Problemas de rutas con recogida y entrega de mercancías
Aplicación a una empresa canaria

Vehicle Routing Problems with Pickups and Deliveries
Aplication to a Canarian company

Vanessa León Osta

Trabajo de Fin de Grado

Matemáticas, Estadística e Investigación Operativa

Sección de Matemáticas. Facultad de Ciencias

Universidad de La Laguna

La Laguna, 16 de junio de 2016

Dr. D. **Hipólito Hernández Pérez**, con N.I.F. 45.452.715-T profesor adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

"*Problemas de rutas con recogida y entrega de mercancías.*"

ha sido realizada bajo su dirección por D. **Vanessa León Osta**, con N.I.F. 78.759.042-G.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 16 de junio de 2016.

(a) Hipólito Hernández Pérez

(b) Vanessa León Osta

Agradecimientos

En primer lugar, quiero mostrar mi agradecimiento a mi tutor, Hipólito Hernández Pérez, por toda la paciencia y tiempo que me ha prestado durante la realización de este trabajo.

A todos los profesores que han contribuido tanto en mi desarrollo académico como personal.

A mi familia por su constante apoyo durante estos años.

A Joaquín por su infinita paciencia.

Resumen

Los problemas de rutas de vehículos surgen de forma natural como problemas centrales en los campos de transporte, distribución y logística. En ellos, en general, se trata de averiguar las rutas de una flota de transporte para dar servicio a unos clientes. Este tipo de problemas pertenecen a la rama de las matemáticas conocida como optimización combinatoria. En la literatura científica, Dantzig y Ramser (1959) fueron los primeros autores en analizar estos problemas cuando en 1959 estudiaron la aplicación real en la distribución de gasolina para estaciones de carburante.

En este contexto el presente documento presenta una revisión de los diferentes problemas de rutas de vehículos. Centrándonos después en los problemas de recogida y entrega de mercancías. Se propone una formulación del "one-to-one Pickup and Delivery Problem" aplicada a la empresa canaria Desguaces Tenerife S.A., se implementa en Gusek y Xpress (dos resolutores de problemas de programación lineal) y se comentan los resultados.

El problema de rutas que plantea esta empresa canaria tiene factores aleatorios. Simplificar el problema considerando estos factores como determinísticos ayuda a determinar una estrategia en el diseño de las rutas. Aún así, debido a la complejidad del problema no somos capaces de resolverlo óptimamente para más de 8 pedidos, por tanto, menos aún para los 50 pedidos diarios que se producen en la empresa. Es por esto que se decide plantear nuevas formulaciones en las que las rutas de los vehículos sean fijas y tengamos que decidir únicamente los tiempos de espera en cada localización y la localización inicial de los vehículos. Se implementan en Xpress y, finalmente, se comentan los resultados obtenidos.

Palabras clave: Rutas de vehículos · Recogida y entrega · Viajante de comercio · Mercancías

Abstract

The vehicle routing problems arise naturally as central problems in the fields of transportation, distribution and logistics.

In general, it comes to figuring out the routes of a transportation fleet to service some customers. Such problems belong to the area of mathematics known as combinatorial optimization. In the scientific literature, Dantzig y Ramser (1959) were the first authors to analyze these problems when in 1959 they studied the actual application in gasoline distribution for fuel stations.

In this context, this paper presents a review of different vehicle routing problems. Then we focus on Pickup and Delivery problems. A formulation of the one-to-one Pickup and Delivery Problem applied to the Canarian company Desguaces Tenerife S.A. is proposed. The model is implemented in Gusek and Xpress (two solvers of linear programming problems) and then we discuss the results.

The routing problem posed by this company has random factors. Simplify the problem by considering these factors as deterministic helps to determine a strategy in designing the routes. Even so, due to the complexity of the problem, we can not solve it optimally for more than 8 orders, therefore, even less for the 50 orders per day produced in the company. This is the reason why we decided to think of new formulations in which the routes of the vehicles are fixed and we only have to decide the waiting times at each location and the initial location of the vehicles. They are implemented in Xpress and finally the results are discussed.

Keywords: Vehicle routing · Pickup and delivery · Traveling salesman · Commodity

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Uno de los problemas típicos y más importantes de la logística del transporte son los de rutas de vehículos.

Los problemas de rutas de vehículos (Vehicle Routing Problems (VRPs) en inglés) es el nombre genérico dado a la clase de problemas en los que se busca la optimización (normalmente reducción de costes, de tiempo o de vehículos) de un conjunto de rutas a realizar por una flota de vehículos localizados en uno o más depósitos, que conjuntamente deben dar servicio (recogida/entrega de mercancías o ambas cosas) a un cierto número de ciudades o clientes geográficamente dispersos.

Son problemas muy conocidos y estudiados en la programación lineal entera, que caen en la categoría denominada $\mathcal{NP}$-duro, esto es, los problemas que no pueden resolverse en un tiempo polinomial. El tiempo y esfuerzo computacional requerido para resolver este problema aumenta exponencialmente respecto al tamaño del problema, es decir, la cantidad de nodos a ser visitados por los vehículos. Para este tipo de problemas es a menudo deseable obtener soluciones aproximadas, para que puedan ser encontradas suficientemente rápido y que sean suficientemente buenas para llegar a ser útiles en la toma de decisiones.

Existen muchos paquetes de software disponibles en el mercado que difieren en muchos aspectos. Vienen con diferentes licencias y, por supuesto, diferentes características. Los resolutores libres y de código abierto más populares son Glpk, LP Solve, Clp, Scip, SoPlex. Entre los resolutores comerciales encontramos Cplex, Xpress y Gurobi. En el artículo publicado por Meindl y Templ (2013) se encuentra un análisis de estos resolutores. En este trabajo nosotros utilizaremos Glpk (utilizando el entorno denominado Gusek) y Xpress.

La motivación para la realización de este trabajo fue una tarea planteada durante la realización de las prácticas externas en la empresa Desguaces Tenerife S.A.

En este proyecto nos centramos en un caso particular del problema de rutas de vehículos que es el de recogida y entrega de mercancías. El trabajo se distribuye de la siguiente manera. En lo que resta del capítulo 1 describimos la empresa Desguaces Tenerife S.A. y comentamos la motivación y objetivos del proyecto. En el capítulo 2 se presentan los fundamentos teóricos que apoyan el tema objeto de la investigación. La formulación del problema para el caso estático así como los resultados computacionales y un análisis de los mismos aparecen en el capítulo 3. En el capítulo 4 se introducen formulaciones del problema

de recogida y entrega de mercancías con las rutas de los vehículos fijas además de los resultados computacionales. Finalmente, en el capítulo 5 se proporcionan las conclusiones obtenidas.

1.1. Desguaces Tenerife S.A.

Desguaces Tenerife S.A. es una empresa canaria pionera en el mundo del desguace y el reciclaje del automóvil. Desde que fue fundada en el año 1977 se ha encontrado en constante adaptación a las nuevas tecnologías y legislaciones de respeto medioambiental.

Desguaces Tenerife S.A. ofrece cuatro servicios principalmente.

- Desguace. Una vez descontaminados los vehículos fuera de uso, se procede a su desarme seleccionando las piezas que cumplan con los controles de calidad. Posteriormente se procede a almacenar dichas piezas para su posterior venta.
- Venta de repuestos nuevos, a los que la empresa denomina Repuestos Cero.
- Reciclaje de materiales. Desguaces Tenerife S.A, tiene capacidad tecnológica para reciclar Vehículos Fuera de Uso (VFU), materiales férricos y no férricos, baterías y elementos eléctricos y electrónicos.
- Exportación de los materiales reciclados.

1.2. Motivación y Objetivos

Durante la realización de las prácticas externas en la empresa, se planteó una tarea que consistía en analizar y resolver un problema de rutas cuyo objetivo era optimizar las rutas de los vehículos que transportan repuestos de un centro a otro.

Desguaces Tenerife S.A. está formada por siete centros distribuídos por la Isla de Tenerife. Tres de estos centros están localizados en la zona Norte (Icod de los Vinos, Los Realejos y La Victoria de Acentejo), dos en la zona Metropolitana (Geneto y Taco), y dos en la zona Sur (Güímar y Arona) (Figura 1.1).

La empresa posee dos vehículos que parten de dos de los centros de la isla (Icod de los Vinos y Arona). La planificación que seguían antes del estudio era, una ruta norte, otra ruta sur y un centro de intercambio (Geneto), cada vehículo realizaba su ruta dos veces al día. Con este sistema se estaban recibiendo quejas puesto que, pedidos que habían sido realizados temprano en la mañana, a la hora del cierre de la empresa no habían podido ser entregados en el centro de destino. Así que, lo que se deseaba saber era si existía alguna alternativa mejor con la cual aumentase el número de pedidos completados y disminuyese el tiempo empleado en el reparto. La resolución de este problema tenía una dificultad añadida, puesto que, se trataba un problema dinámico en el que los pedidos iban siendo añadidos conforme iban apareciendo, por lo tanto, tuvimos que documentarnos para poder abordar el problema. Al final, a la conclusión que llegamos fue que al tratarse de un problema dinámico no existe una solución óptima para todas las situaciones, pero sí pueden proponerse estrategias que en general ayudan en la resolución del problema.

Tras varias pruebas con los datos históricos que poseíamos, y tras varias simulaciones con estimaciones de tiempos, llegamos a la conclusión de que la mejor estrategia en este caso, era dejar atrás el sistema que tenían de una ruta norte y una ruta sur y considerar una única ruta. En ella, cada coche sale de su origen, recorre todos los centros y vuelve al origen por donde ha venido. Desde el primer día que se llevó a cabo esta estrategia, resultó ser mejor que la anterior, puesto que, aparte de que se consiguió el objetivo de completar más pedidos diarios, también se solventó un problema que estaba presente y del que no eran conscientes, y es que alguna de las piezas frágiles que transportaban, se estropeaban en el proceso de intercambio entre vehículos y llegaban rotas a su destino, lo cual ahora, al no hacer transbordo, no sucedía.

Esta tarea despertó nuestra curiosidad hacia los problemas de rutas de vehículos y en particular a los de recogida y entrega de mercancías, lo cuál motivó la realización de este trabajo.

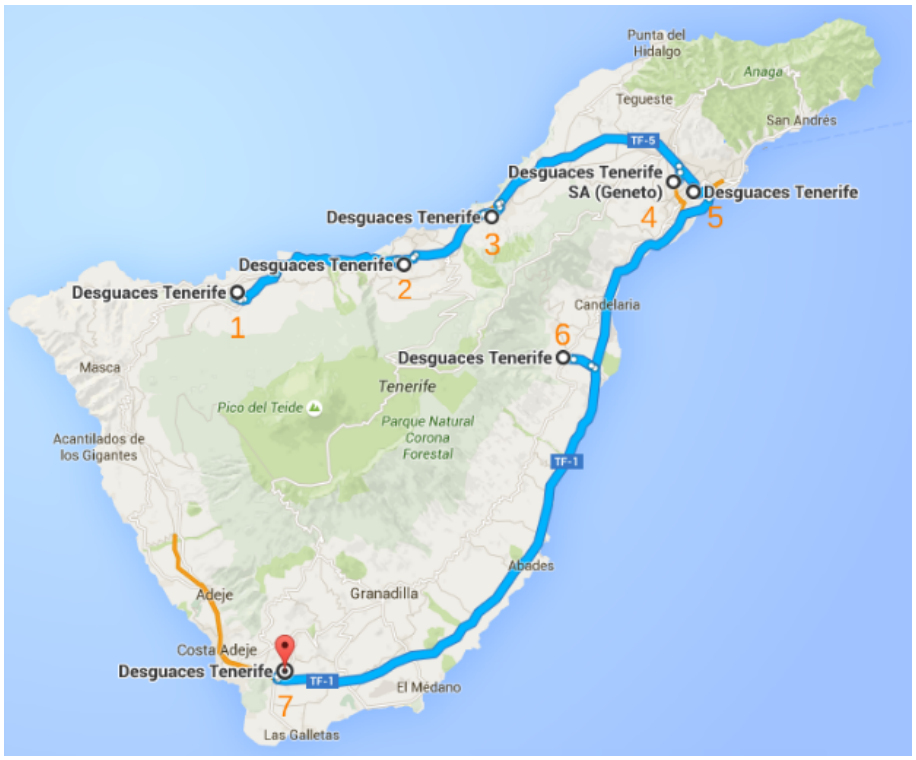


Figura 1.1: Localización de los centros de la empresa Desguaces Tenerife S.A.

Capítulo 2

Fundamentos teóricos

En este capítulo se presentan los problemas de rutas de vehículos (VRPs) que son generalizaciones del problema del viajante de comercio, conocido en inglés como *Traveling Salesman Problem* (TSP). Estos problemas, junto al TSP, están entre los problemas más importantes de la Optimización Combinatoria, debido a su dificultad y a su relevancia práctica. De hecho, estos problemas no sólo modelan los problemas de recogida y entrega de mercancías sino que, en general, aparecen como un ingrediente clave en muchos de los sistemas de transporte, como son los de recogida de residuos sólidos, limpieza de calles, rutas de autobuses, sistemas de recogida y entrega de pasajeros (*dial-a-ride*), rutas de las unidades de mantenimiento, transporte para personas con discapacidad. Otra área en la que problemas muy similares juegan un papel muy importante son las redes de telecomunicaciones modernas.

2.1. Problemas de rutas de vehículos

Las investigaciones teóricas y aplicaciones prácticas en el campo de la planificación de rutas para vehículos empezaron con el problema propuesto por Dantzig y Ramser (1959), en el que estudiaron la aplicación real en la distribución de gasolina para estaciones de carburante. Mediante el uso de un método basado en una formulación de un modelo de programación lineal, sus cálculos hechos a mano producen una solución casi óptima al problema con doce estaciones de servicio. En esa solución aparecen cuatro rutas. Los autores declararon: *"No practical applications of the method have been made as yet."*

Unos años después Clarke y Wright (1964) propusieron un algoritmo aproximado o heurístico (que es como se conoce a aquellos procedimientos que no nos garantizan que hayan alcanzado la solución óptima) efectivo que mejoraba el enfoque de Dantzig y Ramser. Después de estos dos artículos cientos de modelos y algoritmos han sido propuestos y a pesar del esfuerzo realizado en esta área en los últimos casi 60 años, incluso los problemas de rutas más básicos son todavía tremendamente difíciles de resolver óptimamente aun para tamaños relativamente pequeños y de escasa utilidad práctica. Por este motivo, el interés de la comunidad científica orientado a la resolución de estos problemas ha crecido de forma importante mediante el diseño de métodos sofisticados tanto exactos como aproximados.

