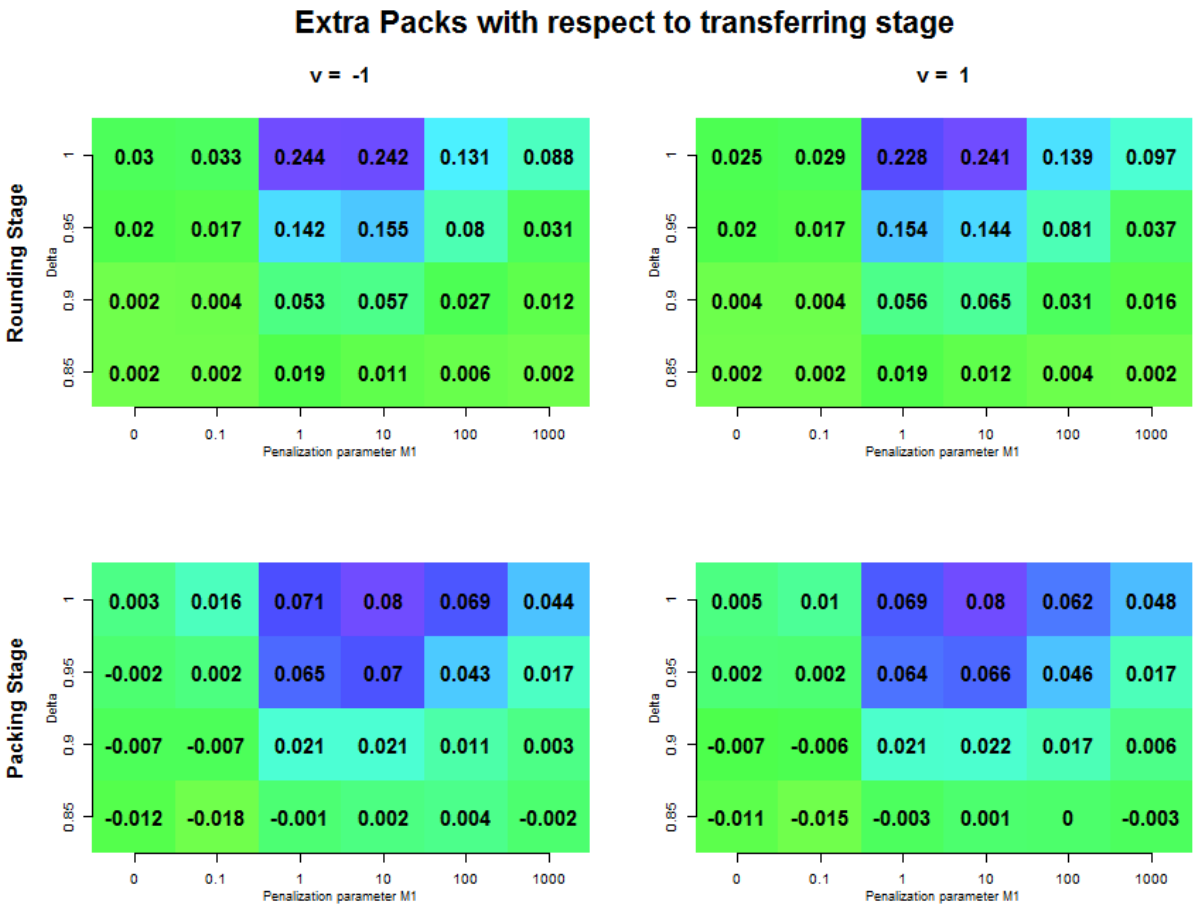


Tabla 7.4: Paquetes extra con respecto a la Etapa 1

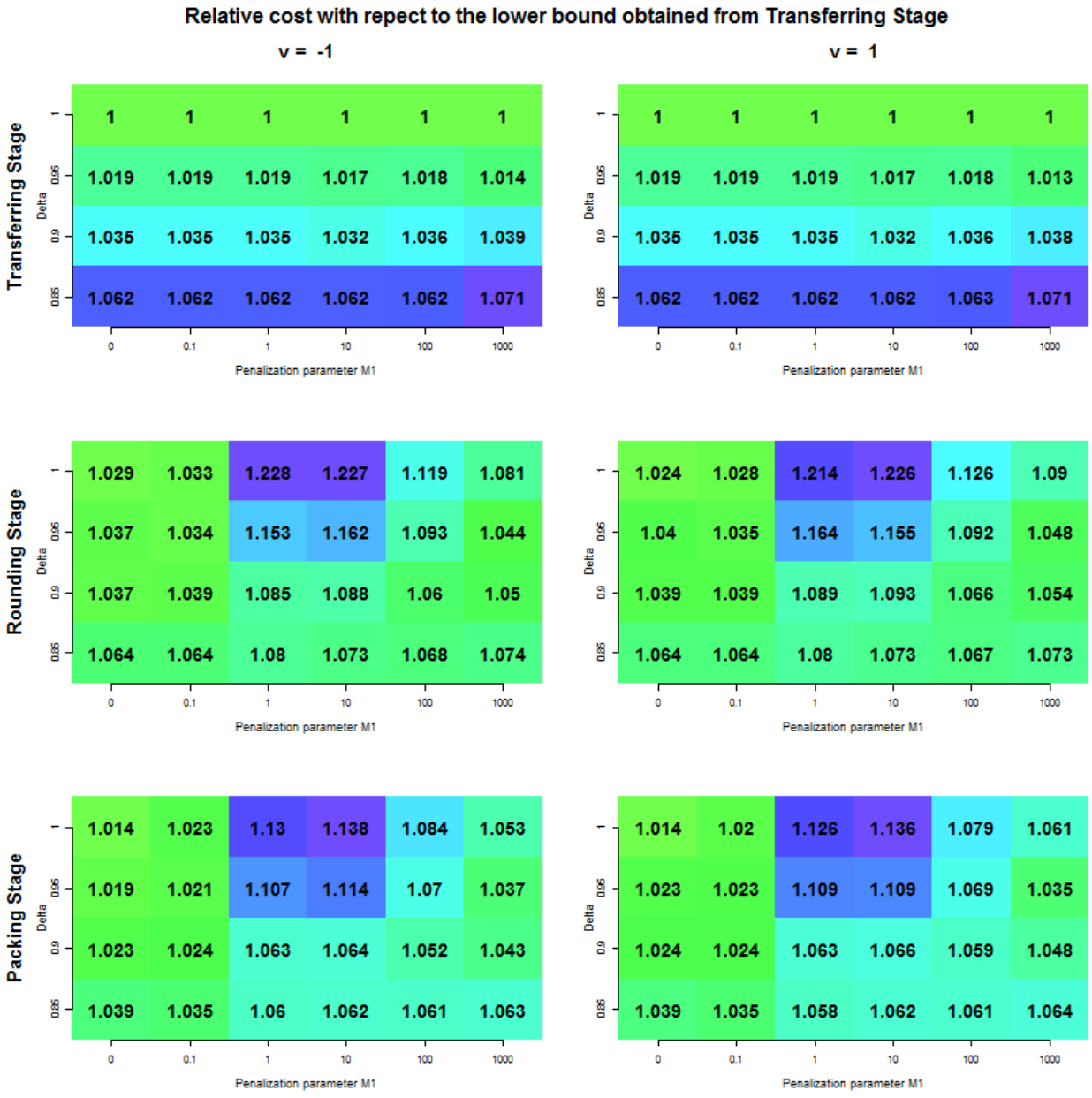


Tabla 7.5: Distancia con respecto a la cota inferior del óptimo

Análisis de δ y porcentaje de demanda variable satisfecha. Tabla 7.6

La estructura de esta última tabla es diferente con respecto a las que se han mostrado anteriormente, pues en esta sólo se muestran los resultados finales. Es decir, después de haber resuelto la fase 3. Por columnas, de nuevo, en la de la izquierda se representan los resultados sin haber resuelto la fase 0, y en la de la derecha habiéndolos resuelto con $v = 1$.

En la primera fila se muestra el porcentaje de veces que, para el mismo valor de M_1, el valor es mínimo. Es decir, se chequea si una celda determinada contiene el valor mínimo de su columna, y se cuenta el número de veces que lo hace. El hecho de que los valores de cada columna no sume 1 no es un error, pues puede que para varios valores de δ se consiga la misma solución y por tanto contabilizaría para todas. Por supuesto, los valores que se representan son una media de las 50 simulaciones.

En la segunda fila se muestra el porcentaje de veces en las que cada celda toma el valor mínimo por columna, y además es el único que lo hace. Es decir, que la solución que se obtiene es mínima y no es la misma que se obtiene con los otros valores de δ.

