Sección de Matemáticas
Universidad de La Laguna

Yanira García Hernández

Optimización inteligente de rutas de recogida de residuos

Intelligent Optimization of Waste Collection Routes

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Junio de 2019

Dirigido por
José Andrés Moreno Pérez

José Andrés Moreno Pérez
Ingeniería Informática y de Sistemas
Universidad de La Laguna
38271 La Laguna, Tenerife

Agradecimientos

En primer lugar me gustaría agradecerle a mi tutor, José Andrés Moreno Pérez, por todo su tiempo y dedicación en este proyecto.

Gracias a mi familia, sobretodo a mis padres que me apoyaron sin reservas desde el principio. A mi hermana por darme este año el mejor regalo posible y a mis tíos por ayudarme en todo momento.

Gracias a mis amigos por estar siempre conmigo y hacer de este largo viaje nuestra gran aventura.

Y, por último pero no menos importante, gracias a Dani por decidir acompañarme y ser un apoyo incondicional para mí.

Yanira García Hernández
La Laguna, Junio de 2019

Resumen · Abstract

Resumen

La creciente preocupación por los efectos negativos en el medio ambiente, debido a la contaminación provocada por la generación de residuos, ha motivado la movilización no sólo de gobiernos, sino también de empresas y la sociedad con la finalidad de establecer las medidas necesarias para controlar y revertir esta situación. Una de las actividades en las que más se ha hecho hincapié es en la clasificación y recogida de residuos. El principal objetivo de esta tarea es aumentar la cantidad de residuos que pueden ser reciclados. En esta memoria trataremos de presentar el uso de herramientas matemáticas para optimizar las rutas de recogida de residuos y así maximizar la cantidad de residuos recogidos para su posterior reciclaje.

Palabras clave: *Optimización Combinatoria – Recogida de residuos – Problemas de rutas – GUSEK.*

Abstract

The growing negative effects concerning the environment, due to pollution caused by waste generation, have induced the movements of, not only governments, but also, companies and society with the purpose of establishing the necessary activities to control and reverse this situation. One of the activities that has been made with more emphasis is the waste classification and collection. The main goal of this task is to increase the amount of waste that can be recycled. In this work we aim to show the use of mathematical tools for optimizing the routes for waste collection and so maximizing the amount of collected waste to be recycled.

Keywords: *Combinatorial Optimization – Waste Collection – Routing Problems – GUSEK.*

Contenido

Introducción

Varias áreas científicas y técnicas están implicadas en la resolución de problemas de optimización que surgen en situaciones reales, como pueden ser la gestión de recursos y la toma de decisiones. Para esta tarea se aplican distintas herramientas matemáticas, como modelos y algoritmos, para representar y abordar diversos problemas basados en situaciones reales, facilitando la búsqueda de soluciones que satisfagan los requerimientos del problema de manera óptima. Dentro del campo de la Investigación Operativa encontramos la *Programación Matemática* cuyo principal objetivo es resolver los llamados problemas de optimización. Los problemas de optimización consisten en buscar soluciones de manera que se optimice una o varias funciones, denominadas como función objetivo, y satisfaga una serie de restricciones. Una de las claves importante a la hora de aplicar herramientas de Programación Matemática para resolver este tipo de problemas, es el modelo matemático que refleja la situación a estudiar. Dicho modelo representa los elementos que interactúan en el problema real en forma de variables y las ecuaciones o inecuaciones son las encargadas de establecer la relación entre dichas variables. Según las características de las variables y de las ecuaciones (o inecuaciones) podemos encontrar diversas áreas de la Programación Matemática como son Programación Lineal cuando las ecuaciones son funciones lineales de las variables, o la Programación Entera, cuando las variables pueden tomar sólo valores enteros.

En este trabajo tratamos con un tipo de problema de *Optimización Combinatoria*. Estos son problemas de optimización de la forma:

$$\{Minf(x) : x \in S\}$$

determinados por un espacio de soluciones S y una función objetivo f con la característica de que el espacio de soluciones es finito. Esto no tiene por qué implicar que el problema sea fácil de resolver ya que describir las soluciones

y seleccionar la óptima puede ser inviable de realizar en la práctica debido a que el número de soluciones puede ser excesivamente grande. Algunos de los problemas más estudiados en optimización combinatoria son el *Problema del Viajante de Comercio* (Travelling Salesman Problem, TSP) o el *Problema del Arbol Generador Mínimo* (Minimum Spanning Tree Problem, MSTP). En este trabajo aplicaremos herramientas matemáticas para resolver un caso real de este tipo de problemas.

En particular, uno de los problemas reales que más nos afecta es el cambio climático y sus efectos en el medio ambiente, y para luchar contra ello han nacido iniciativas de reciclaje que pretenden reducir la contaminación que provoca la actividad humana en el medio. Algunas de estas iniciativas consisten en aumentar la cantidad de residuo que se recicla, y para ello, se propone mejorar tanto el proceso de separación de residuos como el de la recogida de los mismos. Es esta línea se enmarca este trabajo que trata de estudiar, inspirado en la situación real, el problema de optimización combinatoria que consiste en determinar las rutas óptimas de recogida de residuos para reciclaje. La información real corresponde al sistema de recogida de residuos de la isla de La Palma donde se conoce la capacidad y ubicación de los contenedores de residuos reciclables y la velocidad con la que se llenan los contenedores con los residuos reciclables que deposita en ellos los ciudadanos.

La intención de minimizar, en la medida de lo posible, el impacto contra el medio ambiente que se produce mientras los residuos están depositados en los contenedores situados en las vías públicas en espera de ser recogidos. El problema consistirá en definir una ruta que pase por los puntos donde se encuentren los contenedores más llenos para su recogida, maximizando así la cantidad de residuo recogida para su posterior reciclaje. La tarea de establecer la ruta para el recorrido de un vehículo aparece frecuentemente en la planificación y logística del transporte y este tipo de problemas se conocen como *Problemas de Rutas de Vehiculos*, llamados comúnmente por su literatura en inglés *Vehicle Routing Problems* (VRP), dentro de la programación combinatoria.

De esta manera, la memoria queda estructurada en dos capítulos. El primero consiste un una breve revisión del estado actual del problema de optimización de rutas además del desarrollo teórico del mismo. En el segundo capítulo, podemos encontrar los resultados de aplicar datos reales al problema estudiado en el primer capítulo haciendo uso de un software libre matemático diseñado para resolver problemas de programación matemática.

1

Capítulo 1. Rutas de Recogida de Residuos.

En este capítulo se recoge en la primera sección una revisión de las investigaciones publicadas sobre la optimización en el proceso de recogida de residuos para reciclaje. En la segunda sección se describen y formulan los problemas de programación matemáticas más importantes que aparecen en la optimización de las rutas de recogida de residuos para reciclaje.

1.1. Revisión del estado del arte

En esta sección realizaremos una breve revisión de los avances que se han llevado a cabo sobre la optimización del proceso de reciclaje en general y sobre la optimización en rutas de recogida de residuos en particular, con la finalidad de hacer una selección de las fuentes más importantes además de los principales conceptos que ya se han estudiado.

1.1.1. Optimización del reciclaje

Es un hecho que toda actividad humana tiene como consecuencia la generación de algún tipo de residuo, aunque esto no provocaba ninguna dificultad mayor cuando la población mundial era relativamente pequeña. El problema surge cuando debido al aumento de la población, la urbanización y el desarrollo económico, se disparan los niveles de residuos que se generan. El último estudio, realizado por el Banco Mundial, llamado *What a Waste 2.0* [14] prevé que, si seguimos en las mismas condiciones, respecto a los 2.001 millones de toneladas que aproximadamente se generaron en el 2016, para el 2050 esta cantidad ascenderá

a los 3.040 millones de toneladas. Una mala gestión de dichos residuos puede conducir a la contaminación del agua, suelo y atmósfera y a tener un impacto negativo sobre la salud pública. El estudio realizado por *Giusti* [8] trata los efectos que tienen sobre la salud pública las distintas formas de tratar los residuos como son: vertederos, incineradores, etc. Aunque no hay evidencias suficientes para concluir una relación directa entre las enfermedades de la población cercana a las zonas de tratamiento de residuos y las plantas de tratamiento de los mismos, la población en los países desarrollado, debido al miedo de desarrollar alguna de las enfermedades, ha priorizado la mejora de la gestión de residuos. Mientras, en los países en desarrollo debido a alta tasa de mortalidad por culpa de la falta de agua potable o la pobre sanidad, entre otras cosas, hacen que la gestión de residuos pase a un segundo plano.

En las Islas Canarias el turismo, y el sector de servicios en general, es el principal motor de la economía canaria, lo que incrementa, aunque no de forma homogénea, el impacto en el medio de la gestión de residuos. Además de las consecuencias del aumento y concentración de la población, a las Islas Canarias debemos sumarle también la cantidad de visitantes que provoca un aumento del 46 % en la densidad de la población. Debido al aumento de turistas el sector se ha visto obligado a aumentar la cantidad y tamaño de las infraestructuras turísticas lo que, junto con la actividad de los propios turistas, proporcionan efectos negativos en el medio ambiente. Estos efectos son, por ejemplo, el aumento del consumo de energía, que genera una aumento de los niveles de contaminación al vernos obligados a aumentar el número de plantas de generación de energía, o el aumento de la generación de residuos en lugares de interés turísticos o cercanos a infraestructuras turísticas lo que ha obligado a cambiar la estrategia de gestión de residuos [10].

Ante esta situación se ha estudiado las distintas vías por las cuales podemos disminuir, o incluso revertir, los efectos negativos que tiene la generación de residuos en el medio. Y, aunque la mejora de la gestión de residuos no es la única alternativa, destacamos el desarrollo de la *Economía Circular*[1] como uno de los frutos que nace del reciclaje de los residuos. El término surge a partir de tres acciones principales: reducir, reutilizar y reciclar y fue utilizado por primera vez por Pearce y Turner [17]. El concepto que describe es un sistema donde economía y medio ambiente interaccionan por igual, es decir, un sistema que haga un uso más responsable, medioambientalmente hablando, de los recursos. Así, un recurso destinado para un determinado producto después de terminar su vida útil puede volver a convertirse de nuevo en un recurso evitando así generar nuevos residuos. Por tanto, llegamos a contemplar la gestión de residuos, no como la vía para deshacernos de los deshechos sino, como una parte de le

[1] https://www.ecoembes.com/es/ciudadanos/ecoembes-y-el-medio-ambiente/la-economia-circular-en-espana

economía circular donde la prioridad es reciclar los residuos evitando un impacto negativo en el medio a través de la aplicación de tecnologías innovadoras [7]. En España, la organización medioambiental Ecoembes en pro de la economía circular ha creado **TheCircularLab**, un laboratorio que mediante la investigación cooperativa tiene como finalidad innovar en el área de gestión de residuos y reciclaje mediante las siguientes líneas de investigación: el desarrollo de envases más sostenibles, la creación de herramientas que faciliten la separación de los envases para la población y el uso de la tecnología para mejorar los procesos de recogida, selección y reciclado.

Es en este punto donde la tecnología y el proceso de recogida y tratamiento de residuos se mezclan con el objetivo de reducir el impacto de este proceso en el medio. Gracias al nacimiento del Internet de las cosas, conocido como **IoT** por sus siglas en inglés *Internet of Things* y que potencia a que los objetos cotidianos se comuniquen a través de Internet, se han podido desarrollar las herramientas necesarias para el uso de las Tecnologías de la Información y Comunicación las cuales mejorarán, con el uso de la información a tiempo real, la toma de decisión en sus operaciones. Aunque esto no es suficiente para crear un modelo eficiente de recogida de residuos, sino que se deberá complementar con optimización de rutas de vehículos [18]. Un ejemplo de esto se muestra en *Faccio et al.* [4] donde muestran como, a través de sensores volumétricos o tecnología como GPS, obtienen datos en tiempo real que ayudarán a implementar un modelo de recogida de residuos eficiente y respetuoso en la medida de lo posible con el medio ambiente. El modelo que proponen logra minimizar el número de vehículos necesarios y el tiempo total aumentando la distancia recorrida mientras reduce el impacto medioambiental como emisiones, ruidos o atascos de tráfico. Esta interacción entre el internet de las cosas y la optimización de rutas es lo que en inglés se denomina *Smart Waste Collection*, es decir, recogida inteligente de residuos.