2.2. Problemas de recogida y entrega de mercancías

Dentro de los problemas de rutas de vehículos encontramos los problemas de recogida y entrega de mercancías. Parragh, Doerner y Hartl (2008) clasifican estos problemas en dos grupos.

En el primero se encuentran los problemas que se ocupan de transportar la mercancía del depósito a los clientes y de los clientes al depósito. Estos problemas se denominan *Vehicle Routing Problems with Backhauls* (VRPB). En estos problemas los clientes están divididos en dos grupos los *linehauls* y los *backhauls*. Entre los VRPB se encuentran, *Vehicle Routing Problem with Clustered Backhauls* (VRPCB-todos los *linehauls* antes que los *backhauls*), *Vehicle Routing Problem with Mixed linehauls and Backhauls* (VRPMB-cualquier secuencia permitida entre *linehauls* y *backhauls*), *Vehicle Routing Problem with Divisible Delivery and Pickup* (VRPDDP- los clientes con servicio de entrega y recogida de mercancía pueden ser visitados dos veces), y *Vehicle Routing Problem with Simultaneous Delivery and Pickup* (VRPSDP- los clientes con servicio de entrega y recogida de mercancía son visitados exactamente una vez).

En el segundo grupo encontramos los problemas en los que la mercancía es transportada entre las localizaciones de recogida y entrega. Entre estos problemas encontramos: *the Pickup and Delivery Vehicle Routing Problem* (PDVRP- nodos de recogida y entrega no emparejados), *the classical Pickup and Delivery Problem* (PDP- nodos de recogida y entrega emparejados), y *the Dial-A-Ride Problem* (DARP-transporte de pasajeros entre puntos de recogida y entrega, teniendo en cuenta los inconvenientes de los usuarios). Las dos clases de problemas así como las subclases están representadas en la Figura 2.1.

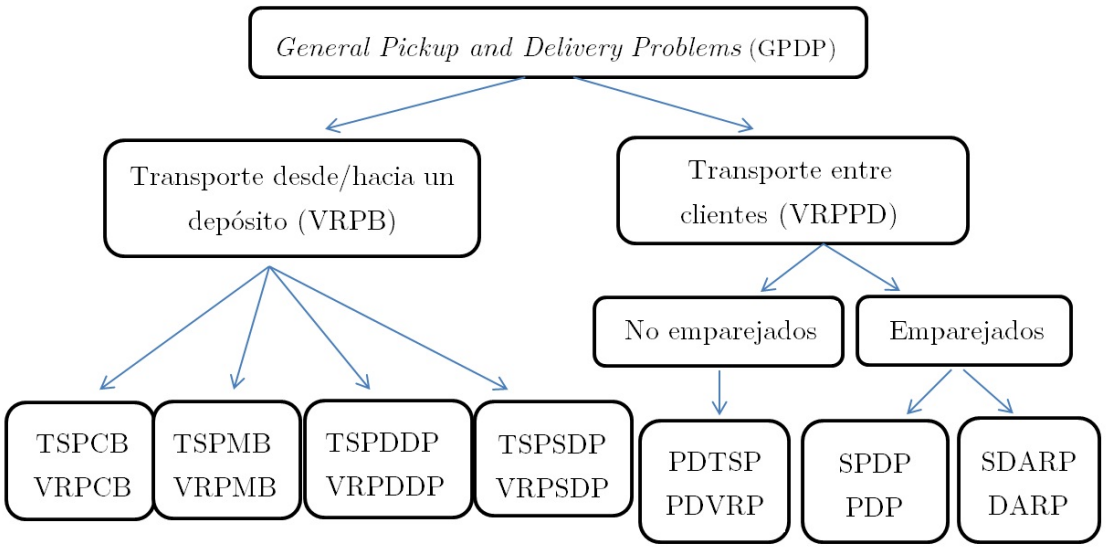
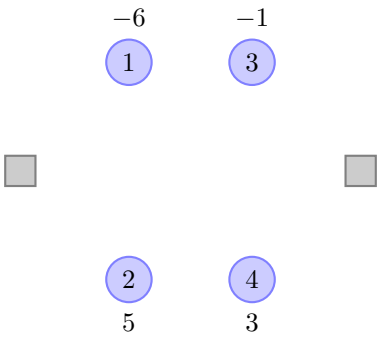
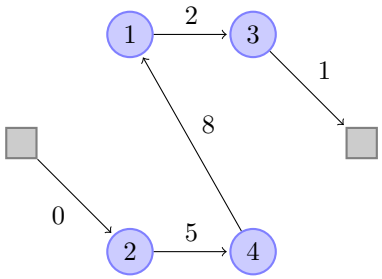


Figura 2.1: Clasificación de los problemas de recogida y entrega de mercancías por Parragh et al. (2008).

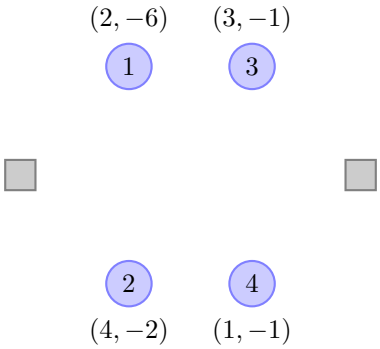


(a) Las etiquetas de los vértices positivas representan que el vértice entrega x unidades, las negativas $-y$ que el vértice demanda y unidades.

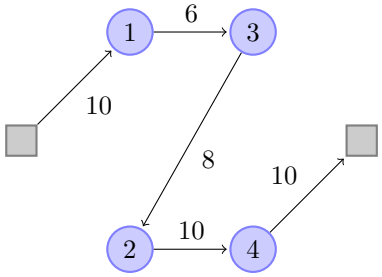


(b) Las etiquetas de los arcos indican la carga del vehículo.

Figura 2.2: Many-to-many problem.

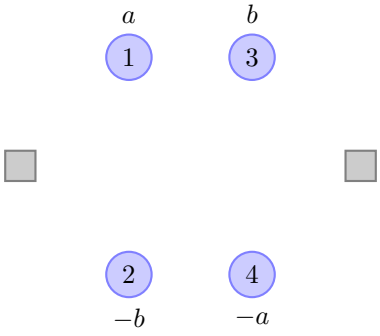


(a) Las etiquetas de los vértices $(x, -y)$ representan que el vértice entrega x unidades y demanda y unidades.

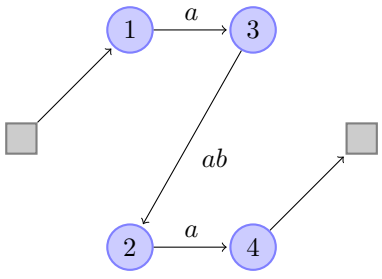


(b) Las etiquetas de los arcos indican la carga del vehículo.

Figura 2.3: One-to-many-to-one problem.



(a) Las etiquetas z de los vértices representan el origen de la mercancía, las etiquetas $-z$ representa el destino de la mercancía z.



(b) Las etiquetas de los arcos indican las mercancías que lleva el vehículo.

Figura 2.4: One-to-one problem.

Para describir las características de los diferentes PDPs, podemos utilizar el esquema [Estructura|Visitas|Vehículos] propuesto en el artículo de Berbeglia, Cordeau, Gribkovskaia y Laporte (2007).

- El primer campo, llamado estructura, especifica el número de orígenes y destinos de las mercancías.
  - En los problemas many-to-many (M-M), cualquier vértice puede ser tanto origen como destino de cualquier mercancía (Figura 2.2).
  - En los problemas one-to-many-to-one (1-M-1), las mercancías que se encuentran inicialmente en el deposito son entregadas a los clientes, y las mercancías disponibles en los clientes son enviadas al depósito. Los clientes de recogida o entrega no son necesariamente disjuntos y normalmente coinciden, como puede ser el caso de distribución de bebidas y recogida de las botellas vacías (Figura 2.3).
  - Finalmente, en los problemas one-to-one (1-1) cada mercancía tiene un origen y un destino fijado (Figura 2.4).
- El segundo campo proporciona información del modo en que la recogida o entrega de mercancías ha sido llevada a cabo en cada cliente.
  - Usamos PD para indicar que cada cliente se visita exactamente una vez para cada operación combinada de recogida y entrega.
  - P-D se usa si la recogida y entrega puede ser realizada de forma conjunta o por separado.
  - Escribimos P/D en el caso en el que el cliente solamente puede entregar o recoger mercancía y no ambas cosas a la vez.

  Además, se usa la letra T cuando alguno de los vértices puede ser usado como punto de transbordo.
- El campo Vehículos indica el número de vehículos utilizados en la solución.

Otro aspecto importante a tener en cuenta en los PDPs es la disponibilidad de la información. Si toda la información es determinista y conocida a priori, se trata de un problema estático, si por el contrario, la información se revela gradualmente con el paso del tiempo, es un problema dinámico.

2.3. Problemas de recogida y entrega de mercancías estáticos

Siguiendo el esquema anterior propuesto por Berbeglia et al. (2007) podemos realizar la siguiente clasificación de los PDPs estáticos.

- *Problemas Many-to-Many*

  1. *The Swapping Problem (SP)* [M-M|P-D|1]

     Este problema fue introducido por Anily y Hassin (1992). Es un problema en el que se transportan n-mercancías. Trata de realizar un intercambio de objetos entre los vértices de $V\backslash\{0\}$ con un solo vehículo (el vértice 0 se considera el depósito). El SP puede ser *preemptive* o *non-preemptive*. En el caso *preemptive* donde $T = V$ los objetos se pueden dejar temporalmente en cualquier vértice. Este caso se denota por [M -M|P-D-T|1]. En el caso *non-preemptive* no se permiten los transbordos, es decir, $T = \emptyset$. La versión del SP en la que $T \neq \emptyset$ y $T \neq V$ es conocida como *Mixed Swapping Problem*.

  2. *The One-Commodity Pickup and Delivery Traveling Salesman Problem* [M-M|P/D|1]

     En el problema *One-Commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP) se considera un único tipo de mercancía. El conjunto de clientes está dividido entre clientes de entrega y clientes de recogida y son atendidos por un único vehículo localizado inicialmente en el depósito. Este problema fue introducido por Hernández-Pérez y Salazar-González (2004).

  3. *The Q-Delivery Traveling Salesman Problem* [M-M|P/D|1]

     Este problema fue planteado por Chalasani y Motwani (1999). Consiste en recoger y entregar objetos individuales desde los vértices fuente a los vértices sumidero.

- *Problemas One-to-Many-to-One*

  1. *The 1-M-1-PDP with Combined Demands* [1-M-1|P-D|-]

     En los problemas con *combined demands* al menos uno de los clientes demanda servicio de recogida y entrega de mercancías. *The Single Vehicle Hamiltonian 1-M-1-PDP with Combined Demands* normalmente se denomina *the Traveling Salesman Problem with Simultaneous Pickups and Deliveries* (TSPSPD). Corresponde al caso [1-M-1|PD|1]. *The Multi-Vehicle Hamiltonian 1-M-1-PDP with Combined Demands* se denomina *the Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD) y corresponde al caso [1-M-1|PD|m].

  2. *The 1-M-1-PDP with Single Demands* [1-M-1|P/D|-]

     En los problemas con *single demands* cada cliente requiere un único servicio, es decir, servicio de recogida o servicio de entrega de mercancía.

     - *The Single Vehicle 1-M-1-PDP with Single Demands and Backhauls* [1-M-1|P/D|1]
       Este problema normalmente se denomina *the Traveling Salesman Problem with Backhauls* (TSPB). Es un caso especial del *Clustered Traveling Salesman Problem* en el cual $V\backslash\{0\}$ está particionado en p grupos en lugar de en dos (Chisman 1975).

- *The Single Vehicle 1-M-1-PDP with Single Demands and Mixed Solutions* [1-M-1|P/D|1]
  Comúnmente denominado *Traveling Salesman Problem with Pickups and Deliveries* (TSPPD)(Mosheiov 1994). El TSPPD es $\mathcal{NP}$-duro y está íntimamente relacionado con el TSP.
- *The Multi-Vehicle 1-M-1-PDP with Single Demands and Backhauls* [1-M-1|P/D|m]
  También denominado *Vehicle Routing Problem with Backhauls*(VRPB). Es un problema $\mathcal{NP}$-duro e incluye al *Capacitated Vehicle Routing Problem* (CVRP) como un caso especial.
- *The Multi-Vehicle 1-M-1-PDP with Single Demands and Mixed Solutions* [1-M-1|P/D|m]
  Es un caso especial del *Multi-Vehicle Hamiltonian 1-M-1-PDP with Combined Demands.*

- *Problemas One-to-One*

  En los problemas one-to-one cada mercancía tiene exactamente una localización de recogida y una de entrega.

  1. *Stacker Crane Problem* (SCP) [1-1|P/D|1].
     En este problema un vehículo de capacidad 1 transporta un único objeto desde su localización de origen a su destino cada vez. El SCP es un caso especial del *Swapping Problem* (SP). Es un problema $\mathcal{NP}$-duro aunque hay disponibles algoritmos polinomiales para ciertos tipos de gráficos.
  2. *Vehicle Routing Problem with Pickups and Deliveries* (VRPPD) [1-1|P/D|-]
     Se aplica en situaciones en las que lo que se desea transportar son objetos. Consiste en planificar las rutas de una flota de vehículos con el fin de satisfacer un conjunto de pedidos de ciertos clientes. En los pedidos se especifica el tamaño de la carga a transportar, así como los lugares de recogida y entrega. Cada pedido debe ser atendido por un vehículo y el lugar de recogida debe ser visitado antes del lugar de entrega. El VRPPD es $\mathcal{NP}$-duro ya que generaliza al VRP. Una variante de este problema es el VRPPDTW en el que si un vehículo llega a una localización antes de empezar su ventana de tiempo debe esperar hasta la hora de empezar su servicio.
  3. *Dial-a-Ride Problem* (DARP) [1-1 |P/D|-]
     Este problema es un caso especial del VRPPD en el cual ciertos usuarios solicitan ser transportados desde un origen a un destino. El DARP puede tener ciertas restricciones como ventanas de tiempo, tiempos máximos de viaje, y otras restricciones relacionadas con la calidad del servicio (Cordeau y Laporte (2003a), Cordeau y Laporte (2003b), Cordeau (2006)). El objetivo es minimizar una combinación ponderada del tiempo total de servicio y la insatisfacción de los usuarios. La principal aplicación del DARP son los servicios de transporte

de puerta a puerta que se ofrecen en muchas ciudades para las personas mayores y discapacitados.