Con estas dos tablas se pretende ilustrar que el hecho de introducir el parámetro δ en el modelo marca la diferencia, pues para distintos de valores de M_1 el valor de δ que proporciona la solución óptima varía.

Finalmente, en la última fila se representa la tabla de demanda variable satisfecha, cuyos valores aumentan de izquierda a derecha, y de abajo a arriba (exceptuando para $M_1 = \{0, 0.1\}$ en los que no se cumple porque la satisfacción de demanda variable no importa).

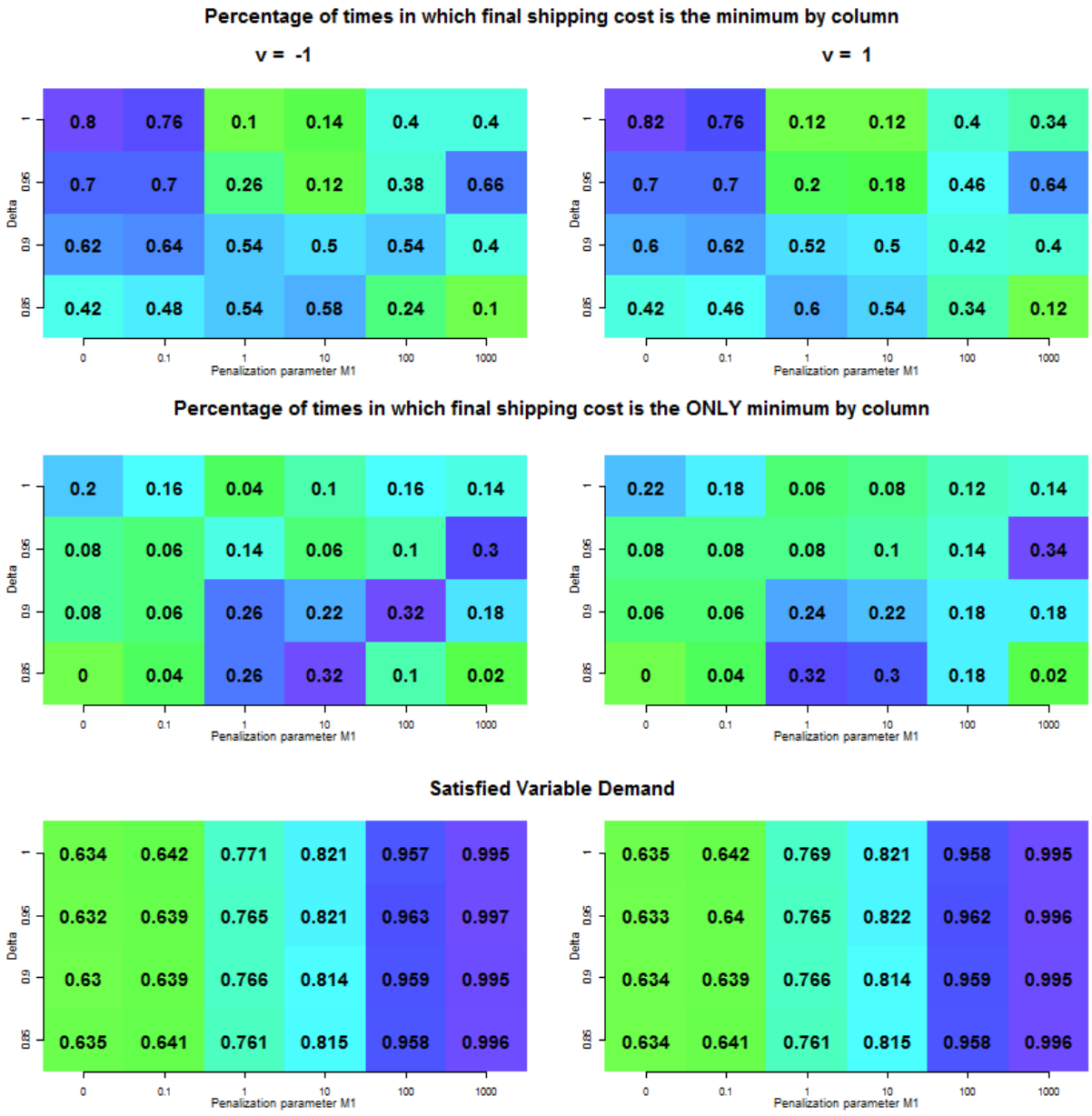


Tabla 7.6: Análisis de δ y porcentaje de demanda variable satisfecha

7.2.3. Comparaciones con las heuristicas

A continuación se mostrarán unas pequeñas tablas de resultados en las que se pretende ilustrar que las heurísticas se comportan peor que el TRP. Estos resultados se han obtenido tras la ejecución de la misma batería de problemas. Recordamos también que los resultados de las heurísticas sólo son comparables cuando $M_1 = 0$, pues al ejecutarlas no se tiene en cuenta la demanda variable.