1.1.2. Optimización de rutas de recogida

La logística es el término que engloba a cualquier procedimiento en el que está involucrado el transporte, almacenamiento y tratamiento de productos desde la fuente hasta su punto final. Hasta hace no mucho tiempo el objetivo de la logística era exclusivamente el maximizar los beneficios de manera que solo pretendía construir modelos que básicamente minimizaran los costos económicos. Pero en los últimos 20 años, la presión tanto social como de algunos gobiernos para reducir el impacto medioambiental de los procedimientos de logística, ha provocado el nacimiento de un nuevo término: *Green Logistic*. La logística verde es el tipo de logística que tiene como estrategia priorizar la conservación del medio en todos sus procesos [15].

Una parte de esta logística verde, y la que vamos a estudiar en esta memoria, trata los problemas sobre cómo plantear una ruta de una flota de vehículos conocidos como *Vehicle Routing Problems* (VRP) pero teniendo en cuenta el impacto medio ambiental de estas rutas. En la literatura los primeros en tratar con este tipo de problemas fueron Dantzig y Ramser en (1959) [2] con un problema llamado *"Truck Dispatching Problem"* en el cual modelaban una ruta para la recogida de aceite de un número determinado de gasolineras con una flota homogénea de vehículos de manera que se intentara minimizar la distancia a recorrer.

De forma más específica, y aunque con menos representación en la literatura, los problemas que se centran en la recolección de residuos y que tienen en cuenta reducir el impacto medioambiental, ya sea usando el menor número posible de vehículo, obteniendo la ruta más corta, etc., se conocen como WCVRP *(Waste Collection Vehicle Routing Problem)*. Se basan en el VRP y lo que ha llamado la atención del WCVRP es el uso de datos en tiempo real, así como el tipo de restricciones que pueden usarse. En *Idrus et al.* [13] podemos encontrar diferentes tipos de restricciones que pueden combinarse con el VRP clásico para formular nuevos problemas dependiendo de las necesidades que queramos priorizar. Algunas de estas restricciones son: capacidad del vehículo, se conoce el volumen o peso del vehículo y cuando se alcanza el máximo se envía al depósito; ventanas de tiempo, entre las que encontramos las que restringe la recogida de residuos, el tiempo total de recolección o aquellas para dar descanso a los conductores de los vehículos; las que controlan los números de vehículos o el número de depósitos. Cabe destacar que en la literatura no se contempla en ningún momento el maximizar la cantidad de residuos que se debe recoger sino que la mayoría de los artículos que encontramos se centran en minimizar la longitud de las rutas que debemos seguir o en reducir la cantidad de vehículos a usar, entre otros objetivos. Esto se debe a la idea de que,por ejemplo, cuanto menor sea el número de vehículos que circulen y menor sea el recorrido de estos la cantidad de contaminación que generan será menor. Pero esto no es el único aspecto a tener en cuenta en las rutas dado que los residuos deben recogerse lo antes posible para reducir el impacto ambiental.

En cuanto a la resolución de estos problemas debemos tener en cuenta que VRP es NP-hard, es decir, no existen algoritmos deterministas que resuelvan estos problemas en tiempo polinomial. [5] Así en WCVRP se tratan problemas difíciles de resolver ya que debido a las restricciones se convierte en un problema aún más complicado que el VRP. Para abordar este tipo de problemas debemos usar otro tipo de herramientas que nos permitan llegar, en un tiempo razonable, a una solución de alta calidad, aunque no se garantice que sea la óptima. A este tipo de herramientas se les denomina heurísticas y metaheurísticas. Una heurística es un procedimiento inteligente para realizar una tarea que nace del co-

nocimiento experto del tema en relación y no de un análisis formal del problema *Melián et al.* [16]. Las metaheurísticas son estrategias generales para construir heurísticas para problemas de optimización con un bien rendimiento. Este tipo de procedimientos aportan buenas soluciones si comparamos la calidad de estas con los recursos utilizados. En la literatura encontramos algunos ejemplos de los algoritmos heurísticos que proveen buenas soluciones en un tiempo razonable, entre ellos destacamos: los algoritmos de construcción, métodos de dos y tres fases, etc [12]. Entre las metaheurísticas destacan: los algoritmos genéticos, la búsqueda tabú, algoritmo de la colonia de hormigas, los procesos GRASP, o la Búsqueda de Entorno variable o VNS [16].

1.2. Optimización de rutas de recogida

En esta sección describimos la problemática de la planificación de las rutas de recogida de residuos, a continuación definimos el problema el problema de optimización de rutas en el que se centra el Trabajo Fin de Grado y establecemos su formulación como un problema de programación lineal combinatoria. La sección finaliza con algunas consideraciones sobre las extensiones apropiadas para una aplicación a situaciones más reales.

1.2.1. Descripción de los problemas de optimización de las rutas

En la planificación de la recogida de residuos depositado en contenedores, el objetivo fundamental es establecer las rutas entre los puntos donde se ubican contenedores que permita recoger la mayor cantidad de residuos posible. Para elegir los puntos que determinaran la ruta se tiene en cuenta un valor asignado a cada punto que representa la cantidad de residuo depositada en los correspondientes contenedores. Así, empezando y terminando en el punto inicial y final, respectivamente, los puntos a incluir en el camino serán aquellos que den lugar al mayor valor total posible. En general existe un límite de tiempo en el que recorrer los puntos de de recogida. Este tipo de problemas se han denominado en la literatura sobre problemas de optimización como *Orienteering Problems* (Problemas de Orientación) debido a un conocido deporte de origen Escandinavo al aire libre llamado en inglés *Orienteering*. En este juego existen los llamados "puntos de control", con un valor asociado, donde los competidores con ayuda de un mapa y una brújula, tienen que visitar un subconjunto de dichos puntos de control y obtener el mayor valor en el menor tiempo posible [9]. El *Orienteering Problem* puede ser considerado como un problema relacionado con el famoso problema del viajero (*Travelling Salesman Problem*). De forma más exacta, el

Selective Traveling Salesman Problem (STSP) [6] que consiste en establecer un tour o ciclo que visite una sola vez un subconjunto de puntos, algunos de ellos obligatorios, que maximice el beneficio sin sobrepasar un límite de tiempo.

En el diseño y optimización de las rutas de recogida pueden intervenir más condiciones de las descritas anteriormente. Las rutas suelen estar limitados por un tiempo máximo de recorrido y la capacidad de los vehículos puede hacer inviable la recogida de todos los contenedores que se pretende. Además, las visitas a algunos puntos de recogida pueden estar limitadas a ciertos intervalos de tiempo y pueden existir contenedores que es inevitable recoger. En general, interesa que las rutas sean lo más cortas posible pero que contengan el mayor número de contenedores y con el mayor nivel de llenado que se pueda, atendiendo a estas y otras restricciones que se deben añadir al problema.

1.2.2. Definición formal del problema

En esta subsección abordamos la definición formal del problema anteriormente descrito que coincide esencialmente con un problema estudiado en la literatura científica y conocido como Orienteering Problem (OP) [21] y varias vesriones o extensiones del mismo. Formalmente, el Orienteering Problem (OP) consiste en determinar una ruta óptima visitando una selección de un conjunto de puntos donde a cada uno se le tiene asignado un valor positivo, que se suele llamar beneficio. Se conoce el tiempo empleado en ir un punto a otro, y además se conoce el punto de origen O de inicio de la ruta y el punto final F que marcarán donde empieza y acaba la ruta. El objetivo del problema es construir un camino que recorra un subconjunto de nodos maximizando el beneficio total en el tiempo límite. Además, debe tenerse en cuenta que cada punto puede visitarse como mucho una vez. Formalmente, el Orienteering Problem (OP) es un problema NP-duro.

1.2.3. Modelo matemático

El problema que se ha definido es un OP que se puede formular como un problema lineal entero siguiendo la metodología que aparece en [21] con los parámetros y variables que le siguen.

Denotemos por $m \in \mathbb{Z}$ el número de puntos que es posible visitar y representamos por $I = \{0, ..., m+1\}$ el conjunto total de puntos, donde denotaremos al puntos de inicio con el índice 0 y al punto final con el índice $m+1$.

Parámetros:

Para cada punto $i \in I$, $1 \leq i \leq m$, le corresponde un valor o beneficio v_i. El tiempo para viajar desde un punto i a otro punto j viene representado por el parámetro t_{ij} $(i, j \in I)$,y es conocido de antemano. Además T_{max} representa el tiempo máximo de ruta.

Por tanto, consideramos los siguientes parámetros:

$$v_i : \text{beneficio del punto } i \text{ de visita} \qquad \forall i = 1, ..., m \tag{1.1}$$

$$t_{ij} : \text{tiempo de ir de } i \text{ hasta } j \qquad \forall i, j = 0, ..., m+1 \tag{1.2}$$

$$T_{max} : \text{tiempo máximo de la ruta} \tag{1.3}$$

Variables:

Las rutas posibles se identifican con las variables x_{ij}, $i, j \in I$, que establecen si el vehículo va del punto i al punto j, $i, j \in I$ y las variables p_i, $i \in I$ que establecen la posición del punto i en la ruta. Por tanto, las variables para definir las soluciones del problema son:

$$x_{ij} = \begin{cases} 1 & \text{si la ruta va de } i \text{ a } j \\ 0 & \text{en otro caso} \end{cases}, \ \forall i, j = 0, ..., m+1$$

$$p_i = \text{posición del nodo } i \text{ en la ruta, } \forall i = 0, ..., m+1$$

Objetivo:

El objetivo más general al planificar las rutas de recogida es recoger la mayor cantidad de residuo posible, que en nuestro caso es la suma de las cantidades depositadas en los puntos que visita la ruta. Por tanto, el objetivo de maximizar el valor total de la ruta se establece por:

$$\text{Max} \sum_{i=1}^{m} \sum_{j=1}^{m+1} v_i x_{ij} \tag{1.4}$$

Restricciones:

Las restricciones que deben cumplir las variables para que correspondan a una solución factible son las siguientes:

$$\sum_{j=1}^{m+1} x_{0j} = \sum_{i=0}^{m} x_{i(m+1)} = 1, \tag{1.5}$$

$$\sum_{i=0}^{m} x_{ik} = \sum_{j=1}^{m+1} x_{kj} \leq 1, \quad \forall k = 1, .., m \tag{1.6}$$

$$\sum_{i=0}^{m} \sum_{j=1}^{m+1} t_{ij} x_{ij} \leq T_{max}, \tag{1.7}$$

$$p_i - p_j + 1 \leq m(1 - x_{ij}), \quad \forall i, j = 0, ..., m+1 \tag{1.8}$$

$$1 \leq p_i \leq m, \quad \forall i = 0, ..., m+1 \tag{1.9}$$

$$p_i \in \mathbb{Z}, \quad \forall i = 0, ..., m+1 \tag{1.10}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 0, ..., m+1 \tag{1.11}$$

La función objetivo (1.4) maximiza el beneficio total de la ruta. Las restricciones (1.5) se encargan de que la ruta se inicie en el nodo 0 y finalice en el nodo $m + 1$. La restricción (1.6) se asegura de que los nodos seleccionados para el camino sean visitados solo una vez y a la vez garantiza la conectividad de la ruta. Las restricciones (1.7) limita el tiempo de ruta determinado por el parámetro T_{max}. Las restricciones (1.8) eliminan los posibles subciclos que se pueden generar en la ruta. La restricción (1.9) establecen el rango de variación de las variables p_i. Por último, las restricciones (1.10) y (1.11) establecen el tipo de las variables.