Uno de los primeros estudios del DARP con un único vehículo [1-1 |P/D |1], fue llevado a cabo por Psaraftis (1980) que examinó el caso de petición inmediata, en el que los usuarios desean ser servidos tan pronto como sea posible. En la versión con varios vehículos se debe realizar una asignación de los pedidos a los vehículos así como la planificación y diseño de cada ruta. Como consecuencia, el espacio de soluciones es considerablemente mayor que en el del caso de un único vehículo y, por tanto, el problema es mucho más difícil de resolver.

4. *Vehicle Routing Problem with Pickups, Deliveries and Transshipments* (VRPPDT) [1-1 |P/D-T|-]

   VRPPD en el que los objetos pueden ser transportados desde su origen a su destino por diferentes vehículos. Esto se consigue permitiendo que los vehículos dejen las mercancías temporalmente en alguna localización de transbordo (*transshipment point*) para posteriormente volver a ser cogida por otro vehículo. Debido a la existencia de estas localizaciones de transbordo, el número de soluciones para el VRPPDT es mucho mayor que en el problema VRPPD estándar.

A continuación se presenta un esquema que recoge la clasificación anterior.

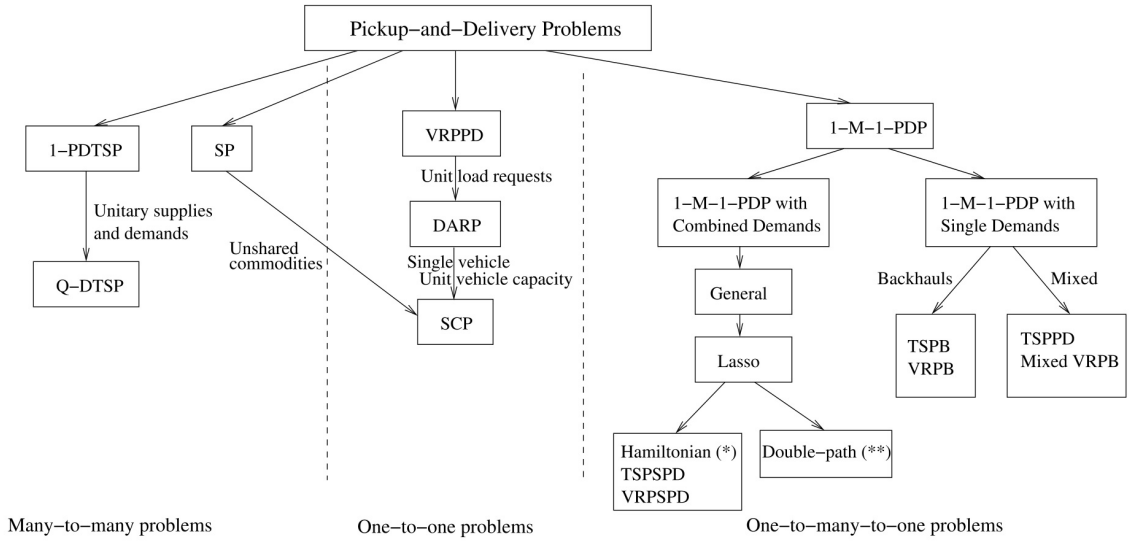


Figura 2.5: Clasificación de los problemas de recogida y entrega de mercancías por Berbeglia et al. (2007).

∗ *The single vehicle Hamiltonian 1-M-1-PDP with combined demands* es equivalente a *the single vehicle 1-M-1-PDP with single demands and mixed solutions.* ∗∗ *The 1-M-1-PDP with combined demands and double-path solution* es equivalente a *the 1-M-1-PDP with single demands and backhauls.*

2.4. Problemas de recogida y entrega de mercancías dinámicos

En la última década, ha habido una creciente dedicación a la investigación en problemas de rutas de vehículos dinámicos. En esta sección nos centramos en una variante de estos problemas, los denominados problemas de recogida y entrega dinámicos, en que los objetos o las personas tienen que ser recogidos y entregados en tiempo real.

Un problema de rutas de vehículos se dice que es estático cuando todos los datos de entrada del problema son conocidos antes del momento de planificación de las rutas. En un problema de rutas de vehículos dinámico, algunos de los datos de entrada se revelan o actualizan durante el período de tiempo en el que las rutas se están llevando a cabo. En un problema PDP dinámico, los datos de entrada que se revelan con el tiempo son generalmente los pedidos de los usuarios. En contraste con lo que sucede en un problema estático, el horizonte de planificación de un problema dinámico puede ser no acotado. Por lo tanto, una solución a un problema dinámico no puede ser una solución estática exacta, sino más bien una estrategia que, utilizando la información revelada, especifica qué acciones se debe realizar a medida que va pasando el tiempo.

La mayoría de los estudios sobre los problemas de recogida y entrega de mercancías se han centrado en el caso estático y poco se ha trabajado en la parte dinámica de los PDPs. Un ejemplo de un PDP dinámico aparece en el transporte de personas discapacitadas y de ancianos en las zonas urbanas. El aspecto dinámico de este problema, el *Dynamic Dial-a-Ride Problem* (*Dynamic DARP*), proviene del hecho de que las solicitudes de transporte a veces se reciben el mismo día que necesitan ser realizadas. Otro ejemplo aparece en las empresas de mensajería con los servicios de transporte de correspondencia que necesita ser entregada el mismo día que es enviada, o en empresas como Desguaces Tenerife S.A., que necesita enviar repuestos de un centro a otro el mismo día que son solicitados. Estos son algunos ejemplos de PDP dinámicos (*one-to-one*).

En lo que sigue, nos centraremos en el clásico *One-to-one Static Pickup and Delivery Problem*. Problema que ha sido estudiado durante más de 30 años y que ha aparecido en diversos contextos como la logística, servicios ambulatorios y robótica.

Capítulo 3

Un problema de recogida y entrega de mercancías estático

En este capítulo se proporciona una descripción del problema. También se presenta una formulación del problema para el caso estático aplicado a Desguaces Tenerife S.A., así como una sección con los resultados computacionales y finalmente un análisis de los mismos.

3.1. Descripción del problema

En el problema general se deben diseñar un conjunto de m rutas, que empiezan y terminan en el depósito de cada uno de los vehículos, para satisfacer las demandas de transporte. Una flota de m vehículos está disponible para operar estas rutas. La solución se desea óptima, esto es, de mínimo coste total. Para cada pedido se especifica su origen, su destino y la ventana de tiempo. Cada pedido debe ser transportado por un único vehículo desde su origen a su destino, sin transbordo. El nodo de origen (i) es visitado antes que el de destino $(n + i)$ (precedencia).

Listamos las características del problema referido a la empresa Desguaces Tenerife S.A:

- Tenemos 7 centros distribuidos por la isla de Tenerife, dos vehículos para realizar los pedidos y dos depósitos donde se encuentran los vehículos, Icod de los Vinos y Arona correspondiendo a las localizaciones 1 y 7 respectivamente.
- Se desea trasportar los pedidos realizados en un cierto día desde su localización de origen a su destino.
- Las distancias y los tiempos de desplazamiento son conocidos.
- Hay que respetar unas ventanas de tiempo según el pedido, es decir, hay que realizarlo dentro de un horario predeterminado.
- Las visitas solo pueden realizarse durante el horario de trabajo de Desguaces Tenerife S.A.

- Aunque es lógico pensar que dependiendo del tipo de localización, se necesitará un tiempo de proceso diferente, en nuestro modelo consideraremos que en todos los centros se tiene un tiempo de proceso de 10 minutos, dedicado a la carga y descarga de la mercancía.

3.2. Formulación del problema para el caso estático

Se considera el grafo dirigido $G = (N, A)$ donde N es el conjunto de nodos y A el conjunto de arcos que viene definido como $A = \{(i, j) : i, j \in N\}$. El conjunto de nodos está formado por $N{=}T \cup R \cup D$, donde $R = \{1, \ldots, n\}$ es el conjunto de nodos donde se recogen mercancías, $D = \{n + 1, \ldots, 2n\}$ el conjunto de nodos donde se entregan mercancías y $T = \{\tau_1, \ldots, \tau_m\}$ contiene al conjunto de depósitos, lugares donde los vehículos comienzan y terminan su ruta.

Denotamos al conjunto de vehículos como $V = \{1, \ldots, m\}$ con $\tau_k =$ depósito del vehículo k, $k \in V$.

Sea $L = \{1, \ldots, l\}$ el conjunto de localizaciones. En cada localización $i \in L$ existe un tiempo de proceso p_i. Para cada nodo $i \in N$ tenemos su localización $l(i) \in L$ y la ventana de tiempo $[a_i, b_i]$, donde a_i representa el tiempo a partir del cual podemos empezar a servir al nodo i y b_i el tiempo límite.

Para cada $i, j \in L$ conocemos la distancia entre localizaciones d_{ij} y el tiempo de viaje t_{ij}. Notar que cada pedido se representa por los nodos i y $n + i$, con $i \in R$ y $n + i \in D$. Los nodos pueden coincidir en las mismas localizaciones ya que pueden corresponder a localizaciones de recogida y entrega.

Para formular el problema introducimos las variables:

$$x_{ij}^k = \begin{cases} 1, & \text{si el vehículo } k \text{ va del nodo } i \text{ al nodo } j, \\ 0, & \text{en otro caso} \end{cases} \qquad \forall i, j \in N,\ k \in V$$

$$y_i^k = \begin{cases} 1, & \text{si el nodo } i \text{ es visitado por el vehículo } k, \\ 0, & \text{en otro caso} \end{cases} \qquad \forall i \in N,\ k \in V$$

$s_i^k =$ tiempo en el cual el vehículo k empieza el servicio en el nodo i, $\qquad \forall i \in N,\ k \in V$

Una vez definidos todos los parámetros y variables ya estamos en condiciones de formular el problema. El objetivo del problema es minimizar la distancia total recorrida por los vehículos además del tiempo total de espera.

$$\text{mín} \sum_{\substack{i,\, j\in N \\ j\neq i}} \sum_{k\in V} d_{l(i),l(j)} x_{ij}^k + \alpha \sum_{i\in R} \sum_{k\in V} (s_{n+i}^k - a_i) \tag{3.1}$$

sujeto a:

$$\sum_{k\in V} y_i^k = 1 \qquad \forall i \in N \tag{3.2}$$

$$\sum_{\substack{j\in N \\ j\neq i}} x_{ij}^k = y_i^k \qquad \forall i \in N, \forall k \in V \tag{3.3}$$

$$\sum_{\substack{j\in N \\ j\neq i}} x_{ji}^k = y_i^k \qquad \forall i \in N, \forall k \in V \tag{3.4}$$

$$y_i^k = y_{n+i}^k \qquad \forall i \in R, \forall k \in V \tag{3.5}$$

$$y_{\tau_k}^k = 1 \qquad \forall k \in V \tag{3.6}$$

$$s_i^k + (p_{l(i)} + t_{l(i),l(j)}) x_{ij}^k \leq s_j^k + M(1 - x_{ij}^k) \qquad \forall i, j \in N, j > 0, \forall k \in V \tag{3.7}$$

$$s_i^k + (p_{l(i)} + t_{l(i),l(n+i)}) \leq s_{n+i}^k \qquad \forall i \in R, \forall k \in V \tag{3.8}$$

$$a_i \leq s_i^k \leq b_i \qquad \forall i \in N, \forall k \in V \tag{3.9}$$

$$x_{\tau_k,n+i}^k = x_{i,\tau_k}^k = 0 \qquad \forall i \in R, \forall k \in V \tag{3.10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i, j \in N, \forall k \in V \tag{3.11}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in N, \forall k \in V \tag{3.12}$$

$$s_i^k \geq 0 \qquad \forall i \in N, \forall k \in V \tag{3.13}$$

En esta formulación la restricción (3.2) indica que cada nodo es visitado a lo sumo una vez por un vehículo, (3.3) y (3.4) que si un nodo es visitado por un vehículo el número de arcos que entran y el número de arcos que salen de él es uno, sin embargo si el nodo no se visita no hay arcos entrantes ni salientes. La restricción (3.5) expresa que si un vehículo visita un nodo en el que se recoge mercancía, éste también visita el correspondiente nodo de entrega de mercancía. La restricción (3.6) indica que cada vehículos regresa a su depósito. Con la restricción (3.7) se controla el tiempo de servicio entre nodos (M es una cantidad suficientemente grande para cada arco, tomamos $M = 600$ ya que es el tiempo máximo que pertenece abierta la empresa) y (3.8) obliga a que en cada pedido se visite antes el nodo de recogida de mercancía que el de entrega. Con la restricción (3.9) cada vértice es visitado dentro de su ventana de tiempo, es decir, después del tiempo a_i y antes de b_i. Con (3.10) hacemos que no existan arcos de los depósitos hacia nodos de entrega de mercancía ni de los nodos de recogida de mercancía hacia los depósitos. Finalmente, (3.11) y (3.12) representan las variables binarias y (3.13) que la variable tiempo en que empieza el servicio en cada nodo sea mayor o igual que cero.

3.3. Resultados computacionales

A continuación presentamos los resultados computacionales obtenidos al ejecutar el modelo anterior en Xpress (ver Apéndice A, Sección A.1). Se ha ejecutado en un ordenador portátil intel i5, 2.4 GHz, y 4 GB de memoria RAM. También se ejecutó con Gusek pero el tiempo computacional era muchísimo mayor.

Se considera que conocemos los pedidos a priori. Los datos y parámetros del problema han sido leídos desde ficheros externos ("fichero.dat").

Las columnas de la Tabla 3.1 representan:

- *Número de pedidos*: El número total de pedidos considerados en la ejecución.
- α: Cantidad por la cual ha sido pesado el tiempo de espera en la función objetivo.
- *Coste total*: Suma ponderada de la distancia total y el tiempo de espera.
- *Distancia total*: Distancia total recorrida por los vehículos. Medida en kilómetros.
- *Tiempo total*: Tiempo total que están los vehículos realizando las rutas. Medido en minutos.
- *Tiempo de espera*: Tiempo de espera total de ambos vehículos. Medido en minutos.
- *Gap*: El $Gap = \dfrac{UB - LB}{LB} * 100$. Donde UB es la mejor solución encontrada y LB la cota inferior que devuelve el resolutor.
- *Tiempo de ejecución*: Tiempo computacional de cada ejecución. Medido en segundos. Se ha impuesto un tiempo límite de 1000 segundos.

En la tabla se observa que a medida que aumenta el número de pedidos, el modelo tarda más tiempo en resolver el problema. Xpress es capaz de resolverlo de forma exacta con 6 pedidos para todos los valores de α. Para 7 pedidos lo resuelve de forma óptima para los valores de $\alpha = 0.0$, 0.1 y 0.2. Pero a partir de 8 pedidos ya van apareciendo *Gaps* considerablemente grandes (menos para 8 pedidos con $\alpha = 0.0$ que también lo resuelve óptimamente).