Heurísticas tipo 1

Con las heurísticas 1 se pretende mostrar que no se puede hacer un simple redondeo en la fase 2, pues esto produciría problemas de factibilidad. En la Tabla 7.7 se puede ver el porcentaje de demanda fija insatisfecha y en la Tabla 7.8 el porcentaje de violación de stock.

	Heurística 1a	Heurística 1b	Heurística 1c
Delta = 0.85	0.0007142857	0	0
Delta = 0.9	0.0004081633	0	0
Delta = 0.95	0.0003061224	0	0
Delta = 1	0.0000000000	0	0

Tabla 7.7: Porcentaje de demanda fija insatisfecha para las heurísticas 1a, 1b y 1c

	Heurística 1a	Heurística 1b	Heurística 1c
Delta = 0.85	0.0000000000	0.03253683	0.000000000
Delta = 0.9	0.0003030303	0.03197510	0.001060606
Delta = 0.95	0.0003030303	0.03000858	0.000000000
Delta = 1	0.0000000000	0.02790858	0.000000000

Tabla 7.8: Porcentaje de violación de stock para las heurísticas 1a, 1b y 1c

Se puede observar que si una de las heurísticas no viola la restricción de satisfacción de demanda fija entonces no respetará el stock y viceversa.

Heuristicas tipo 2

Debido a que en las heurísticas 2 tampoco se hace uso del parámetro δ, lo que se representa en la Tabla 7.9 en la columna TRP se corresponde con el peor resultado de entre los obtenidos para los 4 valores de δ. Como estas heurísticas aseguran la factibilidad, la comparación se hace en función objetivo.

	TRP	Heurística 2a	Heurística 2b
Coste	615.82	930.22	807.66
Paquetes	9.62	13.36	13.06

Tabla 7.9: Coste, número de productos y paquetes para las heurísticas 2a y 2b

Como se puede observar, el resultado de ejecución de las heurísticas 2 está muy lejos del peor obtenido por el Transfer. Como se esperaba, se observa que el resultado de la heurística 2b es mucho mejor que la de la 2a.

7.3. Aplicación a datos reales

Antes de comenzar a presentar los resultados obtenidos de la ejecución de los datos reales, se procede a la descripción de la dimensión del problema:

- Número de tiendas: 34 incluyendo el almacén
- Número de referencias: 362
- Número de tallas: 12
- Número de tipos de paquete: 2

Como puede observarse, la dimensión del caso real es mucho mayor a la de las simulaciones del apartado anterior. Esto se traducirá en un mayor tiempo de ejecución en todas las fases, pero en particular para la etapa del Transferring. De hecho, es posible que la ejecución dure días hasta que encuentre la solución óptima del Transferring, si la encuentra.

Por tanto, se decide poner un límite de tiempo por ejecución para que el proceso se detenga en un tiempo razonable. Se propuso un límite de 8 horas por ejecución, estimando que es el tiempo máximo que la empresa estaría dispuesta a invertir para resolver el problema (se

dejaría ejecutando la noche anterior y a la mañana siguiente se obtendrían los resultados). En caso de que la ejecución se detenga por límite de tiempo, el solver proporcionará la mejor solución que haya encontrado hasta el momento y la mayor cota inferior de la solución que se ha encontrado.

En la Figura 7.1 se representa para una ejecución del Transferring la evolución de la función objetivo y la de la mejor cota inferior de la solución. En el eje vertical se representa la función objetivo en miles de euros, y en el eje horizontal el tiempo transcurrido en miles de segundos. En la Figura 7.2 se muestra la distancia relativa entre estos dos valores, a la que llamaremos GAP. En el eje vertical se representa el valor GAP en unidades y en el horizontal el tiempo en miles de segundos.