Las restricciones de eliminación de subciclos nacen en las primeras formulaciones del TSP debido a que, a pesar a las restricciones de ruta y de conservación de flujo, no es capaz de evitar que en la ruta aparezcan subciclos que no incluyan a los nodos de inicio y final. Por ello, se modelaron restricciones que evitaran estos subciclos como, por ejemplo, las restricciones de *Dantzig, Fulkerson, and Johnson (1954)* quienes plantearon que la ruta buscada debe entrar y salir de cualquier subconjunto S del conjunto de nodos a visitar y que no contenga a los nodos de inicio y final. Estas restricciones se pueden plantear de alguna de las siguientes maneras:

$$\sum_{i \in S, j \notin S} x_{ij} \geq 2;$$

$$\sum_{i \in S, j \notin S} x_{ij} \leq |S| - 1 :$$

Estas restricciones son las más usadas para evitar subciclos aunque sean del orden de 2^n, donde n representa el número de nodos. Se introducen a medida que aparecen en las rutas los posibles subciclos. Como alternativa a estas restricciones, y aunque son menos usuales en la literatura moderna, están las conocidas como restricciones MTZ que fueron propuestas por *Miller, Tucker, and Zemlin (1960)* y son del orden de n^2.

$$u_j \geq u_i + 1 + (n-1)(1 - x_{ij}), \forall i, j$$

Además, la variable u_i que usan estas restricciones dan el orden de los nodos que aparecen en la ruta [1].

1.2.4. Extensiones

En esta subsección describimos las extensiones o versiones del problema obtenidas al tener en cuenta otros aspectos de las aplicaciones reales no contempladas en el modelo básico de la optimización de una ruta de recogida de residuos del apartado anterior. Las extensiones más comunes del OP son las denominadas en la literatura por OPTW (*Orienteering Problem with Time Windows*) y TOP (*Team Orienteering Problem*). El OPTW surge al considerar la limitación de la recogida de algunos contenedores a ciertos intervalos de tiempo (las denominadas *Time Windows*) mientras que el TOP surge al considerar una flota (el equipo o *Team*) fija de vehículos en lugar de uno solo. Otras extensiones ya propuestas en la literatura para el OP son relevantes para su aplicación en la recogida de residuos.

Orienteering Problem with Time Windows (OPTW)

En las aplicaciones prácticas reales frecuentemente hay limitaciones sobre el momento en el que el vehículo de recogida puede pasar por determinados puntos de recogida. El Orienteering Problem with Time Window (OPTW) se deriva del OP al añadir restricciones en forma de ventanas de tiempo en algunos puntos, es decir, solo podremos visitar tales nodos dentro del intervalo de tiempo correspondiente que se exige.

Para establecer el modelo matemático del OPTW usaremos los parámetros y variables que aparecen en el OP añadiendo las ventanas de tiempo como nuevos

parámetros y una nueva variable que nos dará el tiempo en el que se visita cada punto. La función objetivo para este modelo sigue siendo la misma que en el modelo anterior. Por ello, para evitar repetirnos y volver a reescribir todo el modelo, a continuación solo mostraremos lo que debemos añadir al modelo del OP para obtener el nuevo modelo con ventanas de tiempo.

Consideramos los siguientes nuevos parámetros:

$$a_i : \text{Inicio de la ventana de tiempo del punto } i \qquad \forall i = 0, ..., m+1 \tag{1.12}$$

$$b_i : \text{Final de la ventana de tiempo del punto } i \qquad \forall i = 0, ..., m+1 \tag{1.13}$$

Para los casos especiales del punto inicial, $i = 0$ y final, $i = m+1$ se consideran las ventanas dadas por los valores $a_0 = b_0, a_{m+1} = 0$ y $b_{m+1} = T_{max}$.

Las nuevas variables para este modelo son las siguientes:

$$T_i = \text{Tiempo en el que se accede al nodo } i \text{ en la ruta,} \qquad \forall i = 0, ..., m+1 \tag{1.14}$$

Las restricciones que debemos añadir al OP para obtener el modelo matemático para el OPTW son las siguientes:

$$T_i + t_{ij} + T_j \leq M(1 - x_{ij}), \qquad \forall i, j = 0, .., m+1 \tag{1.15}$$

$$a_i \leq T_i \leq b_i, \qquad \forall i = 0, .., m+1 \tag{1.16}$$

$$T_i \geq 0 \qquad \forall i = 0..., m+1 \tag{1.17}$$

La restricción (1.15) determina la relación entre el tiempo de acceso a cada punto de la ruta y al siguiente, mientras que la restricción (1.16) limita el tiempo de acceso a cada punto a la correspondiente ventana. Por último, las restricciones (1.17) establecen el rango de variación de las nuevas variables.

Team Orienteering Problem (TOP)

La extensión del problema básico (OP) al considerar una flota fija de vehículos en lugar de uno solo es conocida como TOP (*Team Orienteering Problem*). El término *Team* hace referencia al equipo que forman los participantes el OP que se corresponden con los vehículos.

Para formular este problema existen varias alternativas. Una de las más directas consiste en considerar un nuevo índice l correspondiente a los vehículos. Se consideran las siguientes variables para definir las posibles rutas:

$$x_{ij}^l = \begin{cases} 1 & \text{si la } l\text{-ésima ruta va de } i \text{ a } j \\ 0 & \text{en otro caso} \end{cases}, \ \forall i, j = 0, ..., m+1, \forall l$$

$p_i^l =$ posición del nodo i en la l-ésima ruta, $\forall i = 0, ..., m+1, \forall l$

El objetivo de maximizar el valor total de las rutas se establece ahora por:

$$\text{Max} \sum_l \sum_{i=1}^{m} \sum_{j=1}^{m+1} v_i x_{ij}^l \tag{1.18}$$

Las restricciones que deben cumplir las variables para que correspondan a una solución factible son las siguientes:

$$\sum_{j=1}^{m+1} x_{0j}^l = \sum_{i=0}^{m} x_{i(m+1)}^l = 1, \ \forall l \tag{1.19}$$

$$\sum_{i=0}^{m} x_{ik}^l = \sum_{j=1}^{m+1} x_{kj}^l \leq 1, \ \forall k = 1, .., m, \forall l \tag{1.20}$$

$$\sum_{i=0}^{m} \sum_{j=1}^{m+1} t_{ij} x_{ij}^l \leq T_{max}, \ \forall l \tag{1.21}$$

$$p_i^l - p_j^l + 1 \leq m(1 - x_{ij}^l), \ \forall i, j = 0, ..., m+1, \forall l \tag{1.22}$$

$$1 \leq p_i^l \leq m, \ p_i^l \in \mathbb{Z}, \ \forall i = 0, ..., m+1, \forall l \tag{1.23}$$

$$x_{ij}^l \in \{0, 1\}, \ \forall i, j = 0, ..., m+1, \forall l \tag{1.24}$$

Estas restricciones mantienen el mismo significado que las restricciones del OP pero referidas a cada una de las rutas de cada vehículo en lugar de a la única ruta considerada en el OP.

Otra extensión interesante surge cuando se considera un horizonte de planificación de varios días donde los vehículos realizan una ruta cada día. Para el caso de un solo vehículo, el problema se puede modelizar también como el TOP si los parámetros no cambian de una día para otro; en lugar de tener varios vehículos en un mismo día, tendríamos un mismo vehículo varios días. Este extensión del OP surge en la aplicación real considerada en este trabajo pero no puede utilizarse el modelo de programación lineal anterior con el índice l porque los valores de los puntos cambian cada día dependiendo de la solución del día anterior.

Otras extensiones

Existe una gran variedad de extensiones del OP que han recibido diferente grado de atención. Una de las más relevantes es la que incorpora las dos extensiones consideradas anteriormente dando lugar al problema conocido como *Team Orienteering Problem with Time Windows* (TOPTW). La formulación de este problema se obtiene de la del TOP extendiendo los parámetros y variables añadiendo el índice l.

En el contexto de la recogida de residuos, una extensión importante surge al considerar la obligación de recoger algunos contenedores. En el OP, si $J \subseteq I$ es el conjunto de índices de los contenedores que tienen que recogerse obligatoriamente, se introduce las restricciones siguientes.

$$\sum_{j \in I+\{F\}} x_{ij} = 1, \ \ \forall i \in J \tag{1.25}$$

Además de las extensiones comentadas anteriormente en [11] podemos encontrar una descripción detallada de una gran variedad de otras extensiones del OP como son: *Arc Orienteering Problem* (AOP) , *Capacitated Team Orienteering Problem* (CTOP), *Orienteering Problem with variable profits* (OPVP), etc.

Por su interés en los problemas de recogida de residuos cabe mencionar el problema conocido como *Time Dependent Orienteering Problem* (TDOP) el cual no considera el tiempo de viaje entre nodos como un valor constante. Esto nos acerca a un problema más parecido a la realidad ya que el tiempo que se tarda de viajar de un nodo a otro puede verse afectado en la realidad por distintos factores como el tráfico, zonas de construcción, etc. En relación a este tema, también encontramos una extensión conocida como *Stochastic Orienteering Problem* (SOP) en la cual se consideran los tiempos de viajes casi imposibles de predecir de forma determinista (como suele pasar en la realidad).

En particular, el *OP with Stochastic Profits* (OPSP), tiene como objetivo maximizar la probabilidad de que la puntuación total obtenida (el beneficio en nuestro caso) sea mayor que un valor objetivo predefinido (K) sin violar la restricción de tiempo límite (T_{max}). En el OPSP, los beneficios de cada nodo i son aleatorios con una distribución normal conocida (S_i). Dado un beneficio se formula la función objetivo de la siguiente manera:

$$\text{Max}\,\text{Pr}\left(\sum_{i=1}^{m}\sum_{j=1}^{m+1} S_i x_{ij} \geq K\right) \tag{1.26}$$

2

Capítulo 2. Experimentación.

En este capítulo nos centraremos en la aplicación práctica de los modelos descritos en el capítulo anterior a las instancia clásicas como a un caso real. En la primera sección encontraremos la implementación tanto de los programas, OP y OPTW, como de los datos que usaremos. En la segunda sección, encontraremos los resultados de la ejecución de ambos programas con los datos descritos anteriormente y, por último, en la tercera sección se hará una descripción de los datos utilizados y los resultados obtenidos en el estudio de un caso real. Recordemos que el problema del Orienteering Problem tiene como finalidad establecer la ruta óptima que maximice el beneficio con una limitación de tiempo y ese será nuestro principal objetivo en el caso real.

El caso real que estudiaremos será el de la recogida de residuos en la isla de La Palma. Disponemos de los datos históricos de la recogida de los contenedores de dos tipos de residuos reciclable: la fracción azul que consiste en residuos de papel y cartón, y la fracción amarilla constituido los envases de plástico. Se conocen las ubicaciones de los contenedores y, a partir de los datos históricos durante años del volumen de llenado de cada contenedores en el día de su recogida se han obtenido unas tasas de llenado. Las tasas de llenado indican la cantidad media diaria de residuo que se deposita en cada contenedor y con ello determinamos el nivel de llenado de cada contenedor multiplicando la tasa de llenado por el número de días que han transcurrido desde la recogida anterior. Los beneficios que queremos maximizar consistirán, en este caso, de los niveles de llenado de los contenedores. Por tanto, lo que nos interesa es recoger aquellos contenedores que tengan el mayor nivel de llenado. Además, se deberá cumplir ciertas condiciones de tiempo y de ruta ya que se tendrá un tiempo límite (por el horario de los conductores de los camiones que realizarán las rutas) y sólo podrá recogerse como mucho una vez cada contenedor.