Analizando los resultados de la Tabla 3.1 vemos que existen soluciones dominadas (eficiencia de Pareto)(ver Apéndice B). Si por ejemplo nos fijamos en el caso para 15 pedidos, la solución obtenida con $\alpha = 0.1$ domina a las obtenidas en los casos $\alpha = 0.0$, 0.2, 0.5 y 1.0 puesto que, tanto la distancia total como el tiempo de espera es menor. También en el caso para 10 pedidos tenemos soluciones dominadas. La solución con $\alpha = 0.5$ domina la obtenida con $\alpha = 0.2$ puesto que aunque se tiene el mismo valor de la distancia total recorrida, el valor del tiempo de espera sí es menor. También domina a la obtenida con $\alpha = 1.0$. Llamamos al resto de soluciones conjunto "no dominado" ya que ninguna de las soluciones está dominada.

Número de pedidos	α	Coste total	Distancia total	Tiempo total	Tiempo de espera	*Gap* %	Tiempo de ejecución
6	0.0	298.9	298.9	291	3977	0.0	1.6
	0.1	407.9	303.8	302	1041	0.0	3.9
	0.2	511.7	304.5	301	1036	0.0	6.2
	0.5	822.5	304.5	301	1036	0.0	66.3
	1.0	1340.5	304.5	301	1036	0.0	233.7
7	0.0	300.1	300.1	295	3366	0.0	18.9
	0.1	427.8	307.1	309	1207	0.0	85.2
	0.2	548.5	307.1	309	1207	0.0	252.2
	0.5	910.6	307.1	309	1207	17.3	1000.0
	1.0	1514.1	307.1	309	1207	19.9	1000.0
8	0.0	301.6	301.6	276	4595	0.0	863.3
	0.1	476.2	307.1	309	1691	19.1	1000.0
	0.2	647.0	306.4	309	1703	24.5	1000.0
	0.5	1110.1	350.6	352	1519	30.4	1000.0
	1.0	1928.3	347.3	344	1581	32.8	1000.0
10	0.0	301.6	301.6	276	4337	27.5	1000.0
	0.1	525.6	310.5	309	2151	37.3	1000.0
	0.2	747.4	351.0	360	1982	37.4	1000.0
	0.5	1336.0	351.0	360	1970	37.9	1000.0
	1.0	2354.0	352.0	358	2002	40.4	1000.0
15	0.0	390.3	390.3	366	4407	67.5	1000.0
	0.1	737.2	370.4	354	3668	58.2	1000.0
	0.2	1360.3	492.7	422	4338	65.6	1000.0
	0.5	3336.4	639.4	607	5394	71.7	1000.0
	1.0	4356.1	555.1	503	3801	60.5	1000.0

Tabla 3.1: Resultados computacionales para el modelo PDP estático

Como se observa en la Tabla 3.1 este problema es difícil de resolver y solamente somos capaces de encontrar una solución óptima hasta el caso con a lo sumo 8 pedidos ($\alpha = 0$). Esto nos hace pensar que resolver el problema para los casi 50 pedidos diarios que tiene la empresa Desguaces Tenerife S.A es prácticamente imposible, por lo tanto, decidimos pensar en un problema en el que las rutas sean fijas y tengamos que decidir solamente el tiempo de espera en cada localización.

Capítulo 4

Un problema de recogida y entrega de mercancías con rutas fijas

En este capítulo presentamos formulaciones del problema de recogida y entrega de mercancías pero con las rutas de los vehículos fijas. Ahora el objetivo de nuestro problema es minimizar el tiempo de espera así como la cantidad de pedidos que quedan sin cumplir.

4.1. Descripción del problema

El problema consiste ahora en decidir los tiempos de salida de cada localización para que el tiempo de espera total y el número de pedidos sin cumplir sea mínimo. En este problema, al contrario que el considerado en el capítulo 3, el conjunto de rutas de los m vehículos vienen fijado de antemano. Siguen partiendo y terminando en sus respectivos depósitos. La solución se desea óptima, esto es, de mínimo coste total.

Las características del problema referido a la empresa Desguaces Tenerife S.A son las siguientes:

- Tenemos 7 centros distribuidos por la isla de Tenerife, dos vehículos para realizar los pedidos (conocen la ruta que van a seguir a lo largo del día) y dos depósitos donde se encuentran los vehículos, Icod de los Vinos y Arona correspondiendo a las localizaciones 1 y 7 respectivamente.

- Se desea trasportar los pedidos realizados en un cierto día desde su localización de origen a su destino.

- Normalmente al día suelen surgir unos 50 pedidos.

- Los tiempos de desplazamiento entre localizaciones son conocidos.

- Las visitas solo pueden realizarse durante el horario de trabajo de Desguaces Tenerife S.A., desde las 8:00 a las 18:00.
- Se conoce la hora a la que se realiza cada pedido.
- Consideraremos que en todos los centros se tiene un tiempo de proceso de 10 minutos, dedicado a la carga y descarga de la mercancía. También se tendrá en cuenta el descanso de los trabajadores para comer al medio día (50 minutos).
- Cada pedido debe ser transportado por un único vehículo desde su origen a su destino, sin transbordo.
- El nodo de origen (i) es visitado antes que el de destino $(n+i)$ (precedencia).

4.2. Formulación del problema para el caso lineal con rutas fijas y sentidos

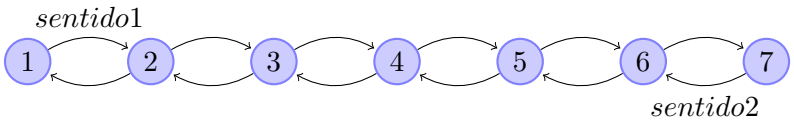


Figura 4.1: Grafo lineal

Los nodos del grafo anterior corresponden con los siete centros de Desguaces Tenerife S.A. (Ver figura 1.1).

Parámetros

Sea $L = \{1, \ldots, l\}$ el conjunto de localizaciones, $V = \{1, \ldots, k\}$ el conjunto de vehículos, $R = \{1, \ldots, n\}$ el conjunto de pedidos y $S = \{1, 2\}$ el conjunto de sentidos. Para cada pedido $i \in R$ se especifica:
$o(i)$ = origen del pedido i
$d(i)$ = destino del pedido i
$S(i)$ = sentido del pedido i
$a(i)$ = hora a la que se realiza el pedido i (en minutos a partir de las 8:00)

Se denota p_i al tiempo de proceso dedicado a la carga y descarga del vehículo en cada localización y t_{ij} al tiempo de viaje entre localizaciones. M es una cantidad suficientemente grande para cada arco (tomamos $M = 600$ puesto que es el tiempo que pertenece abierta la empresa) y α y β los pesos que se le dan en la función objetivo al tiempo de espera y a los pedidos no realizados respectivamente.

Variables

Para la formulación de este problema introducimos las siguientes variables:

$$y_r^{k,s} = \begin{cases} 1, & \text{si el pedido } r \text{ es cargado en el vehículo } k \text{ que va en el sentido } s \\ 0, & \text{en otro caso} \end{cases}$$

$e_i^{k,s}$ = hora (en minutos a partir de las 8:00) a la que el vehículo k llega al nodo i en sentido s

$l_i^{k,s}$ = hora (en minutos a partir de las 8:00) a la que el vehículo k sale del nodo i en sentido s

h_r = hora (en minutos a partir de las 8:00) a la que el pedido r es entregado en caso de que sea cargado

Modelo matemático

$$\text{mín}\, \alpha \sum_{r \in R} (h_r - a_r) + \beta \sum_{\substack{r \in R \\ s \in S(r)}} (1 - (y_r^{1,s} + y_r^{2,s})) \tag{4.1}$$

sujeto a:

$$h_r \geq e_{d(r)}^{k,s} - M(1 - y_r^{k,s}) \qquad \forall r \in R, \forall k \in V, \forall s \in S \tag{4.2}$$

$$h_r \geq a_r \qquad \forall r \in R \tag{4.3}$$

$$a_r \leq l_{o(r)}^{k,s} + M(1 - y_r^{k,s}) \qquad \forall r \in R, \forall k \in V, \forall s \in S | s = S(r) \tag{4.4}$$

$$l_i^{k,s} \geq e_i^{k,s} + p_i \qquad \forall i \in L, \forall k \in V, \forall s \in S \backslash \{^{i=7,k=1,s=2}_{i=1,k=2,s=1}\} \tag{4.5}$$

$$l_7^{1,2} \geq e_7^{1,1} + p_7 + 50 \tag{4.6}$$

$$l_1^{2,1} \geq e_1^{2,2} + p_1 + 50 \tag{4.7}$$

$$e_{i+1}^{k,1} = l_i^{k,1} + t_{i,i+1} \qquad \forall i \in 1, .., 6, \forall k \in V \tag{4.8}$$

$$e_{i-1}^{k,2} = l_i^{k,2} + t_{i,i-1} \qquad \forall i \in 2, .., 7, \forall k \in V \tag{4.9}$$

$$e_i^{k,s} \leq M \qquad \forall i \in L, \forall k \in V, \forall s \in S \tag{4.10}$$

$$y_r^{1,s} + y_r^{2,s} \leq 1 \qquad \forall r \in R, \forall s \in S \mid s = S(r) \tag{4.11}$$

$$y_r^{1,s} = y_r^{2,s} = 0 \qquad \forall r \in R, \forall s \in S \mid s \neq S(r) \tag{4.12}$$

$$l_i^{k,s} \geq 0 \qquad \forall i \in L, \forall k \in V, \forall s \in S \tag{4.13}$$

$$e_i^{k,s} \geq 0 \qquad \forall i \in L, \forall k \in V, \forall s \in S \tag{4.14}$$

$$h_r \geq 0 \qquad \forall r \in R \tag{4.15}$$

$$y_r^{k,s} \in \{0, 1\} \qquad \forall r \in R, \forall k \in V, \forall s \in S \tag{4.16}$$

En esta formulación la restricción (4.2) indica que la hora a la que cada pedido es

entregado en caso de que haya sido cargado es superior a la de llegada a la localización de destino del pedido. La restricción (4.3) indica que la hora a la que se entrega cada pedido es mayor que la hora a la que se ha realizado y (4.4) expresa que la hora a la que se sale de la localización de origen de un pedido es posterior a la hora a la que surge el mismo. La restricción (4.5) expresa que la hora a la que se sale de un centro es superior a la hora a la que se llega más el tiempo de proceso. Las restricciones (4.6) y (4.7) indican que para cada coche la hora de salida de los nodos donde paran para almorzar es superior a la hora de llegada más el tiempo de proceso más el receso para comer (50 minutos). Con las restricciones (4.8) y (4.9) se controla el tiempo de llegada de cada vehículo a los extremos del grafo. La restricción (4.10) obliga a llegar a cada localización antes del cierre de la empresa (M suficientemente grande por ejemplo $M = 600$). Con (4.11) se consigue que cada pedido sea cargado a lo sumo una vez. Con (4.12) hacemos que no se carguen pedidos en los vehículos que van en sentido opuesto al del pedido. Finalmente, (4.13) y (4.14) obligan a que los tiempos de llegada y salida de los vehículos en cada sentido a cada centro sean mayores o iguales que cero. La restricción (4.15) también expresa que la hora en que los pedidos son entregados si han sido cargados sea mayor o igual que cero y (4.16) representa las variables binarias.

4.3. Formulación del problema para el caso circular con rutas fijas y localizaciones duplicadas

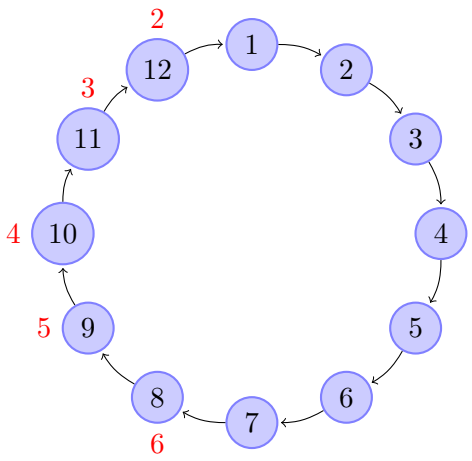


Figura 4.2: Grafo circular

En esta formulación lo que se pretende conseguir es un modelo equivalente al anterior pero en el que no se tenga en cuenta el sentido. Para ello lo que se hace es duplicar las localizaciones $\{2,\ldots,6\}$ (Figura 4.2). Con esto se consigue que los vehículos realicen la misma ruta que antes, es decir, partan de su depósito recorran todos los centros y vuelvan al depósito en sentido contrario.

Parámetros

Los parámetros de este modelo son prácticamente iguales a los del modelo lineal anterior. La única diferencia reside en que ya no tenemos en cuenta el sentido y que tenemos que definir una función que nos indique la localización del depósito de cada vehículo, $dep(k)$. Ahora el número de localizaciones es mayor.

Variables

$$y_r^k = \begin{cases} 1, & \text{si el pedido } r \text{ es cargado en el vehículo } k \\ 0, & \text{en otro caso} \end{cases}$$

e_i^k = hora (en minutos a partir de las 8:00) a la que el vehículo k llega al nodo i

l_i^k = hora (en minutos a partir de las 8:00) a la que el vehículo k sale del nodo i

h_r = hora (en minutos a partir de las 8:00) a la que el pedido r es entregado en caso de que sea cargado

Modelo matemático

$$\text{mín}\, \alpha \sum_{r \in R} (h_r - a_r) + \beta \sum_{r \in R} (1 - (y_r^1 + y_r^2)) \tag{4.17}$$

sujeto a:

$$h_r \geq e_{d(r)}^k - M(1 - y_r^k) \qquad \forall r \in R, \forall k \in V \tag{4.18}$$

$$a_r \leq l_{o(r)}^k + M(1 - y_r^k) \qquad \forall r \in R, \forall k \in V \tag{4.19}$$

$$h_r \geq a_r \qquad \forall r \in R \tag{4.20}$$

$$l_i^k \geq e_i^k + p_i \qquad \forall i \in L : i \neq dep(k), \forall k \in V \backslash \{^{i=7,k=1}_{i=1,k=2}\} \tag{4.21}$$

$$l_7^1 \geq e_7^1 + p_7 + 50 \tag{4.22}$$

$$l_1^2 \geq e_1^2 + p_1 + 50 \tag{4.23}$$

$$l_{dep(k)}^k \geq p_{dep(k)} \qquad \forall k \in V \tag{4.24}$$

$$e_{i+1}^k = l_i^k + t_{i,i+1} \qquad \forall i \in 1, .., 11, \forall k \in V \tag{4.25}$$

$$e_1^k = l_{12}^k + t_{12,1} \qquad \forall k \in V \tag{4.26}$$

$$e_i^k \leq M \qquad \forall i \in L, \forall k \in V \tag{4.27}$$

$$y_r^1 + y_r^2 \leq 1 \qquad \forall r \in R \tag{4.28}$$

$$l_i^k \geq 0 \qquad \forall i \in L, \forall k \in V \tag{4.29}$$

$$e_i^k \geq 0 \qquad \forall i \in L, \forall k \in V \tag{4.30}$$

$$h_r \geq 0 \qquad \forall r \in R \tag{4.31}$$

$$y_r^k \in \{0, 1\} \qquad \forall r \in R, \forall k \in V \tag{4.32}$$

Notar que las restricciones de este modelo son prácticamente iguales a las del modelo lineal. Las diferencias subyacen en que ahora no tenemos un superíndice para el sentido y que aparece la restricción (4.24) que indica que el tiempo de salida del depósito para cada vehículo es mayor que el tiempo de proceso en el mismo. Con (4.25) y (4.26) se controla el tiempo de llegada de los vehículos a todas las localizaciones.