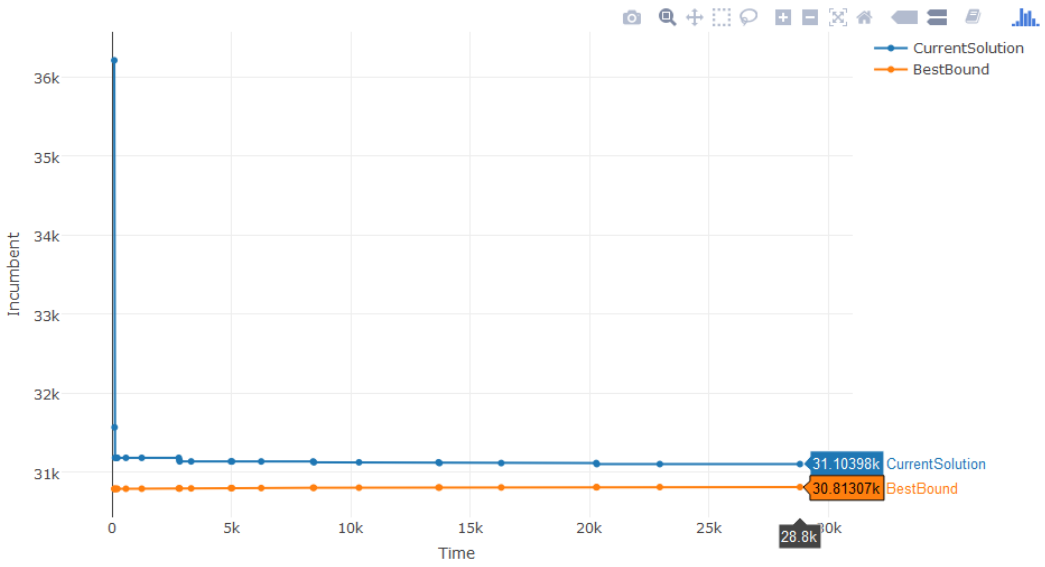


Figura 7.1: Representación de la mejor cota y de la función objetivo por tiempo

A la vista de estas gráficas se tienen muy buenas noticias, porque en menos de un minuto y medio de ejecución se consigue estar a menos de un 0.015 % de la mejor solución, y en menos de una hora por debajo del 0.01 %. Por tanto, se obtendrán buenas soluciones con poco tiempo de ejecución.

Las gráficas se han representado utilizando la librería *plotly* disponible para python y para R, que permite hacer de manera rápida gráficas interactivas. En la Figura 7.3a se puede ver una captura de la posibilidad de ampliación de la gráfica, y en la Figura 7.3b el resultado.

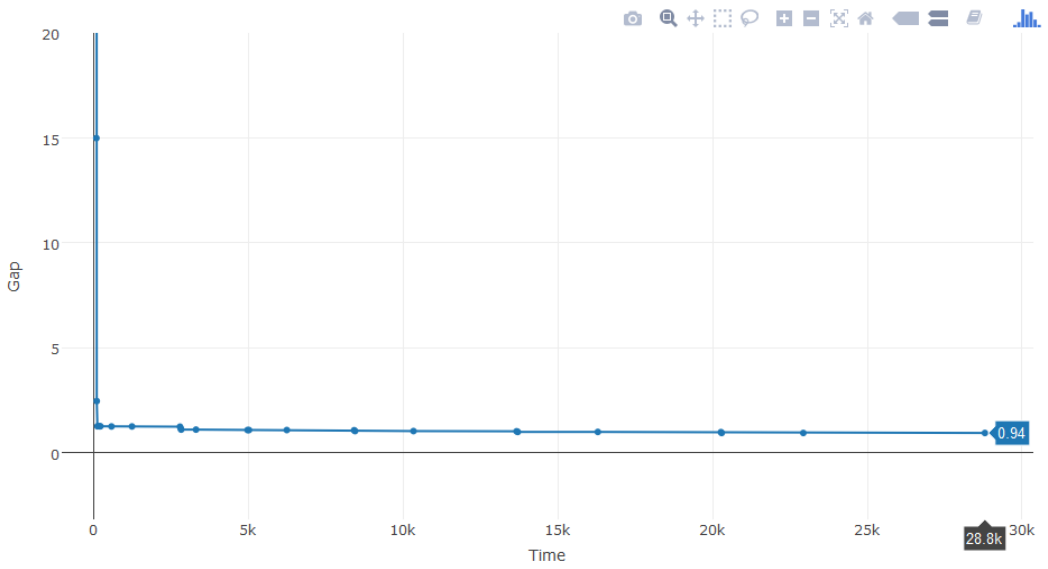
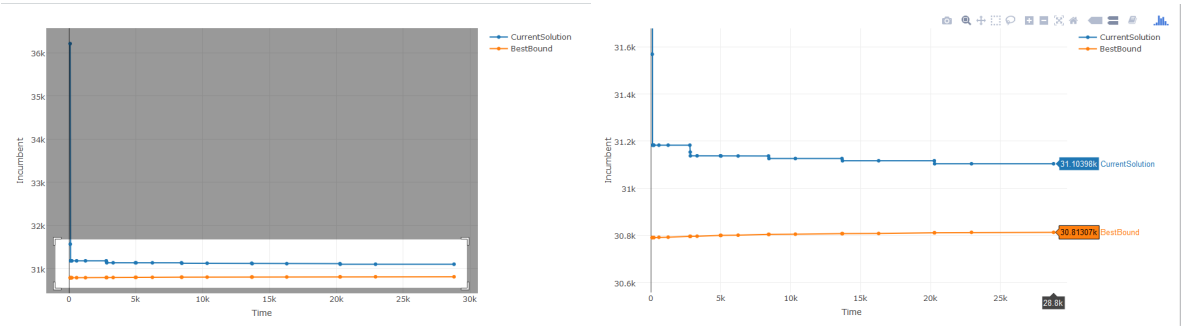