2.1. Implementación en GUSEK

Para la resolución de los distintos modelos matemáticos se ha usado el programa *GUSEK (GLPK Under Scite Extended Kit)* el cual nos proporciona una interfaz gráfica del programa *GLPK (GNU Linear Programming Kit)*, cuya función es la de resolver problemas de programación lineal y otros relacionados, gracias a la integración de este último a un editor de texto denominado *scite*. Existen otros programas como *XPRESS*, *Gurobi* o *CPLEX* los cuales son más potentes y pueden resolver en menor tiempo problemas con una gran cantidad de variables. Se ha decidido la utilización del GUSEK por, entre otras cuestiones, el conocimiento previo de dicho programa.

2.1.1. Programas

En esta subsección, mostraremos la implementación de los modelos matemáticos en el programa GUSEK con el objetivo de comprobar si el programa nos permite obtener buenas soluciones para las instancias que hemos seleccionados, y además, dichos resultados nos permitirán valorar el esfuerzo que requiere GUSEK para hallar las soluciones. Los modelos se dividen de la misma forma: en primer lugar encontramos la declaración de los parámetros, en segundo lugar tenemos la declaración de las variables, y por último, la función objetivo y las restricciones del problema.

Mostramos en las figuras 2.1, 2.2, 2.3 y 2.4 la implementación del problema OP para GUSEK. Para el OPTW y otras extensiones solo mostramos las sentencias que hay que añadir al anterior en las figuras 2.5, 2.6 y 2.7.

Orienteering Problem

La figura 2.1 muestra el código con las especificaciones de los parámetros para el problema. Se incluye el parámetro m del número de contenedores, el conjunto de índices $I = \{0, 1, ..., m+1\}$ y el parámetro T_{max} de duración máxima de las rutas. Usando este conjunto se definen las series de parámetros: $t_{ij}, i, j \in I$, de los tiempos de viaje entre los puntos i y j, y, finalmente, los beneficios v_i, $i \in I$ de cada contenedor i.

Como podemos observar en el código de la figura 2.1 solo hemos usado un conjunto de índices $I = 0..m + 1$ para todos los puntos, incluyendo los m puntos con contenedores y además el origen 0 y final $m + 1$ de las rutas. Otra opción consiste en utilizar varios conjuntos que representen todas las situaciones

```
/*****************PARAMETERS*****************/

param m, integer,  <= 1;
/* Number of containers */

set I := 0..m+1;
/* Set of containers where 0 is the origin and m+1 is the final */

param Tmax, >=0;
/* Time max to pick up all container */

param t { i in I, j in I }, >=0;
/* Time travel from node i to j */

param v { i in I }, integer, >=0;
/* Benefit of node i */

/*****************END OF PARAMETERS****************/
```

Figura 2.1: Parámetros del OP

que podamos encontrarnos al formular el problema como pueden ser conjuntos en donde se exceptúen solo el punto de inicio, el punto final o ambos. Si, por ejemplo, queremos sumar todos los puntos menos el punto de origen, en vez de trabajar con sentencias como $i \in I : i \neq 0$, se pueden cambiar por sentencias donde los nodos pertenezcan a un solo conjunto (que no contenga el nodo de origen) como sería $I1 = 1..m + 1$ en este caso. Así, tendremos sentencias como $i \in I1$ que son más compactas pero menos intuitivas. En la figura 2.2 podemos ver una muestra de como sería la implementación del OP si nos decantáramos por la opción de utilizar varios conjuntos de distintos elementos que se ajusten a nuestras necesidades reales.

A continuación, en la figura (2.3) mostramos el código de la sección correspondiente a las variables. Aparecen las variables binarias x_{ij}, $i, j \in I$ y las variables enteras p_i. $i \in I$. Si la variable x_{ij} toma el valor 1 significa que el vehículo va del punto i al punto j sin pasar por ningún punto intermedio, y en cualquier otro caso toma el valor 0. El valor de la variable p_i representa la posición u orden del punto i en la ruta. En estos códigos solo hemos usado un conjunto de nodos como se muestra en la figura (2.1).

Finalmente, en la figura (2.4), mostramos el código del modelo matemático del problema OP para el programa GUSEK. Aparece en primer lugar la función

```
/*********************PARAMETERS*********************/

set I := 0..m+1;
/* Set of points consisting of the containers 1..m, */
/* and also the origin 0 and the final m+1          */

set I1 := 1..m;
/* Set of points with the containers                */

set I2 := 1..m+1;
/* Set of points with containers and the final      */

set I3 := 0..m;
/* Set of points with containers and the origin     */

/*****************END OF PARAMETERS*****************/
```

Figura 2.2: Parámetros con varios conjuntos

```
/************************VARIABLES************************/

var x {i in I, j in I }, binary;
/* x[i,j] = 1 means that the vehicle goes from node i to j */
/* x[i,j] = 0 in other case                                */

var p {i in I}, integer;
/* p[i] position of node i in path                         */

/*********************END OF VARIABLES*********************/
```

Figura 2.3: Variables del OP

objetivo lineal a maximizar consistente en la suma de los beneficios de los puntos de recogida incluidos en la ruta siguiendo la fórmula (1.4). A continuación, aparecen (precedidas por las iniciales "s.t." correspondientes a *such that*) las restricciones descritas en las ecuaciones (1.5) a (1.9). En particular, aparece la restricción (1.5) descompuesta en dos ecuaciones, una para el comienzo de la ruta y otra para el final. La restricción (1.6) también aparece separada por un lado la conservación de flujo en cada punto y, por otro, la limitación de selección del contenedor a visitar a lo sumo una vez cada punto. Siguen las restricciones (1.7)

del tiempo máximo de la ruta y la restricción (1.8) de eliminación de subruta. Y finalmente, aparece la restricción (1.9) del límite de la variable de posición.

```
/**********************CONSTRAINS**********************/

maximize benefit:
sum{i in I, j in I : i!=0 and i!=m+1 and j!=0} v[i]*x[i,j];
/* The objetive function (maximise total benefit)       */

s.t. StartEnd:
sum { j in I : j!=0} x[0,j] = 1;
/* Guarantee that route starts in node 0                */

sum{ i in I : i!= m+1} x[i,m+1] = 1;
/* Guarantee that route ends in node m+1                */

s.t. cons{d in I: d!=0 and d!=m+1} :
sum{ i in I : i!=m+1 } x[i,d] = sum { j in I : j!=0 } x[d,j] ;
/* Make sure the connectivity of route                  */

s.t. select{d in I: d!=0 and d!=m+1}:
sum { j in I : j!=0 } x[d,j] <= 1;
/* Guarantee that every node is visited at most once    */

s.t. timemax1 : sum{i in I , j in I: i!=m+1 and j!=0}
t[i,j]*x[i,j] <= Tmax;
/* Limited of time                                      */

s.t. position{i in I}:
1 <= p[i] <= m+1;
/* Limited postion in the route                         */

s.t. sub{i in I, j in I} :
p[i] - p[j] + 1 <= (m)*(1 - x[i,j]);
/* Subtour elimination constraints                       */

/******************END OF CONSTRAINS*****************/
```

Figura 2.4: Restricciones del OP

Con esto terminamos la implementación del programa OP en GUSEK.

Orienteering problem with time windows

Para el programa con ventanas de tiempo solo mostraremos el código adicional que tenemos que añadir al código que se muestra en las figuras (2.5), (2.6) y (2.7) del OP para obtener el programa con ventanas de tiempo. Estos códigos hacen referencia a los nuevos parámetros, variables y restricciones respectivamente. Como nuevos parámetros tenemos el intervalo de tiempo $[a_i, bi]$ para cada punto de recogida i y las nuevas variables T_i que establecen el tiempo en el que se accede a cada uno de los puntos i, $i \in I$.

```
/*****************NEW PARAMETERS FOR OPTW*****************/

param M, >0;   /* A large constant                         */

param a{i in I}, integer , >=0;
param b{i in I}, integer , >=0;
/* Parameters for the interval of time (a,b) in i          */

/*****************END OF NEW PARAMETERS*****************/
```

Figura 2.5: Nuevos parámetros para el OPTW

```
/******************NEW VARIABLES FOR OPTW*****************/
var T{i in I}, >=0;
/* Time to acces to node i */

/*****************END OF NEW VARIABLES*****************/
```

Figura 2.6: Nuevas variables para el OPTW

Las nuevas restricciones para el OPTW descritas en las ecuaciones (1.15) y (1.16) aparecen en la figura (2.7). La primera describe la relación que existe entre el tiempo de salida de un nodo con el tiempo de acceso del siguiente nodo en la ruta. Por otro lado, la segunda limita el tiempo de acceso al nodo i con la correspondientes ventanas de tiempo.

```
/*********************NEW CONSTRAINS*********************/

s.t. acces{i in I , j in I}: T[i] + t[i,j] - T[j] <= M*(1 - x[i,j]);
/* Ensure the timeline of route */

s.t. timewindow{i in I}: a[i] <= T[i] <= b[i];
/* Restrict time to acces to node i */

/******************END OF NEW CONSTRAINS*****************/
```

Figura 2.7: Nuevas restricciones para el OPTW

```
/*****************ADDITIONAL SENTENCES***************/

printf {i in I, j in I:  i!=m+1 and x[i,j]=1 }
"Nodes of path x[%d,%d] \n",i,j;

printf "Benefit: %d \n",sum{i in I, j in I : i!=0 and i!=m+1 and j!=0}

/*************END OF ADDITIONAL SENTENCES************/
```

Figura 2.8: Sentencias adicionales

Finalmente añadimos las sentencias que observamos en el código de la figura (2.8) antes de terminar ambos programas para que nos devuelvan por consola la ruta y el objetivo que han obtenido los programas.

2.1.2. Datos

En esta sección describimos los datos utilizados en la experimentación. Corresponden a casos o instancias usuales en los estudios de los problemas de OP y sus extensiones y un caso real de recogida de residuos para reciclaje en la isla de La Palma.

Para las pruebas del OP y sus variantes usamos las instancias o casos usuales en los experimentos publicados en la literatura especializada y que proporciona el grupo del profesor Vansteenwegen de la Universidad Católica de

Lovaina a través de su página web[1]. Las primeras instancias que hemos usados para el OP son las llamadas de *Tsiligirides* [19]. En particular, usamos las instancias de 21 y 32 puntos que podemos encontrar en *Tsiligirides 2* y *Tisiligirides 3* respectivamente . Los archivos de texto de estas instancias contienen los datos que se describen a continuación. En la primera linea encontramos dos valores que corresponden al tiempo que se dispone por trayecto (T_{max}) y el número de rutas respectivamente (NR) (en nuestro caso siempre será una ruta). A continuación, encontramos una línea por cada punto del problema en la que aparecen tres valores que representan, en el siguiente orden, las coordenadas x e y del punto, y el beneficio del mismo. Los datos descrito anteriormente se puede encontrar en la tabla 2.1 y 2.2.