4.4. Formulación del problema para el caso circular con rutas fijas, localizaciones duplicadas y sin depósitos prefijados

La idea de este modelo es decidir cuales serían las localizaciones de los depósitos de los vehículos para que se consiga realizar mayor número de pedidos y con un tiempo de espera menor.

Parámetros

En este modelo seguimos utilizando los mismos parámetros que en el modelo anterior, aunque con la diferencia de que desaparece la función depósito de los vehículos, $dep(k)$.

Variables

Debido a que deseamos dar libertad al modelo para poder decidir donde deberían estar localizados los depósitos de los vehículos para optimizar la función objetivo, introducimos la siguiente variable:

$$D_i^k = \begin{cases} 1, & \text{si el nodo } i \text{ es el depósito del vehículo } k \\ 0, & \text{en otro caso} \end{cases}$$

El resto de variables se definen como en la formulación de la sección 4.3.

Modelo matemático

$$\text{mín}\, \alpha \sum_{r \in R} (h_r - a_r) + \beta \sum_{r \in R} (1 - (y_r^1 + y_r^2)) \tag{4.33}$$

sujeto a:

$$h_r \geq e^k_{d(r)} - M(1 - y^k_r) \qquad \forall r \in R, \forall k \in V \tag{4.34}$$

$$a_r \leq l^k_{o(r)} + M(1 - y^k_r) \qquad \forall r \in R, \forall k \in V \tag{4.35}$$

$$h_r \geq a_r \qquad \forall r \in R \tag{4.36}$$

$$l^k_i \geq e^k_i + p_i + 50D^k_{i-6} - M(D^k_i) \qquad \forall i \in 7, .., 12, \forall k \in V \tag{4.37}$$

$$l^k_i \geq e^k_i + p_i + 50D^k_{i+6} - M(D^k_i) \qquad \forall i \in 1, .., 6, \forall k \in V \tag{4.38}$$

$$l^k_i \geq p_i \qquad \forall i \in L, \forall k \in V \tag{4.39}$$

$$\sum_{i \in L} D^k_i = 1 \qquad \forall k \in V \tag{4.40}$$

$$e^k_{i+1} = l^k_i + t_{i,i+1} \qquad \forall i \in 1, .., 11, \forall k \in V \tag{4.41}$$

$$e^k_1 = l^k_{12} + t_{12,1} \qquad \forall k \in V \tag{4.42}$$

$$e^k_i \leq M \qquad \forall i \in L, \forall k \in V \tag{4.43}$$

$$y^1_r + y^2_r \leq 1 \qquad \forall r \in R \tag{4.44}$$

$$l^k_i \geq 0 \qquad \forall i \in L, \forall k \in V \tag{4.45}$$

$$e^k_i \geq 0 \qquad \forall i \in L, \forall k \in V \tag{4.46}$$

$$h_r \geq 0 \qquad \forall r \in R \tag{4.47}$$

$$y^k_r \in \{0, 1\} \qquad \forall r \in R, \forall k \in V \tag{4.48}$$

$$D^k_i \in \{0, 1\} \qquad \forall i \in L, \forall k \in V \tag{4.49}$$

Las restricciones (4.34) – (4.36) y (4.41) – (4.48) son exactamente iguales a las del modelo matemático de la sección 4.3. Las diferencias aparecen en (4.37) – (4.39) que son las que permiten controlar el tiempo salida de las localizaciones dependiendo de si son localizaciones de paso, depósitos o localizaciones en las que se descansa. En las localizaciones de paso el tiempo de salida tiene que ser mayor al de llegada más el tiempo de proceso. En los depósitos el tiempo de salida es superior al de proceso de carga y descarga y en las localizaciones de descanso, el tiempo de salida es mayor o igual al de llegada más el tiempo de proceso además de 50 minutos de descanso. Con (4.40) se indica que cada vehículo tiene un único depósito y (4.49) representa las variables binarias depósito.

4.5. Resultados computacionales

Implementamos los tres modelos matemáticos en Xpress (Apéndice A). Se han ejecutado en un ordenador portátil intel i5, 2.4 GHz, y 4 GB de memoria RAM. Todos los experimentos tienen una limitación en su ejecución de 2000 segundos. Los datos y parámetros considerados en todos ellos son datos proporcionados por la empresa Desguaces Tenerife S.A. Los parámetros han sido leídos desde ficheros externos ("fichero.dat").

4.5.1. Modelo lineal con rutas fijas y sentidos

Los datos considerados han sido:

- 7 localizaciones correspondiendo a los 7 centros. Las localizaciones 1 y 7 son los depósitos de los vehículos.
- 2 sentidos, $s = 1$ a favor de las agujas del reloj, $s = 2$ en contra.
- 2 vehículos.
- 1118 pedidos correspondientes al periodo comprendido entre el 9 de octubre y el 11 de noviembre de 2015.

Se ejecuta con estos 1118 pedidos puesto que esto nos va a permitir decidir cuál es la mejor estrategia a seguir en cuanto a la planificación de los horarios de las rutas en media. Es decir, si hubiéramos aplicado los mismos horarios todos los días del 9 de octubre al 11 de noviembre sería la mejor solución.

Se han obtenido los siguientes resultados:

β	Valor óptimo	Pedidos sin cumplir	Tiempo de espera	Gap
1000	380269	218	162269	0.43
500	270715	225	158215	1.81
250	200278	490	77778	4.32

Tabla 4.1: Resultados computacionales para el modelo lineal con sentidos

β	Vehículo	Sentido	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5	Loc. 6	Loc. 7
1000	1	1	**10:39**	11:14	11:42	12:12	12:30	13:04	13:49
		2	18:08	17:32	17:05	16:34	16:12	15:32	14:45
	2	2	13:51	13:15	12:48	12:17	11:55	11:20	**10:30**
		1	15:00	15:35	16:03	16:33	16:51	17:25	18:10
500	1	1	**9:54**	10:29	10:57	11:28	11:47	12:21	13:06
		2	18:08	17:32	17:05	16:34	16:12	15:28	13:56
	2	2	13:41	13:05	12:38	12:07	11:45	11:10	**10:23**
		1	14:31	15:06	15:34	16:22	16:40	17:14	17:59
250	1	1	**8:43**	9:18	9:47	10:18	10:36	11:10	11:55
		2	18:08	17:32	17:05	16:34	16:12	13:32	12:45
	2	2	13:20	12:44	12:17	11:46	11:19	9:58	**9:11**
		1	14:10	14:45	15:13	15:43	16:01	16:35	17:20

Tabla 4.2: Solución para el modelo lineal con sentidos ($\alpha = 1$)

En la Tabla 4.2 aparecen representados los horarios de salida de los vehículos de las diferentes localizaciones en ambos sentidos que hacen óptima la solución para diferentes valores de β. En todos los casos hemos considerado $\alpha = 1$. A modo ilustrativo hemos representado en un mapa las soluciones obtenidas para cada vehículo con $\beta = 1000$.

(a) Vehículo 1 (b) Vehículo 2

Figura 4.3: Solución para el modelo lineal con 1118 pedidos y $\beta = 1000$

Notar que la mejor estrategia a seguir cuando $\beta = 1000$ es comenzar a operar las rutas dos horas y media después de la apertura de la empresa. Sin embargo, cuando no se pesa tanto los pedidos no realizados sino el tiempo de espera, las rutas comienzan antes.

4.5.2. Modelo circular con rutas fijas y localizaciones duplicadas

Los datos considerados en este modelo son:

- 12 localizaciones correspondiendo de la 1 a la 7 con los 7 centros y de la 8 a la 12 con la duplicación de las localizaciones 6 a 2 respectivamente. Las localizaciones 1 y 7 son los depósitos de los vehículos.
- 2 vehículos.
- 1118 pedidos correspondientes al periodo comprendido entre el 9 de octubre y el 11 de noviembre de 2015.

Como hemos comentado anteriormente, este modelo resuelve exactamente el mismo problema que el modelo anterior. En las tablas 4.3 y 4.4 se presentan los resultados obtenidos. Este modelo resuelve el problema para $\beta = 1000$ de forma exacta en 1099.6 segundos mientras que su equivalente en el modelo anterior, aún permitiendo un tiempo de ejecución de 3000 segundos no alcanza la solución exacta, de hecho, el *gap* apenas mejora con respecto a la solución obtenida a los 2000 segundos.

β	Valor óptimo	Pedidos sin cumplir	Tiempo de espera	*Gap*
1000	380198	218	162198	0
500	270672	225	158172	1.59
250	200167	477	80917	4.42

Tabla 4.3: Resultados computacionales para el modelo circular con localizaciones duplicadas

β	Veh.	1	2	3	4	5	6	7	8	9	10	11	12
1000	1	**10:39**	11:14	11:43	12:13	12:31	13:05	14:45	15:32	16:12	16:34	17:05	17:32
	2	15:00	15:35	16:03	16:33	16:51	17:25	**10:30**	11:20	11:55	12:17	12:48	13:15
500	1	**9:47**	10:22	10:50	11:20	11:38	12:12	13:54	15:28	16:12	16:34	17:05	17:32
	2	14:31	15:06	15:34	16:22	16:40	17:14	**10:23**	11:10	11:45	12:07	12:38	13:05
250	1	**8:46**	9:21	9:49	10:22	10:40	11:14	12:49	13:36	16:03	16:26	16:57	17:26
	2	14:10	14:45	15:13	15:43	16:01	16:35	**9:44**	10:31	11:21	11:46	12:17	12:44

Tabla 4.4: Solución para el modelo circular con localizaciones duplicadas ($\alpha = 1$)

Volvemos a representar en un mapa las soluciones obtenidas para cada vehículo con $\beta = 1000$. Notar que la solución es prácticamente igual a la obtenida en el modelo lineal (difiere en algunas localizaciones en un minuto) pero esta vez sí sabemos que es la solución óptima.

(a) Vehículo 1 (b) Vehículo 2

Figura 4.4: Solución para el modelo circular con 1118 pedidos y $\beta = 1000$

4.5.3. Modelo circular con rutas fijas, localizaciones duplicadas y sin depósitos prefijados

En este caso hemos considerado los datos:

- 12 localizaciones correspondiendo de la 1 a la 7 con los 7 centros y de la 8 a la 12 con la duplicación de las localizaciones 6 a 2 respectivamente.
- 2 vehículos.
- 497 pedidos correspondientes al periodo comprendido entre el 9 octubre y el 23 de octubre de 2015.

Se ejecuta con 497 pedidos porque al intentar resolverlo con los 1118 considerados anteriormente, cuando finaliza el proceso queda un *gap* muy elevado lo que nos hace tener que considerar una menor cantidad de pedidos.

En las Tablas 4.5 y 4.6 aparecen los resultados obtenidos.

β	Valor óptimo	Pedidos sin cumplir	Tiempo de espera	*Gap*	T. de ejecución
1000	104496	65	39496	0	1258
500	72694	66	39694	13.43	2000
250	54121	101	28871	37.06	2000

Tabla 4.5: Resultados computacionales para el modelo circular con localizaciones duplicadas y sin depósitos prefijados

β	Veh.	Dep.	1	2	3	4	5	6	7	8	9	10	11	12
1000	1	8	13:20	14:45	15:13	15:43	16:01	16:35	17:23	**9:37**	11:18	11:46	12:17	12:44
	2	11	9:32	10:07	10:35	11:05	12:14	12:48	13:33	16:35	17:10	17:38	**8:10**	8:37
500	1	8	13:20	14:45	15:13	15:43	16:01	16:35	17:23	**9:37**	11:18	11:46	12:17	12:44
	2	11	9:41	10:16	10:44	11:14	12:27	13:01	13:46	16:35	17:10	17:38	**8:10**	8:37
250	1	11	9:13	9:48	10:16	10:46	11:54	12:28	13:13	15:43	16:42	17:19	**8:10**	8:37
	2	8	13:20	14:45	15:13	15:43	16:01	16:35	17:23	**9:37**	11:18	11:46	12:17	12:44

Tabla 4.6: Solución para el modelo circular con localizaciones duplicadas y sin depósitos prefijados (497 pedidos y $\alpha = 1$)

En la Tabla 4.6 se indican los horarios de salida de los vehículos de las diferentes localizaciones que son solución del problema para diferentes valores de β. En todos los casos se considera α=1. En la Figura 4.5 está representada la solución obtenida cuando $\beta = 1000$. Notar que en la solución para todos los valores de β los depósitos están localizados en las localizaciones 8 y 11 (Güímar y La Victoria).

(a) Vehículo 1 (b) Vehículo 2

Figura 4.5: Solución para el modelo circular sin depósitos prefijados con 497 pedidos y $\beta = 1000$

Procedemos ahora a realizar una tabla en la que se comparan los resultados computacionales anteriores con los obtenidos al ejecutar el modelo fijando los depósitos a las localizaciones 1 y 7 (Icod de los Vinos y Arona).

	Depósitos fijos			Depósitos no fijos		
β	Valor ópt.	PSC	WT	Valor ópt.	PSC	WT
1000	161698	95	66698	104496	65	39496
500	114198	95	66698	72694	66	39694
250	84619	191	36869	54121	101	28871

Tabla 4.7: Tabla comparativa de resultados computacionales para el modelo circular con localizaciones duplicadas (497 pedidos y α=1)

Las columnas de la Tabla 4.7 representan:

- β: Cantidad por la cual ha sido pesado la cantidad de pedidos sin cumplir en la función objetivo.
- *Valor ópt.*: Suma ponderada del tiempo de espera y la cantidad de pedidos sin cumplir.
- *PSC*: Cantidad de pedidos sin cumplir.
- *WT*: Tiempo de espera total de ambos vehículos. Medido en minutos.