Figura 7.2: Representación del GAP relativo con respecto al tiempo

Además, a medida que el cursor se pasa por encima de las líneas se muestra el valor que está tomando en cada uno de los ejes.



(a) Interactividad de la gráfica

(b) Ampliación de la gráfica

Figura 7.3: Muestra de la interactividad de la librería plotly

Visto el efecto de δ y M_1 en las simulaciones, se decide ejecutar este caso real para $\delta = 0.85, 0.95$ y $M_1 = \{1, 100\}$.

A continuación se muestran los resultados tras el Packing de las ejecuciones del algoritmo en modos A (Tabla 7.10) y B (Tabla 7.11) a modo ilustrativo, pues el modelo para datos reales en modos A y B está aún en fase de validación. No se presenta ningún resultado sobre las ejecuciones en modo C, pues está aún en fase de desarrollo.

M1	Delta	Coste	Paquetes	% DemVar	Items	ElapsedTime	SolveTime
1	0.85	3512.2999	682	0.15482	32207	322.40942	375.948354
1	0.95	3630.7499	705	0.15577	33411	55.896352	82.573712
100	0.85	5495.0499	1067	0.21702	51369	270.09863	467.840336
100	0.95	5412.6499	1051	0.21455	51370	218.86255	393.14994

Tabla 7.10: Resultados de la ejecución del Packing para datos reales en modo A

M1	Delta	Coste	Paquetes	% DemVar	Items	ElapsedTime	SolveTime
1	0.85	3988	200	0.09604	8462	715.13444	583.69226
1	0.95	3880.15	329	0.19113	14963	376.35463	419.86817
100	0.85	14561.95	1064	0.340982	29970	560.99454	735.84913
100	0.95	19268.8	1397	0.341737	29982	619.35876	935.84673

Tabla 7.11: Resultados de la ejecución del Packing para datos reales en modo B

En las tabla anteriores, **% DemVar** representa el porcentaje de demanda variable que se ha satisfecho, **Items** el número total de productos que se han enviado, **ElapsedTime** el tiempo que ha transcurrido y **SolveTime** el tiempo que se ha invertido entre todos los procesadores en resolver el problema. El hecho de que se contabilicen dos parámetros para el tiempo es por el hecho de que las resoluciones se están haciendo utilizando varios procesadores. En el **SolveTime** se contabilizará el tiempo total de resolución, esto es, la suma del tiempo que han tardado todos los procesadores, y en el **ElapsedTime** el tiempo real transcurrido. El tiempo de resolución total siempre será superior al tiempo transcurrido.

Cabe mencionar que los datos reales con los que se han hecho las ejecuciones tienen una peculiaridad que posiblemente haga que los resultados que se obtengan no sean los deseables. Se trata de que cada una de las tiendas tiene stock suficiente como para satisfacer toda su demanda fija. Esto provoca que los únicos productos que se envíen sean para la satisfacción

de la demanda variable.