Datos Tsiligirides 2		
Coordenada x	Coordenada y	Beneficio
4.6	7.1	0
5.0	5.6	0
5.7	11.4	20
4.4	12.3	20
2.8	14.3	30
3.2	10.30	15
3.5	9.8	15
4.4	8.4	10
7.8	11.0	20
8.8	9.8	20
7.7	8.2	20
6.3	7.9	15
5.4	8.2	10
5.8	6.8	10
6.7	5.8	25
13.8	13.1	40
14.1	14.2	40
11.2	13.6	30
9.7	16.4	30
9.5	18.8	50
4.7	16.8	30

Tabla 2.1: Datos de la instancia Tsiligirides 2 para 21 nodos

Por otro lado, en esta misma dirección encontramos las instancias para el problema con ventanas de tiempo. Las que hemos usado son las instancias de *Vansteenwegen pr01* usadas en [20], aunque todas ellas se basan en las instancias de Solomon (1987) clásicas en los problemas de rutas. Los archivos de texto que

[1] www.mech.kuleuven.be/en/cib/op#section-1

Datos Tsiligirides 3		
Coordenada x	Coordenada y	Beneficio
19.1	24.3	0
18.2	24.0	0
12.6	24.9 2	0
14.4	28.0	20
16.9	28.1	20
20.7	28.2	20
12.5	26.6	20
21.8	27.3	20
12.5	22.6	20
22.5	17.0	30
19.9	15.0	30
14.9	15.1	30
11.5	18.6	30
12.4	29.8	30
17.8	28.1	30
9.1	29.8	40
10.0	32.6	40
13.9	33.1	40
19.95	10.3	40
15.2	8.0	40
14.7	31.2	50
7.4	36.5	50
21.0	25.5	50
18.0	25.3	10
19.5	24.7	10
21.4	21.8	10
16.0	21.4	10
18.65	26.2	10
17.9	28.9	10
14.3	19.9	20
17.0	19.0	20
10.80	21.0	20

Tabla 2.2: Datos de la instancia Tsiligirides 3 para 32 nodos

contienen las instancias se organizan de igual manera que en el apartado anterior pero en las lineas correspondientes a los puntos del problema además, aparecen dos términos nuevos: el valor de apertura del intervalo de tiempo y el valor del cierre de dicho intervalo. De estos datos sólo hemos usados los intervalos de tiempo que necesitamos para añadir al OP y que podemos encontrar en la tabla 2.3 para los 21 nodos. Para el caso de 32 nodos simplemente debemos añadir a la tabla anterior los intervalos que encontramos en la tabla 2.4

Intervalos de tiempo para los 21 nodos		
Nodo	Apertura (a)	Cierre(b)
0	0	1000
1	354	509
2	234	401
3	411	573
4	474	622
5	155	295
6	361	509
7	451	629
8	425	588
9	72	199
10	157	318
11	296	444
12	111	249
13	368	528
14	98	251
15	96	208
16	382	535
17	436	580
18	405	528
19	255	414
20	293	470

Tabla 2.3: Intervalos de apertura y cierre para los 21 nodos

Intervalos de tiempo para los 32 nodos		
Nodo	Apertura (a)	Cierre(b)
21	298	408
22	479	607
23	376	536
24	91	238
25	360	505
26	379	528
27	258	428
28	352	509
29	288	380
30	159	324
31	423	564

Tabla 2.4: Intervalos de apertura y cierre para los 32 nodos

La figura 2.9 muestra la implementación de estos datos en GUSEK tanto para el OP como para el OPTW para el problema con 21 nodos.

```
param m := 19;
param Tmax := 20;
param: v:=
    0   0
    1   0
    2   20
    3   20
    4   30
    5   15
    6   15
    7   10
    8   20
    9   20
    10  20
    11  15
    12  10
    13  10
    14  20
    15  40
    16  40
    17  30
    18  30
    19  50
    20  30;
```

(a) Datos para el OP

```
param M :=400;
param: a:=          param: b:=
    0   0           0   1000
    1   354         1   509
    2   234         2   401
    3   411         3   573
    4   474         4   622
    5   155         5   295
    6   361         6   509
    7   451         7   629
    8   425         8   588
    9   72          9   199
    10  157         10  318
    11  296         11  444
    12  111         12  249
    13  368         13  528
    14  98          14  251
    15  96          15  208
    16  382         16  535
    17  436         17  580
    18  405         18  528
    19  255         19  414
    20  293         20  470;
```

(b) Datos para el OPTW

Figura 2.9: Datos para el OP y el OPTW problemas con 21

El parámetro m es el número de nodos, T_{max} es el tiempo límite, v representa el beneficio y el parámetro t la distancia que existe entre cada par de nodos, estos parámetros son comunes para ambos problemas. Además, para el OPTW debemos añadir M para el valor de la constante en la restricción de eliminación de subruta, a para la apertura del intervalo de tiempo y b para la clausura de dicho intervalo. Ambas instancias nos ofrecían las coordenadas cartesianas de cada punto, así que para establecer la distancia entre los distintos pares de nodos hemos usado la *distancia euclídea* que, para un espacio bidimensional con coordenadas $X(x_1, x_2)$ e $Y(y_1, y_2)$ respectivamente, se calcula por:

$$d_E(X, Y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Para el cálculo de estas distancias hemos creado un pequeño programa en C que podemos encontrar en el apéndice de este trabajo A.1. Después de ingresar

las coordenadas de cada punto nos devuelve un archivo con las distancias ya calculadas.

Las distancias para cada par de puntos dependerá del valor de las coordenadas en cada instancia, por ello, sólo mostraremos a continuación la distancia entre cada par de puntos para el problema OP. La implementación de las distancias para el problema con ventanas de tiempo tiene el mismo aspecto sólo que variará el valor de las distancia entre los puntos. En la figura 2.10 podemos encontrar parte de las distancias utilizadas en el OP ya que, debido al gran tamaño de los datos, hemos optado por mostrar solo el principio de estas.

```
param t: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 :=
0 0.00 1.55 4.44 5.20 7.42 3.49 2.91 1.31 5.04 4.99 3.29 1.88...
1 1.55 0.00 5.84 6.73 8.97 5.03 4.46 2.86 6.08 5.66 3.75 2.64...
2 4.44 5.84 0.00 1.58 4.10 2.73 2.72 3.27 2.14 3.49 3.78 3.55...
3 5.20 6.77 1.58 0.00 2.56 2.33 2.66 3.90 3.64 5.06 5.26 4.79...
4 7.42 8.97 4.10 2.56 0.00 4.02 4.55 6.11 5.99 7.50 7.82 7.29...
5 3.49 5.03 2.73 2.33 4.02 0.00 0.58 2.25 4.65 5.62 4.97 3.92...
6 2.91 4.46 2.72 2.66 4.55 0.58 0.00 1.66 4.46 5.30 4.49 3.38...
7 1.31 2.86 3.27 3.90 6.11 2.25 1.66 0.00 4.28 4.62 3.31 1.96...
.
.
.
20 9.70 11.20 5.49 4.51 3.14 6.67 7.10 8.40 6.58 8.11 9.11 9.04...;
```

Figura 2.10: Distancias parciales del OP

2.2. Resultados

En esta sección mostramos los resultados obtenidos con la implementación del OP y el OPTW para instancias usadas en la literatura. Se ha utilizado un ordenador con un procesador Intel(R) Core(Tm) i5-2410M CPU @ 2.30GHz 2.30GHz y una memoria RAM de 4 GB para la ejecución de los distintos programas.

2.2.1. Resultados para el OP

Una vez implementado el Orienteering Problem en GUSEK ejecutamos el programa usando el archivo de datos de 21 nodos probando con diferente tiempos límites (T_{max}) y obteniendo así los distintos archivos de salida con los distintos resultados.

En la tabla 2.5, para el programa de 21 nodos y los distintos valores de T_{max}, se tienen en las tres primeras columnas el tiempo límite, el valor objetivo obtenido y la ruta seleccionada, respectivamente. Mientras que las cuatro últimas columnas hacen referencia al tiempo y memoria que se ha usado para conseguir los resultados en cada ejecución. El tiempo y memoria (1) hace referencia a los resultados obtenidos con los parámetros dónde se hace uso de varios conjuntos como se muestra en la figura 2.2 mientras que el tiempo y memoria (2), hacen referencia al programa dónde sólo se hacía uso de un conjunto de nodos como observamos en 2.1.

Orienteering Problem 21 nodes						
T_{max}	Obj	Route	Time		Memory	
			(1)	(2)	(1)	(2)
15	130	0-7-6-5-3-4-16-20	1.4	1.3	2.1	2.1
20	195	0-13-11-10-9-8-2-3-4-16-20	1.2	1.6	2.3	2.3
25	240	0-7-12-11-13-14-10-9-8-2-3-4-16-20	19.9	21.0	4.1	4.1
30	270	0-12-7-13-14-11-10-9-8-2-5-6-3-4-16-20	120.0	143.5	15.7	15.7

Tabla 2.5: Resultados comparativos del OP para 21 nodos usando varios conjuntos de índices (1) o solo uno (2)

Así, por ejemplo, encontramos que para los 19 nodos, más los nodos de inicio y final, y para un valor $T_{max} = 20$ se ha obtenido un beneficio de 195 y la ruta obtenida es:

$$0 - 13 - 11 - 10 - 9 - 8 - 2 - 3 - 4 - 16 - 20$$

La ruta viene determinada por la posición de los nodos i, $p[i]$, empezando en el nodo 0 (nodo inicial) y acabando en el nodo 20 (nodo final), finalmente, el resto de nodos que aparecen en la ruta son aquellos cuyo valor de posición es distinto de 1 y 19 ordenados de menor a mayor.

Podemos ver que la memoria usada en ambos casos es la misma y los tiempos son muy parecidos siendo el programa que usa un sólo conjunto un poco más lento. A partir de ahora sólo trabajaremos con el programa que hace

uso de un único conjunto de nodos ya que es más fácil a la hora de leer debido a que cada sentencia tiene sus propias especificaciones.

Para el mismo problema pero usando los datos de la instancia con 32 nodos obtenemos los resultados que encontramos en la tabla 2.6.

Orienteering Problem 32 nodes				
T_{max}	Obj	Route	Time	Memory
6	10	0-27-31	0.1	2.6
10	35	0-28-27-26-31	0.1	4.1
15	50	0-27-20-21-22-26-31	1.4	6.3
20	75	0-27-20-21-22-23-25-26-31	0.8	5.4
25	95	0-9-10-11-12-21-22-23-25-26-31	4.0	6.3
30	115	0-28-27-20-19-12-21-22-23-24-25-26-31	13.6	6.5
35	135	0-6-7-3-2-8-10-11-12-21-22-23-25-26-27-31	15.0	8.7
40	155	0-28-18-6-7-3-2-8-10-11-12-21-22-23-24-25-26-31	33.7	10.5
45	170	0-28-18-6-7-3-2-8-9-10-11-12-21-22-23-24-25-26-27-31	20.9	8.6

Tabla 2.6: Resultados del OP para 32 nodos

Podemos observar como en ambos casos, con 21 o 32 nodos, a medida que aumenta el tiempo límite aumenta la ruta seleccionada ya que al disponer de más tiempo podemos visitar más puntos. En general, cuánto mayor es la ruta más tiempo y memoria se requiere para determinar la misma y esto es uno de los elementos que tendremos que tener en cuenta en nuestro problema real ya que, además de obtener la mejor ruta, nos interesa también que el tiempo que se tarda en encontrar dicha ruta sea un tiempo razonable.

Una de las diferencias entre ambos resultados es el objetivo para cada T_{max} ya que para el problema de 21 nodos se alcanzan valores mucho mayores que para el problema con 32 nodos. Por ejemplo, si observamos los resultados para $T_{max} = 20$ tenemos que para 21 nodos se maximiza la función objetivo con un valor de **195**, mientras que para el caso de 32 nodos con el mismo valor de tiempo límite, obtenemos un valor de **75**. Por último, destacar que el problema de 32 es ejecutables con mayores valores para T_{max}.

2.2.2. Resultados para el OPTW

Ejecutando el problema con ventanas de tiempo con el conjunto de 21 nodos hemos logrados los siguientes resultados que encontramos en la tabla 2.7. Mientras que para el problema de ventanas de tiempo con 32 nodos tenemos los resultados que apreciamos en la tabla 2.8.

Orienteering Problem Time Window 21 nodes				
T_{max}	Obj	Route	Time	Memory
10	30	0-7-3-20	0.1	1.9
15	95	0-12-7-6-3-16-20	3.4	2.7
20	140	0-14-1-10-2-3-16-20	3.7	2.5
25	175	0-14-10-9-2-5-6-3-16-20	25.9	4.3
30	205	0-12-14-10-9-8-11-13-7-6-3-16-20	155.1	8.8

Tabla 2.7: Resultados del OPTW con 21 nodos

Orienteering Problem Time Window 32 nodes				
T_{max}	Obj	Route	Time	Memory
10	35	0-28-27-26-31	0.3	4.5
15	50	0-27-20-21-22-26-31	7.2	6.4
20	75	0-27-20-21-22-23-25-26-31	3.4	5.9
25	90	0-28-27-20-19-21-22-23-25-26-31	20.4	6.9
30	100	0-27-20-19-11-10-21-22-23-25-26-31	76.1	10.3
35	110	0-12-11-10-21-19-27-20-22-23-25-26-31	313.2	16.8

Tabla 2.8: Resultados del OPTW con 32 nodos

Los resultados obtenidos son muy parecidos a los que ya obtuvimos con el problema sin ventanas de tiempo, pero en esta caso quizás más restrictivo. Observamos que los tiempos límites ejecutables son menores en ambos casos, siendo el problema de 32 el que se puede ejecutar con un T_{max} mayor. Al igual que pasa con el número de nodos en cada ruta los cuáles son menores en ambos casos. En cuánto al valor objetivo, el problema con 21 nodos sigue obteniendo mejores resultados para un tiempo límite en comparación con el problema de 32 nodos.