En la columna *Depósitos fijos* aparecen los resultados computacionales obtenidos para los 497 pedidos fijando los depósitos a las localizaciones 1 y 7, mientras que en la columna

Depósitos no fijos se presentan los obtenidos al dejar libertad en cuanto a la localización de los depósitos (en la solución localizados en 8 y 11).

Si comparamos ambos resultados se observa que efectivamente se obtiene una mejor solución si los depósitos de los vehículos se encuentran en las localizaciones 8 y 11 (Güímar y La Victoria respectivamente). Consiguiendo así, un menor número de pedidos sin completar al mismo tiempo que un menor tiempo de espera.

Notar que en el caso con depósitos en 8 y 11 la solución obtenida con $\beta = 1000$ domina a la obtenida con $\beta = 500$.

Capítulo 5

Conclusiones

En este Trabajo de Fin de Grado se ha estudiado el problema de recogida y entrega de mercancías estático y se ha modelado de acuerdo a las características propias de la empresa canaria Desguaces Tenerife S.A. Este modelo ha sido implementado en el resolutor Xpress y se han obtenido diferentes resultados para varios pedidos dando diferentes valores a los pesos de la función objetivo. En su resolución llegamos a la conclusión de que solamente puede ser resuelto de forma exacta para una cantidad muy pequeña de pedidos. En nuestro caso con a lo sumo 8 pedidos y $\alpha = 0$.

Debido a que la empresa Desguaces Tenerife S.A. recibe una cantidad aproximada de 50 pedidos diarios, la resolución de este problema no nos ayuda a decidir cuál sería la mejor estrategia a seguir en cuanto a la planificación de las rutas. Este hecho nos motiva a plantear otro problema en el que las rutas sean fijas y tengamos que decidir solamente el tiempo de espera en cada localización y la localización de los depósitos. Tras formular este problema de diversas maneras, se llega a la conclusión de que dependiendo de la importancia relativa de cada objetivo se obtiene una solución diferente, por ejemplo, si lo que premiamos es el completar el número de pedidos sin importarnos el tiempo de espera y los depósitos de los vehículos se siguen localizando en Icod de los Vinos y Arona, la mejor estrategia (en media) es comenzar a realizar las rutas dos horas y media después de la apertura de la empresa. Mientras que si podemos considerar que los vehículos partan de otras localizaciones, lo mejor es localizar los depósitos en Güímar y La Victoria puesto que se consigue realizar un mayor número de pedidos con un tiempo de espera menor.

En un futuro sería interesante, volver a ejecutar los modelos considerados, pero en este caso haciendo que Xpress parta de las soluciones no dominadas y analizar si los resultados que se obtienen mejoran los anteriores.

También sería bueno analizar si se producen simetrías y en el caso de que así sea, introducir en el modelo alguna restricción que las evite para así permitir una mejora en los tiempos de ejecución.

Apéndice A

Código en Xpress

A.1. Implementación del modelo PDP estático en Xpress

```
/##############################################################################
 # PDP_estático.mos
 ##############################################################################
 #
 # 27/04/2016
 #
 ##############################################################################
(!*************************************************

PDP estático

*************************************************!)

model "VRPPD"
uses "mmxprs","mmive","mmsystem";  ! Para obtener acceso a Xpress-Optimizer solver

parameters
N_LOC=7             ! Número de localizaciones
NDEPOTS=2           ! Número de depósitos
NVEHICLES=2         ! Número de vehículos
NREQUEST=8          ! Número de pedidos
ALPHA=1             ! Peso del tiempo de espera
end-parameters

setparam("xprs_loadnames",true)
forward procedure draw

declarations
x: array(-1..N_LOC) of real           !coordenada x de las localizaciones
y: array(-1..N_LOC) of real           !coordenada y de las localizaciones
LOC = (1)..N_LOC                      ! Localizaciones
VEHICLE = 1..NVEHICLES                ! Vehículos
REQUEST = 1..NREQUEST                 ! Nodos con recogida de mercancía
DELIVERY = NREQUEST+1..2*NREQUEST     ! Nodos con entrega de mercancía
DEPOTS = 1..NDEPOTS                   ! Depósitos
```

```
N = -1..2*NREQUEST                    ! Conjunto de todos los nodos
DEPOT: array(VEHICLE) of integer      ! Depósito de cada vehículo
LOCATION: array(N) of integer         ! Localización de cada nodo
DIST: array(LOC,LOC) of real          ! Distancia entre localizaciones
t: array(LOC,LOC) of integer          ! Tiempo de viaje entre localizaciones
c: array(LOC) of integer              ! Tiempo de proceso en las localizaciones
a: array(N)of integer                 ! Tiempo a partir del cual se puede empezar a servir
                                        a cada nodo
b: array(N)of integer                 ! Tiempo límite para servir a cada nodo (time window)
big_M: array(N,N) of real             ! Número suficientemente grande (M) para cada arco
assig: array(N,N,VEHICLE) of mpvar    ! 1 si el vehículo k va de i a j, 0 en otro caso
s:array (N,VEHICLE) of mpvar        ! Tiempo en el que el vehículo k empieza a servir al nodo i
visited: array(N,VEHICLE) of mpvar    ! 1 si el vehículo k visita el nodo i, 0 en otro caso
end-declarations

!Inicializamos el contador del tiempo de ejecución
starttime:= gettime

! Leemos los datos del problema desde un fichero externo "fichero.dat"
fopen("DIST.dat",F_INPUT)
forall(i,j in 1..N_LOC)read(DIST(i,j))
fclose(F_INPUT)

fopen("t.dat",F_INPUT)
forall(i,j in 1..N_LOC) read(t(i,j))
fclose(F_INPUT)

fopen("xy.dat",F_INPUT)
forall(i in -1..N_LOC) do
read(x(i))
read(y(i))
end-do
fclose(F_INPUT)

DEPOT(1) := -1
DEPOT(2) := 0

fopen("LOCATION.dat",F_INPUT)
forall(i in -1..2*NREQUEST)read(LOCATION(i))
fclose(F_INPUT)

fopen("a.dat",F_INPUT)
forall(i in -1..2*NREQUEST)read(a(i))
fclose(F_INPUT)

forall(i in N) do
b(i):=600
end-do

forall(i in LOC) do
c(i):=10
end-do

forall(i,j in N | i<>j) do
```

```
big_M(i,j) := b(i) + c(LOCATION(i)) + DIST(LOCATION(i),LOCATION(j)) - a(j)
if big_M(i,j) < 0 then
big_M(i,j):=0
end-if
end-do


writeln("Solving a PDP with n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA,".")

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!   Modelo

TotalDist:= sum(i,j in N, k in VEHICLE | i<>j) DIST(LOCATION(i),LOCATION(j)) * assig(i,j,k)
WaitingTime:= sum(i in REQUEST, k in VEHICLE) (s(NREQUEST+i,k)-a(i))


! Si se visita un cliente, debe haber un arco que entra y uno que sale
forall(i in N, k in VEHICLE) sum(j in N | i<>j) assig(i,j,k) = visited(i,k)
forall(i in N, k in VEHICLE) sum(j in N | i<>j) assig(j,i,k) = visited(i,k)

! Todo nodo debe ser visitado sólo 1 vez
forall(i in N) sum( k in VEHICLE) visited(i,k) = 1

forall(i in REQUEST, k in VEHICLE) visited(i,k) = visited(NREQUEST+i,k)

forall( k in VEHICLE) visited(DEPOT(k),k) =1

forall(i,j in N,k in VEHICLE | i<>j and j>0)
s(i,k) + (c(LOCATION(i)) +  t(LOCATION(i),LOCATION(j)))*assig(i,j,k)
                       <= s(j,k) + big_M(i,j)*(1-assig(i,j,k))

forall(i in REQUEST, k in VEHICLE)
s(i,k) + (c(LOCATION(i))+t(LOCATION(i),LOCATION(NREQUEST+i))) <= s((NREQUEST+i),k)

! Time Window
forall(i in N,k in VEHICLE) s(i,k) >= a(i)
forall(i in N,k in VEHICLE) s(i,k) <= b(i)


forall(i in REQUEST, k in VEHICLE)
    assig(DEPOT(k),NREQUEST+i,k)=0
forall(i in REQUEST, k in VEHICLE)
    assig(i,DEPOT(k),k)=0

forall(i,j in N,k in VEHICLE | i<>j) assig(i,j,k) is_binary
forall(i in N,k in VEHICLE) visited(i,k) is_binary
forall(i in N,k in VEHICLE) s(i,k)>=0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
setparam("XPRS_MAXTIME",1000)
minimize(TotalDist + ALPHA * WaitingTime)
draw
```

```
! Escribimos los resultados por pantalla
writeln("Optimal cost: ", getobjval)
write("Optimal tour has length ")
write(sum(i,j in N, k in VEHICLE | i<>j) DIST(LOCATION(i),LOCATION(j)) * getsol(assig(i,j,k)))
write(", total travel time ")
writeln(sum(i,j in N, k in VEHICLE | i<>j) t(LOCATION(i),LOCATION(j)) * getsol(assig(i,j,k)))
writeln ("and waiting time: ", getsol(sum (i in REQUEST, k in VEHICLE) (s(NREQUEST+i,k)-a(i))))

writeln("From node (loc) To node  (loc)   Car Distance   Arrive")
forall(k in VEHICLE, i,j in N | i<>j ) do
if (getsol(assig(i,j,k)) > 0.99) then
!writeln("x["+i+","+j+","+k+"]= "+getsol(assig(i,j,k))+ "   s("+j+","+k+")= "+getsol(s(j,k)))
write("     ",strfmt(i,3),"   (",strfmt(LOCATION(i),2),")")
write("     ",strfmt(j,3),"   (",strfmt(LOCATION(j),2),")")
write("   ",strfmt(k,3))
write("   ", strfmt(DIST(LOCATION(i),LOCATION(j)),6,1))
writeln("    ", strfmt(getsol(s(j,k)),5))
end-if
end-do

procedure draw
IVEerase
    IVEzoom(0,0,NREQUEST+1,NREQUEST+1)
    points:=IVEaddplot("locations",IVE_RED)
    depot:=IVEaddplot("depot",IVE_GREEN)
    roads1:=IVEaddplot("roads",IVE_BLUE)
    roads2:=IVEaddplot("roads",IVE_GREEN)
windows:=IVEaddplot("time windows",IVE_YELLOW)
    dist:=IVEaddplot("dist",IVE_BLACK)
    var_time:=IVEaddplot("arrive_time", IVE_MAGENTA)
    proceso:=IVEaddplot("process", IVE_CYAN)
forall(i in DEPOTS, k in VEHICLE) do
IVEdrawlabel(depot,x(DEPOT(i)),y(DEPOT(i)),""+i)
end-do
forall(i in LOC, k in VEHICLE) do
IVEdrawlabel(points,x(i),y(i),""+i)
IVEdrawlabel(windows,x(i),y(i)+1,"("+a(i)+","+b(i)+")")
IVEdrawlabel(var_time,x(i),y(i)-1, ""+round(getsol(s(i,k))))
IVEdrawlabel(proceso, x(i)+1,y(i), ""+c(i) )
end-do
!draw links
forall(i,j in N,k in VEHICLE) do
     if round(getsol(assig(i,j,k))) = 1 then
       if k =1 then
       IVEdrawarrow(roads1,x(LOCATION(i)),y(LOCATION(i)),x(LOCATION(j)),y(LOCATION(j)))
       end-if
       if k= 2 then
       IVEdrawarrow(roads2,x(LOCATION(i)),y(LOCATION(i)),x(LOCATION(j)),y(LOCATION(j)))
       end-if
     end-if
end-do
end-procedure
```

```
procedure write_results
writeln("Solving a PDP with n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA,".")
writeln("Optimal cost: ", getobjval)
write("Optimal tour has length ")
write(sum(i,j in N, k in VEHICLE | i<>j) DIST(LOCATION(i),LOCATION(j)) * getsol(assig(i,j,k)))
write(", total travel time ")
writeln(sum(i,j in N, k in VEHICLE | i<>j) t(LOCATION(i),LOCATION(j)) * getsol(assig(i,j,k)))
writeln ("and waiting time: ", getsol(sum (i in REQUEST, k in VEHICLE) (s(NREQUEST+i,k)-a(i))))

writeln("From node (loc) To node  (loc)   Car Distance   Arrive")
forall(k in VEHICLE, i,j in N | i<>j ) do
if (getsol(assig(i,j,k)) > 0.99) then
!writeln("x["+i+","+j+","+k+"]= "+ getsol(assig(i,j,k))+ "   s("+j+","+k+")= "+ getsol(s(j,k)))
write("     ",strfmt(i,3),"   (",strfmt(LOCATION(i),2),")")
write("     ",strfmt(j,3),"   (",strfmt(LOCATION(j),2),")")
write("   ",strfmt(k,3))
write("   ", strfmt(DIST(LOCATION(i),LOCATION(j)),6,1))
writeln("    ", strfmt(getsol(s(j,k)),5))
end-if
end-do
end-procedure


!Escribimos los resultados computacionales en un fichero "fichero.txt"
fopen("sol_mosel_PDP_estático.txt", F_OUTPUT + F_APPEND)
write_results
fclose(F_OUTPUT)

procedure write_latex
writeln(strfmt(NREQUEST,2)," & ",strfmt(ALPHA,1,1)," & ", strfmt(getobjval,4,1)," & ",
strfmt(sum(i,j in N, k in VEHICLE | i<>j) DIST(LOCATION(i),LOCATION(j))*getsol(assig(i,j,k)),3,1)," & ",
strfmt(sum(i,j in N, k in VEHICLE | i<>j) t(LOCATION(i),LOCATION(j))*getsol(assig(i,j,k)),3)," & ",
strfmt(getsol(sum (i in REQUEST, k in VEHICLE) (s(NREQUEST+i,k)-a(i))),4)," & ",
strfmt(getsol(gettime-starttime),4,1)," \\\\ ")
end-procedure

! Escribimos los resultados computacionales en un fichero "fichero.txt" con formato LaTeX
fopen("sol_latex_PDP_estático.txt", F_OUTPUT + F_APPEND)
write_latex
fclose(F_OUTPUT)

end-model

 ##############################################################################
```