En primer lugar, se observa que los resultados de la ejecución del algoritmo en el modo A difieren mucho de los obtenidos en el modo B. En A, los costes son inferiores en todos los casos y el volumen de productos enviados mayor. A pesar de que el coste sea siempre menor, cuando $M_1 = 1$ y el coste de ambos es similar. Sin embargo, el número de paquetes enviados (y por tanto el número de productos desplazados) es mucho mayor. Es decir, para un mismo coste, el modo A mueve un mayor volumen de productos.

Para $M_1 = 100$ la no satisfacción de la demanda variable tiene un peso mayor en el objetivo. Se observa que en ambos el número de paquetes que se envía es similar, pero que a A le sale mucho más barato enviarlos. Por otra parte, A mueve muchos más productos que B, porque los paquetes que utilice A serán mayores que los que utiliza B. Estos resultados posiblemente se den a causa de la tarifa plana que la empresa tiene contratada con las empresas de mensajería, ya que el precio de enviar a almacén y desde el almacén es mucho menor para todos los tamaños de paquete que el que se ha estimado que se tendría para el envío entre tiendas.

Capítulo 8

Conclusiones

A lo largo de este trabajo se ha presentado y validado un algoritmo para la resolución de un problema de redistribución de inventario para una red de tiendas. Para ello, en primer lugar se ha descrito cada una de la componentes del problema, se ha descrito cada uno de los subproblemas a resolver, y finalmente se ha procedido a su validación. De los resultados obtenidos, las conclusiones que se pueden sacar son las siquientes:

- La unimodularidad del redondeo de matrices es de vital importancia porque permite resolver problemas con variables enteras sin necesidad de acudir a técnicas de programación entera.
- Aún así, no siempre se encuentra una solución factible tras la resolución del problema de redondeo, por lo que hay que hacer un postprocesado de la solución.
- Se ha demostrado que la introducción del parámetro δ que representa el porcentaje de llenado de cada paquete en la fase 1 es de gran utilidad, pues se ha comprobado que no es siempre el mismo valor de ese parámetro el que proporciona el mejor resultado.
- El cuello de botella del algoritmo se encuentra en la fase 1. La incorporación de la fase 0 puede ayudar a que el tiempo de ejecución se reduzca, pero posiblemente a cambio de obtener una solución de peor calidad.
- A pesar de que el tamaño del problema real fuese grande, en poco tiempo de ejecución se puede conseguir una buena solución.

Finalmente, comentar que la empresa sigue interesada con la propuesta y que es probable que se ponga en marcha una colaboración formal entre la empresa e ITMATI, para que alguien continúe con el desarrollo de una herramienta completamente adaptada sus necesidades.

Bibliografía

[1] TW Archibald. Modelling replenishment and transshipment decisions in periodic review multilocation inventory systems. Journal of the Operational Research Society, 58(7):948-956, 2007.

[2] F. Caro and J. Gallien. Inventory management of a fast-fashion retail network. Operations Research, 58(2):257-273, 2010

[3] F. Caro, J. Gallien, M. Marcos Montes, J. Ramos, and J. Correa. Zara uses operations research to reengineer its global distribution process. Informs, 40(1):71-84, 2010.

[4] J. Correa. Optimization of a fast-response distribution network. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2007.

[5] Kwan Eng Wee and Maqbool Dada. Optimal policies for transshipping inventory in a retail network. Management Science, 51(10):1519-1533, 2005.

[6] Christodoulos A. Floudas and Panos M. Pardalos. Encyclopedia of optimization, volume 1. Springer, 2008.

[7] Jason Hu, Edward Watson, and Helmut Schneider. Approximate solutions for multilocation inventory systems with transshipments. International Journal of production economics, 97(1):31-43, 2005.

[8] A. Garro. New product demand forecasting and distribution optimization: a case study at zara. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2011.

[9] Julio González Díaz. Apuntes de la materia de Programación Lineal y Entera. Universidade de Santiago de Compostela, Santiago de Compostela, 2014. Máster de Técnicas Estadísticas.

[10] Silvano Martello and Paolo Toth. Knapsack problems. Wiley New York, 1990.

[11] N. Tokatli. Global sourcing: insights from the global clothing industry the case of zara, a fast fashion retailer. Journal of Economic Geography, 8(1):21-38, 2008.

[12] M. A. Vega González. Stock level optimization at the distribution center through improved supply management practices. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2009.