2.3. Caso real de La Palma

En esta sección describimos los resultados obtenidos usando el programa OP implementado en GUSEK para obtener rutas de recogida de un camión de recogida de residuos para reciclaje en una situación real usando los datos citados en [Expósito-Márquez et al.]. En este caso, conocemos la localización exacta de cada contenedor de residuos reciclables, papel y envases, de la isla de La Palma. Son estos puntos de recogida donde se localizan los contenedores que harán el papel de nodos en nuestro programa. Se dispone de algo más de 300 puntos de recogida donde está localizados contenedores de ambos tipos de residuos. De cada contenedor se conoce la tasa de llenado que representa la cantidad de residuo que

se deposita al día, en términos medios. Esto permite estimar el nivel de llenado de los contenedores conociendo el número de días que han transcurridos desde la última recogida del contenedor.

Además, debido a que la isla cuenta con vehículos de recogida específicos para cada tipo de contenedor,es decir, solo los vehículos de recogida de papel podrán recoger los contenedores del mismo tipo de residuo, hemos decidido estudiar rutas para solo un tipo de residuo. Para el otro tipo de residuos se procederá de manera análoga. Por ello, solo estudiaremos las rutas en las que intervengan residuos de papel.

Debido a que GUSEK no es lo suficientemente potente como para ejecutar el modelo con todos los puntos de los que disponemos, antes habrá que hacer una selección de un subconjunto de puntos con el tamaño adecuado para que pueda ejecutarse.

2.3.1. Subconjunto de nodos aleatorios.

Para elegir los 21 puntos que necesitamos realizamos una selección de manera sistemática entre el total de contenedores de papel. Para seleccionar dichos nodos seguimos los siguientes pasos:

1. Elegimos un número i que marcará la diferencia entre dos nodos consecutivos. Para obtener el subconjunto de tamaño correcto elegimos i como : $i = \frac{N}{m}$, dónde N será el tamaño del conjunto de puntos de recogida (aproximadamente 300) y m el tamaño del subconjunto que queremos obtener ($m = 19$).

2. Entre los más de 300 posibles nodos elegimos un nodo de partida r de manera aleatoria, así, nuestro subconjunto a estudiar será: $I = \{r, r + i, r + 2i, ..., r + 18i\}$. Recordemos que a este subconjunto de 19 nodos tenemos que añadirle los nodos de inicio y final que sumarán en total 21 nodos al subconjunto I.

Procediendo de la manera descrita anteriormente obtenemos como nodos los contenedores que aparecen en la tabla 2.9, con la siguiente información: la primera columna representa el nodo o punto de recogida i, la segunda representa el contenedor elegido, y en la tercera se indica el tipo de residuo reciclable. Las columnas cuarta y quinta contienen las coordenadas x e y respectivamente, que determinan su posición, y en la sexta columna se encuentra el número de contenedores (NC). Por último, la séptima columna es la tasa de llenado de cada contenedor.

En algunos casos, debido a la alta demanda, se han colocado en el mismo lugar más de un contenedor, por eso, en la columna que hace referencia al número de contenedores pueden aparecer valores mayores que 1 ,como en el caso que se refleja en la tabla 2.9 donde podemos ver que en el nodo 12 aparecen 3 contenedores. Para facilitar el estudio supondremos que para cada nodo solamente se tiene un solo contenedor.

Nodos	Contenedores	Tipo	Coord. X	Coord. Y	NC	Tasa llenado
0	0	planta	28.645233	-17.8990968	1	0.0000
1	2051	Papel	28.501715	-17.8608061	1	0.1975
2	2073	Papel	28.6340103	-17.770935	1	0.2403
3	2098	Papel	28.6675544	-17.7864988	1	0.2599
4	2118	Papel	28.6515531	-17.8816258	1	0.1691
5	2142	Papel	28.5960491	-17.8790818	1	0.2706
6	2165	Papel	28.660338	-17.9207753	1	0.2799
7	2188	Papel	28.6584208	-17.92257	1	0.1517
8	2211	Papel	28.7809318	-17.7643915	1	0.1015
9	2229	Papel	28.686571	-17.7651881	1	0.1333
10	2247	Papel	28.6805308	-17.7659866	1	0.0870
11	2267	Papel	28.6387563	-17.931026	1	0.0803
12	2286	Papel	28.6200899	-17.7528396	3	0.5866
13	3625	Papel	28.674519	-17.7815801	1	0.2686
14	5688	Papel	28.6172393	-17.9030998	1	0.2066
15	6275	Papel	28.6395189	-17.9304823	1	0.1753
16	6644	Papel	28.6548648	-17.767114	1	0.2689
17	6801	Papel	28.640717	-17.8965823	1	0.1989
18	7071	Papel	28.6983471	-17.7786636	1	0.1179
19	7766	Papel	28.6908621	-17.7835493	1	0.1271
20	15	planta	28.6618476	-17.7770484	1	0.0000

Tabla 2.9: Datos del subconjunto aleatorio

Además, la columna de las tasas de llenado hará el papel del beneficio en nuestro problema, por lo que nuestro objetivo consistirá en maximizar el valor de dichas tasas ya que cuanto mayor sea este valor objetivo mayor cantidad de residuos serán recogidos.

2.3.2. Subconjunto de nodos de una ruta real

Como también disponemos de los datos de rutas ya realizadas nos interesa determinar otro subconjunto de puntos pero esta vez determinado, no del total de contenedores sino, de los contenedores que han sido recogidos en una ruta

real de las que ya disponemos. Así, procediendo de la misma forma que en el apartado anterior, buscaremos un subconjunto de 21 puntos pero , en este caso, de la ruta realizada por un vehículo de recogida de papel el día 1 de diciembre de 2017. La información de estos nuevos 21 puntos la podemos encontrar en la tabla 2.10 cuyas columnas representan la misma información que encontramos en la tabla 2.9.

Nodos	Contenedores	Tipo	Coord. X	Coord. Y	NC	Tasa llenado
0	0	planta	28.645233	-17.8990968	1	0.0000
1	2110	Papel	28.6526718	-17.8816026	1	0.3010
2	2118	Papel	28.6515531	-17.8816258	1	0.1691
3	2119	Papel	28.6506805	-17.8817105	1	0.2489
4	2151	Papel	28.6599416	-17.9178718	1	0.2413
5	2161	Papel	28.6565825	-17.9080941	1	0.3243
6	2165	Papel	28.660338	-17.9207753	1	0.2799
7	2173	Papel	28.6380553	-17.9075676	1	0.2591
8	2189	Papel	28.65889	-17.9144988	1	0.0803
9	2267	Papel	28.6387563	-17.931026	1	0.1333
10	2268	Papel	28.6411548	-17.9338421	1	0.2877
11	4872	Papel	28.6551648	-17.9051015	1	0.3074
12	5238	Papel	28.6610808	-17.8520436	1	0.2146
13	5688	Papel	28.6172393	-17.9030998	1	0.2066
14	5792	Papel	28.6027936	-17.8810305	1	0.1372
15	5866	Papel	28.6166673	-17.9041021	2	0.2136
16	6275	Papel	28.6395189	-17.9304823	1	0.1753
17	7328	Papel	28.6438228	-17.9338048	1	0.1292
18	7353	Papel	28.6477895	-17.8782155	1	0.0687
19	7835	Papel	28.6499356	-17.9091553	2	0.1773
20	15	planta	28.6618476	-17.7770484	1	0.0000

Tabla 2.10: Datos del subconjunto aleatorio de una ruta real

Los únicos datos que faltan para ambos subconjuntos son las distancias entre todos los pares de nodo. Esta información no se obtuvo a partir de las coordenadas usando la distancia euclídea sino que ya se disponía de ella con antelación y nos ofrece directamente el tiempo real que se tarda en recorrer la distancia que separa a los pares de nodos medido en segundos.

2.3.3. Resultados

Subconjunto de nodos aleatorios

Una vez establecidos los subconjunto de nodos que vamos a utilizar ejecutamos el OP, en este caso, para el subconjunto de nodos aleatorios seleccionados entre el total de contenedores de papel de la isla de La Palma representados en la tabla 2.9. Los resultados obtenidos se muestran en la tabla 2.11.

Orienteering Problem 21 nodes *(La Palma)*				
T_{max}	Obj	Route	Time	Memory
3000	0,8965	0-17-4-13-3-20	0.2	1.5
3500	0,9963	0-17-13-3-16-20	0.4	1.6
4000	1,1654	0-17-4-13-3-16-20	1.8	1.9
4500	1,4057	0-17-4-13-3-2-16-20	2.9	2.0
5000	1,752	0-17-4-13-3-16-12-20	2.7	2.0
5500	1,9923	0-17-4-13-3-2-12-16-20	3.1	1.9
6000	1,9923	0-17-4-13-3-16-2-12-20	10.3	2.3
6500	2,2292	0-17-14-5-4-13-3-16-12-20	66.1	4.2
7000	2,4695	0-17-14-5-4-13-3-16-12-2-20	23.6	3.3
7500	2,4917	0-17-7-6-14-5-13-3-12-16-20	108.2	5.0
8000	2,732	0-17-6-7-14-5-13-3-2-16-12-20	84.6	4.4
8500	2,9164	0-17-6-7-11-15-14-5-4-13-3-16-12-20	147.7	5.8

Tabla 2.11: Resultados del OP para 21 nodos de un subconjunto aleatorio

Podemos observar que los mejores tiempo de ejecución se obtienen para un tiempo máximo con un valor como mucho de 6000 segundos. A partir de este valor, el programa requiere de mucho tiempo para la ejecución y a partir de un valor de 8500 ya dejamos de obtener resultados dentro de un tiempo razonable.

Con ayuda de aplicaciones que dibujen mapas podemos plasmar estas rutas obtenidas con GUSEK en un mapa real. En nuestro caso, hemos utilizado la aplicación *My Maps* que posee *Google Maps* la cuál nos permite describir rutas, mediante el uso de coordenadas.

En la figura 2.11 aparece la representación de todos los nodos seleccionados del subconjunto aleatorio y en la figura 2.12 podemos observar la ruta que corresponde a la solución con tiempo límite 3500 que encontramos en la tabla 2.12. Observamos, en dicha figura, como la ruta recorre una gran cantidad de superficie dejando, por ejemplo entre los nodos que denominamos B y C, un tramo largo donde no se recoge ningún contenedor. Esto se debe a que estos

nodos han sido seleccionados entre todos los nodos de la isla de La Palma y, por tanto, nos arriesgamos a seleccionar nodos cuyas distancias tengan un valor muy elevado.