A.2. Implementación del modelo lineal con rutas fijas y sentidos en Xpress

```
/##############################################################################
 # PDP_arcos_fijos_sentido.mos
 ##############################################################################
```

```
#
# 06/05/2016
#
################################################################################
(!*************************************************

PDP con arcos fijos y sentido

*************************************************!)

model "VRPL"
uses "mmxprs","mmive","mmsystem";  ! Para obtener acceso a Xpress-Optimizer solver

parameters
 N_LOC=7                          ! Número de localizaciones
 NSENTIDOS=2                      ! Número de sentidos
 NVEHICLES=2                      ! Número de vehículos
 NREQUEST=1118                    ! Número de pedidos
 ALPHA=1                          ! Peso del tiempo de espera
 BETA=1000                        ! Peso de los pedidos no realizados
 M=600                            ! Número suficientemente grande. Tomamos 600 puesto
                                    que es el tiempo que permance abierta la empresa
end-parameters

setparam("xprs_loadnames",true)

declarations
 LOC = 1..N_LOC                 ! Localizaciones
 VEHICLE = 1..NVEHICLES         ! Vehículos
 R = 1..NREQUEST                ! Pedidos
 SENTIDO = 1..NSENTIDOS         ! Sentidos (s=1 a favor de las agujas del reloj,
                                  s=2 en contra)
 o: array(R) of integer         ! Origen del pedido
 d: array(R) of integer         ! Destino del pedido
 S: array(R) of integer         ! Sentido del pedido
 t: array(LOC,LOC) of integer   ! Tiempo de viaje entre localizaciones
 p: array(LOC) of integer       ! Tiempo de proceso en las localizaciones
 a: array(R) of integer         ! Hora a la que se realiza el pedido (medido en minutos
                                  a partir de las 8:00)
 y: array(R,VEHICLE,SENTIDO) of mpvar    ! 1 si el pedido r es cargado en el vehículo
                                           k que va en el sentido s, 0 en otro caso.
 e: array(LOC,VEHICLE,SENTIDO) of mpvar  ! Hora (minutos a partir de las 8:00) a la que
                                           el vehículo k llega a i en sentido s
 l: array(LOC,VEHICLE,SENTIDO) of mpvar  ! Hora (en minutos a partir de las 8:00) a la
                                           que sale el vehículo k de i en sentido s
 h: array(R) of mpvar           ! Hora (en minutos a partir de las 8:00) a la que el
                                  pedido r es entregado en caso de que sea cargado,
                                  en otro caso h(r)=a(r)
end-declarations

!Inicializamos el contador del tiempo de ejecución
starttime:= gettime

! Leemos los datos del problema desde un fichero externo "fichero.dat"
```

```
fopen("ODS_oct_nov.dat",F_INPUT)
forall(r in 1..NREQUEST) do
 read(o(r))
 read(d(r))
 read(S(r))
end-do
fclose(F_INPUT)

fopen("CREACION_oct_nov.dat",F_INPUT)
forall(r in 1..NREQUEST)read(a(r))
fclose(F_INPUT)

fopen("t.dat",F_INPUT)
forall(i,j in 1..N_LOC) read(t(i,j))
fclose(F_INPUT)

forall(i in LOC) do
  p(i):=10
 end-do

writeln("Solving a PDP with fixed arcs and n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA," and beta = ", BETA,".")

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!   Modelo

Go:= sum(r in R, s in SENTIDO| s=S(r)) (1-(y(r,1,s)+y(r,2,s)))
WaitingTime:= sum(r in R) (h(r)-a(r))

forall(r in R, k in VEHICLE, s in SENTIDO) h(r) >= (e(d(r),k,s) - M*(1-y(r,k,s)))
forall(r in R, k in VEHICLE, s in SENTIDO|s=S(r)) a(r) <= (l(o(r),k,s) + M*(1-y(r,k,s)))
forall(r in R) h(r)>=a(r)

forall(i in LOC, k in VEHICLE, s in SENTIDO| (i<>7 OR k<>1 OR s<>2)
AND (i<>1 OR k<>2 OR s<>1) ) l(i,k,s) >= e(i,k,s)+ p(i)
l(7,1,2) >= e(7,1,1)+ p(7)+50
l(1,2,1) >= e(1,2,2)+ p(1)+50

forall(k in VEHICLE, i in 1..6) e(i+1,k,1) = l(i,k,1) + t(i,i+1)
forall(k in VEHICLE, i in 2..7) e(i-1,k,2) = l(i,k,2) + t(i,i-1)

forall(i in LOC, k in VEHICLE, s in SENTIDO) e(i,k,s) <= M

forall( r in R, s in SENTIDO| s=S(r)) y(r,1,s) + y(r,2,s)<=1
forall( r in R, s in SENTIDO| s<>S(r) ) y(r,1,s) = 0
forall( r in R, s in SENTIDO| s<>S(r) ) y(r,2,s) = 0

forall(i in LOC,k in VEHICLE, s in SENTIDO) l(i,k,s)>=0
forall(i in LOC,k in VEHICLE, s in SENTIDO) e(i,k,s)>=0
forall(r in R) h(r)>=0
forall(r in R,k in VEHICLE, s in SENTIDO) y(r,k,s) is_binary

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
setparam("XPRS_MAXTIME",3000)
```

```
minimize(ALPHA*WaitingTime + BETA*Go)

! Escribimos los resultados por pantalla
writeln("Optimal cost: ", getobjval)
writeln("Best Bound: "+ getparam("XPRS_BESTBOUND") +".  Running Time: "+
        strfmt(getsol(gettime-starttime),4,1))
write("Number of request unrealized in optimal tour: ")
write(sum(r in R, s in SENTIDO| s=S(r)) (1-(getsol(y(r,1,s))+getsol(y(r,2,s)))))
write(" and total waiting time: ")
writeln(sum(r in R) (getsol(h(r))-a(r)))
writeln("y[r,k,s] o(r) d(r) a(r) h(r)")
forall(k in VEHICLE, r in R, s in SENTIDO) do
 if (getsol(y(r,k,s)) > 0.99) then
 writeln("y["+strfmt(r,2)+","+k+","+s+"]= "+getsol(y(r,k,s))+" "+o(r)+" "+d(r)+"
        "+strfmt(a(r),3)+" h("+strfmt(r,2)+")= "+getsol(h(r)))
 end-if
end-do
writeln(" ")
writeln("e[i,k,s] l[i,k,s]")
forall(k in VEHICLE, s in SENTIDO, i in LOC) do
 writeln("e["+i+","+k+","+s+"]= "+getsol(e(i,k,s))+" "
        +"l["+i+","+k+","+s+"]= "+ getsol(l(i,k,s)))
end-do


procedure write_results
 writeln("Solving a PDP with fixed arcs and n=", NREQUEST," requests.")
 writeln("Solving with alpha = ", ALPHA," and beta = ", BETA,".")
 writeln("Optimal cost: ", getobjval)
 writeln("Best Bound: "+ getparam("XPRS_BESTBOUND") +".  Running Time: "+
         strfmt(getsol(gettime-starttime),4,1))
 write("Number of request unrealized in optimal tour: ")
 write(sum(r in R, s in SENTIDO| s=S(r)) (1-(getsol(y(r,1,s))+getsol(y(r,2,s)))))
 write(" and total waiting time: ")
 writeln(sum(r in R) (getsol(h(r))-a(r)))
 writeln("y[r,k,s] o(r) d(r) a(r) h(r)")
 forall(k in VEHICLE, r in R, s in SENTIDO) do
 if (getsol(y(r,k,s)) > 0.99) then
 writeln("y["+strfmt(r,2)+","+k+","+s+"]= "+getsol(y(r,k,s))+" "+o(r)+" "+d(r)+" "
        +strfmt(a(r),3)+" h("+strfmt(r,2)+")= "+getsol(h(r)))
 end-if
 end-do
 writeln(" ")
 writeln("e[i,k,s] l[i,k,s]")
 forall(k in VEHICLE, s in SENTIDO, i in LOC) do
 writeln("e["+i+","+k+","+s+"]= "+getsol(e(i,k,s))+" "
       +"l["+i+","+k+","+s+"]= "+ getsol(l(i,k,s)))
 end-do
 writeln("Running Time: "+ strfmt(getsol(gettime-starttime),4,1))
 writeln(" ")

end-procedure
```

```
!Escribimos los resultados computacionales en un fichero "fichero.txt"
fopen("sol_mosel_PDP_arcos_fijos_sentido.txt", F_OUTPUT + F_APPEND)
write_results
fclose(F_OUTPUT)

end-model

##############################################################################
```

A.3. Implementación del modelo circular con rutas fijas y localizaciones duplicadas en Xpress

```
##############################################################################
# PDP_arcos_fijos_localizaciones_duplicadas.mos
##############################################################################
#
# 12/05/2016
#
##############################################################################
(!*************************************************

PDP con arcos fijos y localizaciones duplicadas

*************************************************!)

model "VRPL"
uses "mmxprs","mmive","mmsystem"; ! Para obtener acceso a Xpress-Optimizer solver

parameters
N_LOC=12        ! Número de localizaciones
NVEHICLES=2     ! Número de vehículos
NREQUEST=1118   ! Número de pedidos
ALPHA=1         ! Peso del tiempo de espera
BETA=1000       ! Peso de los pedidos no realizados
M=600           ! Número suficientemente grande
end-parameters

setparam("xprs_loadnames",true)

declarations
LOC = 1..N_LOC                  ! Localizaciones
VEHICLE = 1..NVEHICLES          ! Vehículos
R = 1..NREQUEST                 ! Pedidos
DEPOT: array(VEHICLE) of integer      ! Depósito de cada vehículo
o: array(R) of integer                ! Origen del pedido
d: array(R) of integer                ! Destino del pedido
t: array(LOC,LOC) of integer          ! Tiempo de viaje entre localizaciones
p: array(LOC) of integer              ! Tiempo de proceso en las localizaciones
a: array(R) of integer          ! Hora a la que se realiza el pedido (medido en minutos
                                  a partir de las 8:00)
y: array(R,VEHICLE) of mpvar    ! 1 si el pedido r es cargado en el vehículo k, 0 en otro caso
```

```
l: array(LOC,VEHICLE) of mpvar   ! Hora (en minutos a partir de las 8:00) a la que
                                   sale el vehículo k de i
e: array(LOC,VEHICLE) of mpvar   ! Hora (minutos a partir de las 8:00) a la que el
                                   vehículo k llega a i
h: array(R) of mpvar             ! Hora (en minutos a partir de las 8:00) a la que el
                                   pedido r es entregado en caso de que sea cargado,
                                   en otro caso h(r)=a(r)
end-declarations

!Inicializamos el contador del tiempo de ejecución
starttime:= gettime

! Leemos los datos del problema desde un fichero externo "fichero.dat"
fopen("OD_oct_nov.dat",F_INPUT)
forall(r in 1..NREQUEST) do
read(o(r))
read(d(r))
end-do
fclose(F_INPUT)

fopen("CREACION_oct_nov.dat",F_INPUT)
forall(r in 1..NREQUEST)read(a(r))
fclose(F_INPUT)

fopen("tLVRP.dat",F_INPUT)
forall(i in 1..(N_LOC-1)) read(t(i,i+1))
fclose(F_INPUT)
t(12,1):=26

forall(i in LOC) do
p(i):=10
end-do

DEPOT(1):=1
DEPOT(2):=7

writeln("Solving a PDP with fixed arcs and n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA," and beta = ", BETA,".")

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!   Modelo

Go:= sum(r in R) (1-(y(r,1)+y(r,2)))
WaitingTime:= sum(r in R) (h(r)-a(r))

forall(r in R, k in VEHICLE) h(r) >= (e(d(r),k) - M*(1-y(r,k)))
forall(r in R, k in VEHICLE) a(r) <= (l(o(r),k) + M*(1-y(r,k)))
forall(r in R) h(r)>=a(r)

forall(i in LOC, k in VEHICLE| (i<>7 OR k<>1) AND (i<>1 OR k<>2) AND i<>DEPOT(k)) l(i,k)
                            >= e(i,k)+ p(i)
l(7,1) >= e(7,1)+ p(7)+50
l(1,2) >= e(1,2)+ p(1)+50
forall (k in VEHICLE) l(DEPOT(k),k) >= p(DEPOT(k))
```

```
forall(k in VEHICLE, i in 1..11) e(i+1,k) = l(i,k) + t(i,i+1)
forall(k in VEHICLE) e(1,k) = l(12,k) + t(12,1)

forall(i in LOC, k in VEHICLE) e(i,k) <= M

forall( r in R) y(r,1) + y(r,2)<=1

forall(i in LOC,k in VEHICLE) l(i,k)>=0
forall(i in LOC,k in VEHICLE) e(i,k)>=0
forall(r in R) h(r)>=0
forall(r in R,k in VEHICLE) y(r,k) is_binary

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
setparam("XPRS_MAXTIME",2000)
minimize(ALPHA*WaitingTime + BETA*Go)
!draw

! Escribimos los resultados por pantalla
writeln("Optimal cost: ", getobjval)
writeln("Best Bound: "+ getparam("XPRS_BESTBOUND") +".  Running Time: "+
        strfmt(getsol(gettime-starttime),4,1))
write("Number of request unrealized in optimal tour: ")
write(sum(r in R) (1-(getsol(y(r,1))+getsol(y(r,2)))))
write(" and total waiting time: ")
writeln(sum(r in R) (getsol(h(r))-a(r)))
writeln("y[r,k] o(r) d(r) a(r) h(r)")
forall(k in VEHICLE, r in R) do
if (getsol(y(r,k)) > 0.99) then
writeln("y["+r+","+k+"]= "+getsol(y(r,k))+" "+o(r)+" "+d(r)+" "+a(r)+"  h("+r+")= "+getsol(h(r)))
end-if
end-do
writeln(" ")
writeln("e[i,k] l[i,k]")
forall(k in VEHICLE, i in LOC) do
writeln("e["+i+","+k+"]= "+getsol(e(i,k))+" "+"l["+i+","+k+]= "+getsol(l(i,k)))
end-do


procedure write_results
writeln("Solving a PDP with fixed arcs and n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA," and beta = ", BETA,".")
writeln("Optimal cost: ", getobjval)
writeln("Best Bound: "+ getparam("XPRS_BESTBOUND") +".  Running Time: "+
         strfmt(getsol(gettime-starttime),4,1))
write("Number of request unrealized in optimal tour: ")
write(sum(r in R) (1-(getsol(y(r,1))+getsol(y(r,2)))))
write(" and total waiting time: ")
writeln(sum(r in R) (getsol(h(r))-a(r)))
writeln("y[r,k] o(r) d(r) a(r) h(r)")
forall(k in VEHICLE, r in R) do
if (getsol(y(r,k)) > 0.99) then
writeln("y["+r+","+k+"]= "+getsol(y(r,k))+" "+o(r)+" "+d(r)+" "+a(r)+"  h("+r+")= "+getsol(h(r)))
end-if
```

```
end-do
writeln(" ")
writeln("e[i,k] l[i,k]")
forall(k in VEHICLE, i in LOC) do
writeln("e["+i+","+k+"]= "+getsol(e(i,k))+" "+"l["+i+","+k+"]= "+getsol(l(i,k)))
end-do
writeln(" ")

end-procedure


!Escribimos los resultados computacionales en un fichero "fichero.txt"
fopen("sol_mosel_PDP_arcos_fijos_localizaciones_duplicadas.txt", F_OUTPUT + F_APPEND)
write_results
fclose(F_OUTPUT)

end-model

 ##############################################################################
```