Orienteering Problem 21 nodes *(La Palma)*				
T_{max}	Obj	Route	Time	Memory
3500	0,9963	0-17-13-3-16-20	0.4	1.6

Tabla 2.12: Resultados del OP con 21 nodos aleatorios y $T_{max} = 3500$

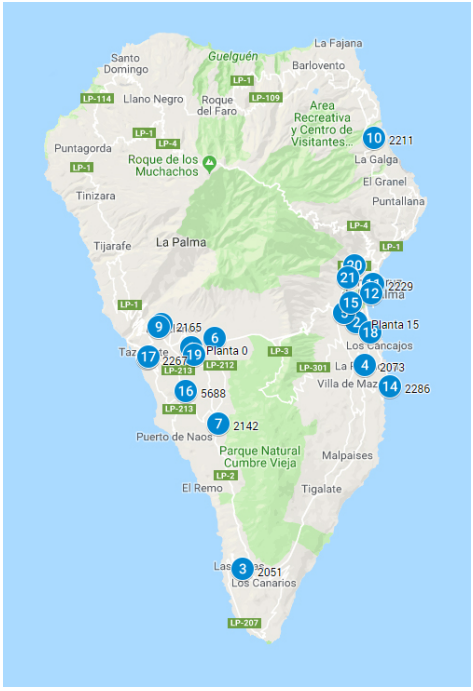


Figura 2.11: Mapa de La Palma con los 21 nodos aleatorios

Como se puede observar en los resultados obtenidos cuánto mayor sea el tiempo límite mayor número de nodos se puede recorrer. Para poder apreciar la diferencia, en la figura 2.13 hemos dibujado sobre el mapa real de La Palma la solución al OP con 21 nodos aleatorios con el T_{max} máximo que hemos conseguido ejecutar. Se han seleccionados más nodos aunque sigue habiendo tramos de la ruta demasiados largos sin recoger ningún contenedor.

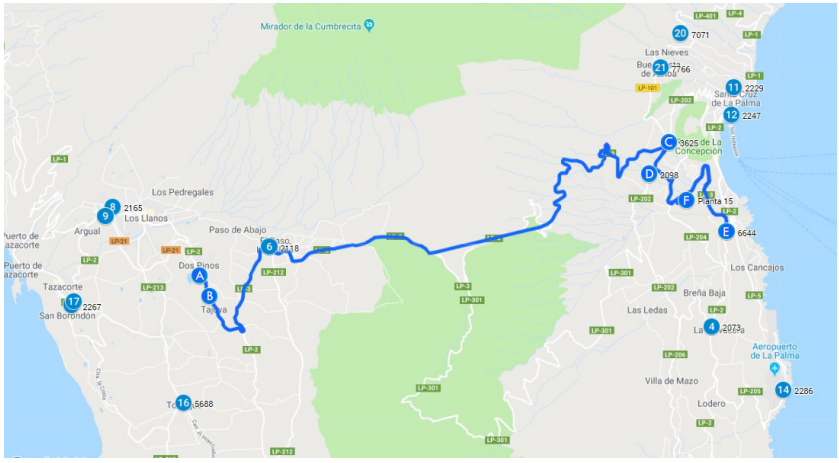


Figura 2.12: Ruta con T_{max}= 3500

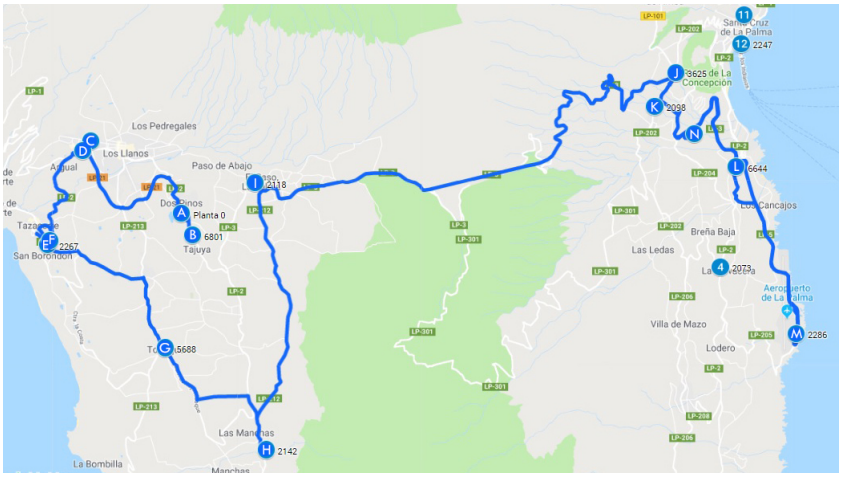


Figura 2.13: Ruta con T_{max}= 8500

Subconjunto de nodos de una ruta real

Si para el subconjunto de 21 nodos de una ruta real, que podemos encontrar en la tabla 2.10, ejecutamos el programa del OP implantado en GUSEK obtenemos los siguientes resultados:

Orienteering Problem 21 nodes *(La Palma)*				
T_{max}	Obj	Route	Time	Memory
2800	0,5499	0-1-3-20	0.1	1.6
3000	0,7877	0-2-1-3-18-20	0.3	1.8
3500	1,0951	0-11-2-1-3-18-20	0.4	1.8
4000	1,6098	0-11-5-7-2-1-3-20	1.1	1.9
4500	2,03	0-11-5-7-13-15-2-1-3-20	1.1	2.6
5000	2,33	0-11-5-8-4-6-7-2-1-3-18-20	5.1	2.7
5500	2,7502	0-11-5-8-4-6-7-15-13-2-1-3-18-20	9.0	3.1
6000	3,0222	0-11-5-4-6-10-9-7-13-15-2-1-3-20	18.3	4.0
6500	3,3858	0-11-5-8-4-6-10-9-7-13-15-2-1-3-18-12-20	18.4	4.6
7000	3,6903	0-11-5-8-4-6-16-17-10-9-7-13-15-2-1-3-18-12-20	19.0	4.9

Tabla 2.13: Resultados del OP con 21 nodos pertenecientes a una ruta real

En comparación con el subconjunto de nodos escogidos de forma aleatoria, en este caso se obtienen mejores resultados para las tasas de llenado a partir del tiempo máximo $T_{max} = 3500$ y aunque es ejecutable solo hasta un tiempo máximo de 7000, el número de nodos que se recorren en cada ruta es mayor. Además, hay que destacar que el tiempo de ejecución que se requiere en este caso es mucho menor. Un ejemplo de las rutas con un valor de T_{max} igual a 3500 y 7000 se muestran en las figuras 2.14 y 2.15.

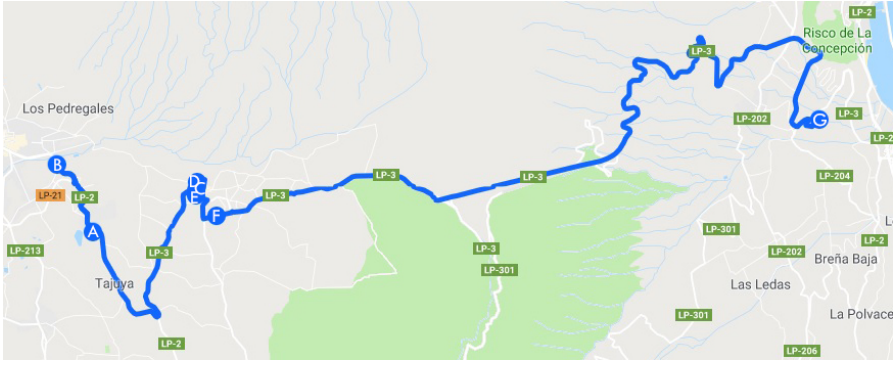


Figura 2.14: Ruta real con T_{max}= 3500

A diferencia del caso anterior, en ambas imágenes se puede apreciar que los nodos que se visitan están más agrupados y por tanto mejora el objetivo obtenido así como la cantidad de números que podemos encontrar en cada ruta.

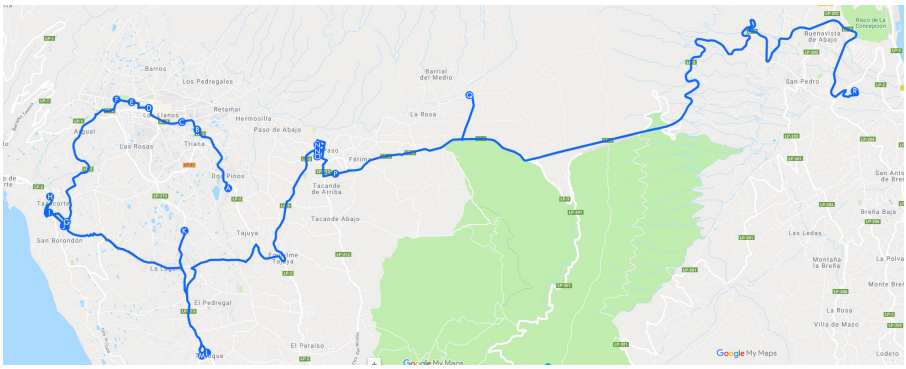


Figura 2.15: Ruta real con T_{max}= 7000

Todo el trabajo realizado podría extenderse a un caso dónde se estudie la totalidad de contenedores de los que se dispone, por desgracia debido a la limitada capacidad de GUSEK para realizar dicha ejecución no hemos podido reflejarlo en este trabajo. Si se aplicase otro software con la capacidad necesaria entonces podríamos disponer en poco tiempo de una ruta de recogida que maximice la cantidad de residuos a recoger que era el principal objetivo en este trabajo.

2.3.4. Extensión del estudio a una semana

Si en vez de tener un solo vehículo de recogida contáramos con una flota de vehículos para la recogida de residuos el problema consistiría en resolver el modelo propuesto en el capítulo uno, como una extensión del OP, conocida como TOP. Si en vez de tener una flota y solo tuviéramos un vehículo el modelo que describe la recogida de residuos de dicho vehículo cada día de una determinada semana también podría modelarse como un TOP.

Sin embargo, en vez de modelar el problema de recogida de residuos con un solo vehículo a lo largo de una semana como un TOP, hemos optado por modelar el problema como una reiteración de OP dónde cada iteración corresponde a cada día de la semana ya que, en este caso, las tasas de llenado de cada contenedor varía cada día según se haya seleccionado para su recogida o no. Si fijamos los datos como los que aparecen en la tabla 2.10 y ejecutamos el OP para el día en el que se realizó dicha ruta (viernes 01 de Diciembre de 2017) y un tiempo máximo para la ruta de 6000 obtenemos el resultado que aparece en la tabla 2.14.

Orienteering Problem 21 nodes *(La Palma)*					
Día	T_{max}	Obj	Route	Time	Memory
Viernes	6000	3,0222	0-11-5-4-6-10-9-7-13-15-2-1-3-20	18.3	4.0

Tabla 2.14: Resultados de la primera iteración del OP

Así, los contenedores que representan los nodos que aparecen en la ruta han sido recogidos, y por tanto, no cambian su tasa de llenado, sin embargo , todos aquellos que no han sido recogidos le sumamos su tasa de llenado correspondiente. Además, como hemos ejecutado el programa para un viernes y los fin de semanas no hay servicio de recogida debemos sumarle también a todos los nodos sus tasas por cada día sin recoger (sábado y domingo). De esta manera obtenemos un nuevo conjunto de datos que usaremos para determinar la ruta a realizar el resto de días. En la tabla 2.16 podemos encontrar las nuevas tasas de llenados para los días en los que se desea conocer la ruta de recogida. Destacamos que el miércoles día 6 y viernes 8 de diciembre fueron días festivos por lo que no aparecen los datos para dichos días sino que las tasas de llenado de cada nodo se vuelven a sumar como pasa en el caso de los días de fin de semana.

Por ejemplo, como el nodo 1 cuya tasa de llenado es 0.3010 está en la ruta del viernes tenemos que mantener su valor de llenado, sin embargo, como en los días de fin de semana no se realizan rutas debemos sumarle su valor correspondiente por cada día del fin de semana. Así, su nueva tasa de llenado será $0.3010 + 0.3010 + 0.3010 = 0.9030$.

De esta forma, si volvemos a ejecutar el modelo en GUSEK con los nuevos datos, para cada día de la semana, obtenemos, los resultados que aparecen en la tabla 2.15.