A.4. Implementación del modelo circular con rutas fijas, localizaciones duplicadas y sin depósitos prefijados en Xpress

```
##############################################################################
# PDP_arcos_fijos_localizaciones_duplicadas_sin_depósitos.mos
##############################################################################
#
# 16/05/2016
#
##############################################################################
(!*************************************************

PDP con arcos fijos, localizaciones duplicadas y sin depósitos fijos

*************************************************!)

model "VRPL"
uses "mmxprs","mmive","mmsystem"; ! Para obtener acceso a Xpress-Optimizer solver

parameters
N_LOC=12        ! Número de localizaciones
NVEHICLES=2     ! Número de vehículos
NREQUEST=497    ! Número de pedidos
ALPHA=1         ! Peso del tiempo de espera
BETA=1000       ! Peso de los pedidos no realizados
M=600           ! Número suficientemente grande
end-parameters

setparam("xprs_loadnames",true)
```

```
declarations
LOC = 1..N_LOC              ! Localizaciones
VEHICLE = 1..NVEHICLES      ! Vehículos
R = 1..NREQUEST             ! Pedidos
o: array(R) of integer      ! Origen del pedido
d: array(R) of integer      ! Destino del pedido
t: array(LOC,LOC) of integer      ! Tiempo de viaje entre localizaciones
p: array(0..N_LOC) of integer     ! Tiempo de proceso en las localizaciones
a: array(R) of integer            ! Hora a la que se realiza el pedido (medido en minutos
                                    a partir de las 8:00)
y: array(R,VEHICLE) of mpvar      ! 1 si el pedido r es cargado en el vehículo k, 0 en otro caso
l: array(LOC,VEHICLE) of mpvar    ! Hora (en minutos a partir de las 8:00) a la que
                                    sale el vehículo k de i
e: array(LOC,VEHICLE) of mpvar    ! Hora (minutos a partir de las 8:00) a la que
                                    el vehículo k llega a i
h: array(R) of mpvar              ! Hora (en minutos a partir de las 8:00) a la que el
                                    pedido r es entregado en caso de que sea cargado,
                                    en otro caso h(r)=a(r)
DEPOT: array(LOC,VEHICLE) of mpvar        ! 1 si el depósito de k es i, 0 en otro caso
end-declarations

!Inicializamos el contador del tiempo de ejecución
starttime:= gettime

! Leemos los datos del problema desde un fichero externo "fichero.dat"
fopen("OD_oct_nov.dat",F_INPUT)
forall(r in 1..NREQUEST) do
read(o(r))
read(d(r))
end-do
fclose(F_INPUT)

fopen("CREACION_oct_nov.dat",F_INPUT)
forall(r in 1..NREQUEST)read(a(r))
fclose(F_INPUT)

fopen("tLVRP.dat",F_INPUT)
forall(i in 1..(N_LOC-1)) read(t(i,i+1))
fclose(F_INPUT)
t(12,1):=26

forall(i in LOC) do
p(i):=10
end-do


writeln("Solving a PDP with fixed arcs and n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA," and beta = ", BETA,".")

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!   Modelo

Go:= sum(r in R) (1-(y(r,1)+y(r,2)))
```

```
WaitingTime:= sum(r in R) (h(r)-a(r))

forall(r in R, k in VEHICLE) h(r) >= (e(d(r),k) - M*(1-y(r,k)))
forall(r in R, k in VEHICLE) a(r) <= (l(o(r),k) + M*(1-y(r,k)))
forall(r in R) h(r)>=a(r)

forall(i in 7..12,k in VEHICLE) l(i,k) >= e(i,k)+ p(i)+50*DEPOT(abs(i-6),k)- M* DEPOT(i,k)
forall(i in 1..6,k in VEHICLE) l(i,k) >= e(i,k)+ p(i)+50*DEPOT(i+6,k)- M* DEPOT(i,k)
forall (k in VEHICLE) sum(i in LOC) DEPOT(i,k) = 1

forall (k in VEHICLE, i in LOC) l(i,k) >= p(i)

forall(k in VEHICLE, i in 1..11) e(i+1,k) = l(i,k) + t(i,i+1)
forall(k in VEHICLE) e(1,k) = l(12,k) + t(12,1)

forall(i in LOC, k in VEHICLE) e(i,k) <= M

forall(r in R) y(r,1) + y(r,2)<=1

forall(i in LOC,k in VEHICLE) l(i,k)>=0
forall(i in LOC,k in VEHICLE) e(i,k)>=0
forall(r in R) h(r)>=0
forall(r in R,k in VEHICLE) y(r,k) is_binary
forall(i in LOC,k in VEHICLE) DEPOT(i,k) is_binary

!DEPOT(1,1)=1
!DEPOT(7,2)=1

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
setparam("XPRS_MAXTIME",2000)
minimize(ALPHA*WaitingTime + BETA*Go)

! Escribimos los resultados por pantalla
writeln("Optimal cost: ", getobjval)
writeln("Best Bound: "+ getparam("XPRS_BESTBOUND") +".  Running Time: "+
         strfmt(getsol(gettime-starttime),4,1))
write("Number of request unrealized in optimal tour: ")
write(sum(r in R) (1-(getsol(y(r,1))+getsol(y(r,2)))))
write(" and total waiting time: ")
write(sum(r in R) (getsol(h(r))-a(r)))
writeln("")
writeln("y[r,k] o(r) d(r) a(r) h(r)")
forall(k in VEHICLE, r in R) do
if (getsol(y(r,k)) > 0.99) then
writeln("y["+r+","+k+"]= "+getsol(y(r,k))+" "+o(r)+" "+d(r)+" "+a(r)+" h("+r+")= "+getsol(h(r)))
end-if
end-do
writeln(" ")
writeln("e[i,k] l[i,k]")
forall(k in VEHICLE, i in LOC) do
writeln("e["+i+","+k+"]= "+getsol(e(i,k))+" " +"l["+i+","+k+"]= "+getsol(l(i,k)))
end-do
writeln(" ")
writeln("depot[i,k]")
```

```
forall(k in VEHICLE, i in LOC) do
if (getsol(DEPOT(i,k)) > 0.99) then
writeln("depot["+i+","+k+"]= "+getsol(DEPOT(i,k)))
end-if
end-do



procedure write_results
writeln("Solving a PDP with fixed arcs and n=", NREQUEST," requests.")
writeln("Solving with alpha = ", ALPHA," and beta = ", BETA,".")
writeln("Optimal cost: ", getobjval)
writeln("Best Bound: "+ getparam("XPRS_BESTBOUND") +".  Running Time: "+
         strfmt(getsol(gettime-starttime),4,1))
write("Number of request unrealized in optimal tour: ")
write(sum(r in R) (1-(getsol(y(r,1))+getsol(y(r,2)))))
write(" and total waiting time: ")
writeln(sum(r in R) (getsol(h(r))-a(r)))
writeln("y[r,k] o(r) d(r) a(r) h(r)")
forall(k in VEHICLE, r in R) do
if (getsol(y(r,k)) > 0.99) then
writeln("y["+r+","+k+"]= "+getsol(y(r,k))+" "+o(r)+" "+d(r)+" "+a(r)+"  h("+r+")= "+getsol(h(r)))
end-if
end-do
writeln(" ")
writeln("e[i,k] l[i,k]")
forall(k in VEHICLE, i in LOC) do
writeln("e["+i+","+k+"]= "+getsol(e(i,k))+" "+"l["+i+","+k+"]= "+getsol(l(i,k)))
end-do
writeln(" ")
writeln("depot[i,k]")
forall(k in VEHICLE, i in LOC) do
if (getsol(DEPOT(i,k)) > 0.99) then
writeln("depot["+i+","+k+"]= "+getsol(DEPOT(i,k)))
end-if
end-do
end-procedure


!Escribimos los resultados computacionales en un fichero "fichero.txt"
fopen("sol_mosel_PDP_arcos_fijos_sin_depósitos.txt", F_OUTPUT + F_APPEND)
write_results
fclose(F_OUTPUT)


end-model

 ##############################################################################
```

Apéndice B

Optimización Multiobjetivo

En los problemas de optimización se trata de encontrar una solución que represente el valor óptimo para la función objetivo.

Como puede observarse en la formulación del problema para el caso estático (Capítulo 3), la función objetivo está compuesta por varios objetivos, se desea minimizar la distancia total recorrida así como el tiempo total de espera. Esto es debido a que se trata de un problema multiobjetivo.

En muchas ocasiones se dan problemas de este tipo que requieren la optimización simultánea de más de un objetivo. Habrá que optimizar por tanto una función de la forma $f : S \to T$, donde $S \subset \mathbb{R}^n$ y $T \subset \mathbb{R}^k$. Pero el problema está en que normalmente no existe un elemento de S que produzca un óptimo de forma simultánea para cada uno de los k objetivos que componen la función objetivo. Esto se deberá a la existencia de conflictos entre objetivos, que harán que la mejora de uno de ellos dé lugar a un empeoramiento de algún otro. Habrá que llegar por tanto a una situación de compromiso en la que todos los objetivos sean satisfechos en un grado aceptable.

La programación multiobjetivo pretende entonces establecer el conjunto de soluciones eficientes (no dominadas o óptimos de Pareto) en vez de buscar un único óptimo.

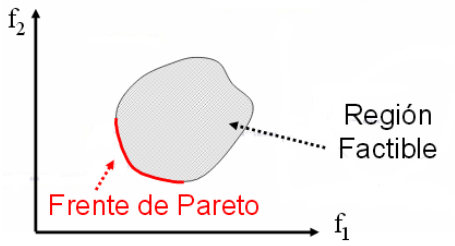


Figura B.1: Región factible con objetivos f_1 y f_2 a ser minimizados.

El conjunto eficiente denominado frente de Pareto está formado por soluciones factibles (esto es, que cumplen las restricciones) tales que no existe otra solución factible que proporcione una mejora en un objetivo sin producir un empeoramiento en al menos otro de

los objetivos. El resto de soluciones que yacen en la región factible son dominadas (siempre hay otra solución en el frente de Pareto que tiene al menos un objetivo mejor). Notar que si el problema es no lineal o muy complejo, el simple hecho de obtener una sola solución tal vez no sea trivial.

A diferencia de los problemas de optimización con un único objetivo, el concepto de óptimo es ahora relativo y será necesario decidir de alguna forma cuál es la mejor solución (o cuáles son las mejores soluciones) al problema.

Para tratar el problema del conflicto entre objetivos se pueden utilizar diversos métodos:

- Métodos basados en el concepto de eficiencia de Pareto.

  Tratan de encontrar en un solo proceso de optimización varias soluciones del frente de Pareto. En varias iteraciones se realiza una mejora progresiva del conjunto no dominado.

- Métodos basados en la combinación de objetivos.

  Dentro de estos métodos se puede mencionar el método de la suma ponderada, en el que se optimizará el valor obtenido mediante la suma de los valores correspondientes a los distintos objetivos, multiplicados cada uno por un coeficiente de peso. Estos coeficientes de peso establecerán la importancia relativa de cada objetivo. El problema de optimización multiobjetivo se transforma así en otro de optimización escalar. Existen variantes del método anterior, como el método de la programación por metas, en el que se establece una meta para cada objetivo y lo que se suma ahora (multiplicado por el correspondiente coeficiente) es la distancia de cada objetivo a su meta.

- Métodos basados en la asignación de prioridades.

  Estos métodos tienen en común que establecen unas prioridades entre los distintos objetivos, teniéndose en cuenta su importancia relativa durante el proceso de optimización.

Para resolver este problema en la función objetivo de nuestra formulación del PDP estático (Capítulo 3, sección 3.2), hemos utilizado un método basado en la combinación de objetivos. Le hemos dado a cada objetivo un peso que refleja la importancia relativa de cada componente. Variando los pesos se obtiene un conjunto de soluciones eficientes. El objetivo es minimizar la suma ponderada de la distancia total recorrida y del tiempo total de espera.

Referencias

Anily, S. y Hassin, R. (1992). The swapping problem. *Networks*, *22*(4), 419–433.

Anily, S. y Pfeffer, A. (2013). The uncapacitated swapping problem on a line and on a circle. *Discrete Applied Mathematics*, *161*(4–5), 454 – 465.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. y Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, *15*(1), 1–31.

Berbeglia, G., Cordeau, J.-F. y Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*(1), 8 – 15.

Berbeglia, G., Pesant, G. y Rousseau, L.-M. (2012). Feasibility of the pickup and delivery problem with fixed partial routes: A complexity analysis. *Transportation Science*, *46*(3), 359–373.

Bodin, L. y Golden, B. (1981). Classification in vehicle routing and scheduling. *Networks*, *11*(2), 97–108.

Chalasani, P. y Motwani, R. (1999). Approximating capacitated routing and delivery problems. *SIAM Journal on Computing*, *28*(6), 2133–2149.

Clarke, G. y Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568–581.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*(3), 573–586.

Cordeau, J.-F. y Laporte, G. (2003a). The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, *1*(2), 89–101.

Cordeau, J.-F. y Laporte, G. (2003b). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, *37*(6), 579–594.

Cordeau, J.-F., Laporte, G. y Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. En B. Golden, S. Raghavan y E. Wasil (Eds.), *The vehicle routing problem* (pp. 327–357). Boston, MA: Springer.

Dantzig, G. y Ramser, J. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80–91.

Desrochers, M., Lenstra, J. y Savelsbergh, M. (1990). A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, *46*(3), 322–332.

Hernández-Pérez, H. y Salazar-González, J.-J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*,

145(1), 126–139.

Meindl, B. y Templ, M. (2013). Analysis of commercial and free and open source solvers for the cell suppression problem. *Transactions on Data Privacy*, *6*(2), 147–159.

Parragh, S. N., Doerner, K. F. y Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, *58*(1), 21–51.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, *14*(2), 130–154.