Orienteering Problem 21 nodes *(La Palma)*					
Día	T_{max}	Obj	Route	Time	Memory
Lunes	6000	9,1767	0-11-5-8-4-6-7-15-13-2-1-3-18-12-20	18.9	5.5
Martes	6000	6,2257	0-11-19-16-17-10-9-15-13-14-2-1-3-20	1.8	2.9
Jueves	6000	7.2978	0-11-5-8-4-6-7-13-15-2-1-3-18-20	2.7	2.8

Tabla 2.15: Resultados de las iteraciones del OP

A continuación, podemos encontrar en la figura 2.16 como se distribuyen los 21 nodos de la ruta real sobre el mapa d la isla de La Palma. Además, las figuras 2.17, 2.18 y 2.19 nos muestran las rutas correspondiente a las soluciones, que podemos encontrar en la tabla 2.15, de la iteraciones del lunes, martes y jueves respectivamente.

Nodos	Contenedores	Viernes	Lunes	Martes	Jueves
0	0	0.0000	0.0000	0.0000	0.0000
1	2110	0.3010	0.9030	0.3010	0.6020
2	2118	0.1691	0.5073	0.1691	0.3382
3	2119	0.2489	0.7467	0.2489	0.4978
4	2151	0.2413	0.7239	0.2413	0.7239
5	2161	0.3243	0.9729	0.3243	0.9729
6	2165	0.2799	0.8397	0.2799	0.8397
7	2173	0.2591	0.7773	0.2591	0.7773
8	2189	0.0803	0.3212	0.0803	0.2409
9	2267	0.1333	0.3999	0.5332	0.2666
10	2268	0.2877	0.8632	1.1509	0.5754
11	4872	0.3074	0.9222	0.3074	0.6148
12	5238	0.2146	0.8584	0.2146	0.6438
13	5688	0.2066	0.6198	0.2066	0.4132
14	5792	0.1372	0.4588	0.6860	0.2744
15	5866	0.2136	0.6408	0.2136	0.4272
16	6275	0.1753	0.7012	0.8765	0.3506
17	7328	0.1292	0.5168	0.6460	0.2584
18	7353	0.0687	0.3435	0.0687	0.2061
19	7835	0.1773	0.7092	0.8865	0.3246
20	15	0.0000	0.0000	0.0000	0.000

Tabla 2.16: Datos para las iteraciones de cada día

Podemos observar, tanto en la tabla como en las figuras que reflejan las rutas de iterar el OP a lo largo de una semana, como el lunes y el jueves comparte una similitud casi exacta de soluciones. Esto nos dice que, como los días anteriores a los mencionados no hay servicios, los nodos que aparecen en dichas rutas son lo que poseen mayor valor de tasas de llenado.

Además, se observa en dichos resultados que a lo largo de toda la semana todos los puntos de recogida son visitados, por lo que la acción de iterar el OP variando la tasa de llenado nos permite recoger los residuos de todos los contenedores visitando más de una vez los que tienen mayor valor de tasa de llenado.

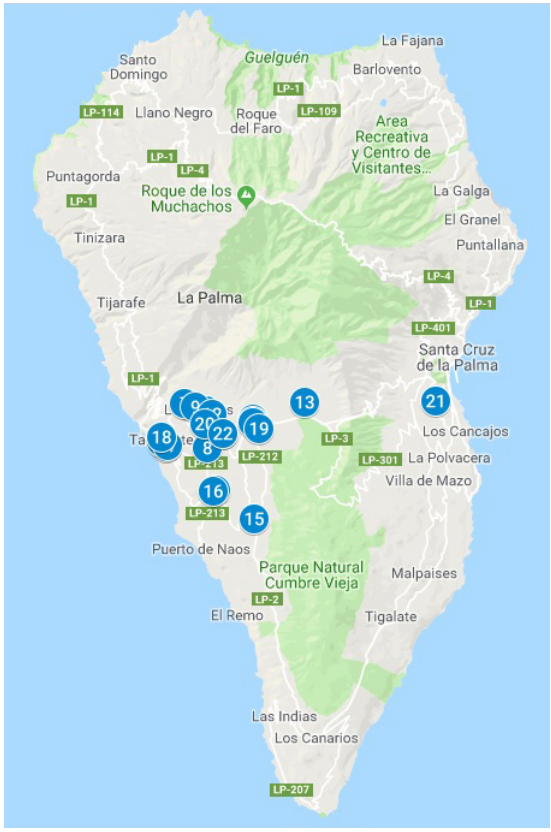


Figura 2.16: 21 nodos de una ruta real

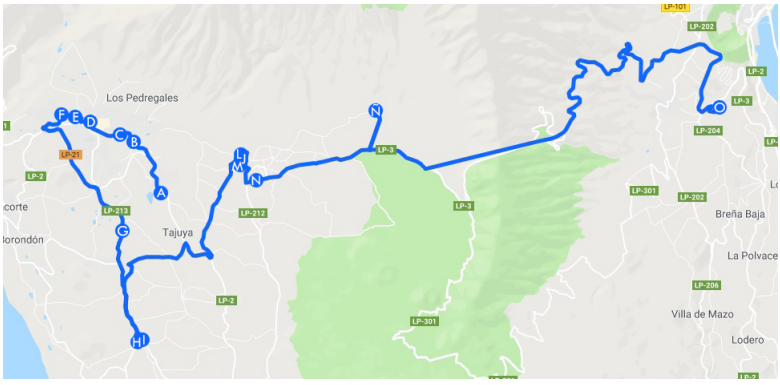


Figura 2.17: Ruta para la iteración del lunes

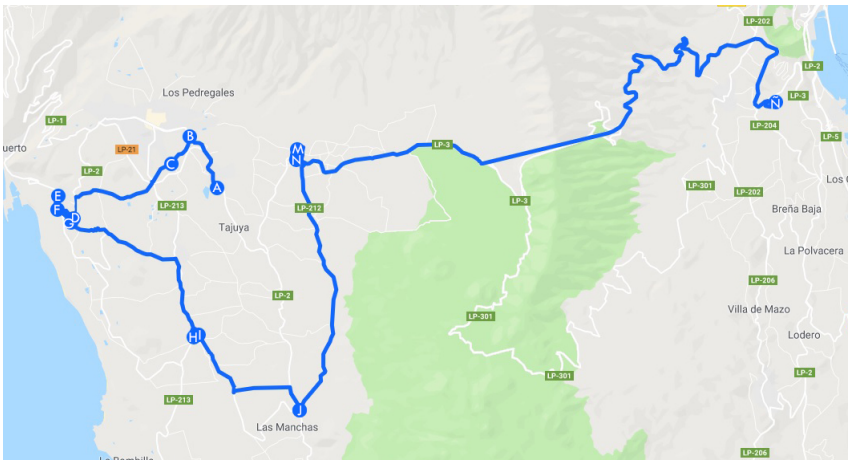


Figura 2.18: Ruta para la iteración del martes

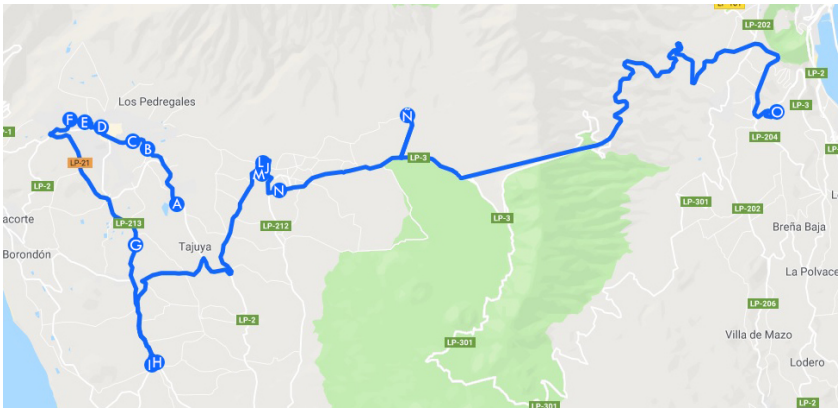


Figura 2.19: Ruta para la iteración del jueves

Conclusiones

En este trabajo se ha mostrado como usar las herramientas matemáticas de optimización para la mejora en la planificación de las rutas de recogida de residuos para el reciclaje. Una revisión de la literatura en este área aporta una visión de la relevancia de este tema en los esfuerzos por aumentar el volumen de residuos reciclados, mejorando así, la eficiencia de todo el proceso de reciclaje para mitigar el impacto medioambiental de la actividad humana y reducir los efectos en el cambio climático.

Se establece una fuerte relación entre la planificación óptima de esta tarea y los estudios en la literatura especializada sobre el Orienteering Problem (OP) y las extensiones del mismo. Esta relación, permite aplicar directamente algunos avances de estos problemas en la optimización de rutas de recogidas pero aparecen también algunas particularidades que hacen necesaria explorar nuevos procedimientos para contemplar situaciones reales. Esto ocurre, por ejemplo, al planificar las recogidas en un horizonte temporal que incluye varios días teniendo en cuenta la obligada inactividad en días no laborables.

Como trabajos futuros que se plantean a partir de este trabajo podemos destacar la necesidad de utilizar procedimientos heurísticos para poder tratar los grandes volúmenes de datos de situaciones reales y aportar soluciones de alta calidad con un esfuerzo de computo razonable. También es necesario contemplar otros aspectos, criterios o indicadores del proceso de recogida, aparte del volumen de residuos, como puede ser el número de contenedores que se llegan a rebosar antes de ser recogidos o la necesidad de reubicar algunos contenedores o aumentar su número, para lo que es necesario decidir cual sería la localización más conveniente. Por otro lado, se hace necesario utilizar metodologías que permitan tratar adecuadamente la incertidumbre inherente a esta problemática utilizando las técnicas de computación flexible, optimización estocástica o simu-

lación. Finalmente, es necesario estudiar con más profundidad la planificación en horizontes temporales más amplios teniendo en cuenta las circunstancias que se presentarán en un futuro cercano. Por ejemplo, al planificar la actividad de una semana es necesario tener en cuenta en el diseño de las rutas no solamente lo que ocurrió anteriormente, sino también lo que está a punto de ocurrir. De esta forma, la planificación de las rutas de un lunes, o un martes, no debe ser la misma cuando se trata de una semana sin interrupciones que cuando se sabe que el miércoles es festivo y por tanto será inevitable posponer la recogida de los contenedores.

A

Apéndice

A.1. Programas con *Dev-C++*

A.1.1. Distancias entre nodos

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <windows.h>
#include <math.h>


main(){//comienzo del programa
    FILE* fichero;
    float b[10],c[10];
    float matrix[21][21];
    int a,i,j;
for(i=0;i<=20;i++){
        printf("Introduzca coordenada x %d \n", i);
        scanf("%f",&b[i]);
        printf("Introduzca coordenada y %d \n", i);
        scanf("%f",&c[i]);
        }
      fichero=fopen("resultado.txt","wt");
       for(i=0;i<=20;i++){
        for(j=0;j<=20;j++){
        matrix[i][j]=sqrt((b[i]-b[j])*(b[i]-b[j])  + (c[i]-c[j])*(c[i]-c[j]));
        fprintf(fichero,"%f.2 \n", matrix[i][j]);
             }
          }


system("pause");
}
```

Figura A.1: Programa para calcular distancias euclídeas

Lista de Figuras

Lista de Tablas

Bibliografía

[1] Bektaş, T. and Gouveia, L. (2014). Requiem for the miller–tucker–zemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832.

[2] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

[Expósito-Márquez et al.] Expósito-Márquez, A., Expósito-Izquierdo, C., Brito-Santana, J., and Moreno-Pérez. Solving an eco-efficient vehicle routing problem for waste collection with grasp.

[4] Faccio, M., Persona, A., and Zanin, G. (2011). Waste collection multi objective model with real time traceability data. *Waste Management*, 31(12):2391–2405.

[5] Garey, M. R. and Johnson, D. S. (2002). *Computers and intractability*, volume 29. wh freeman New York.

[6] Gendreau, M., Laporte, G., and Semet, F. (1998). A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks: An International Journal*, 32(4):263–273.

[7] Ghisellini, P., Cialani, C., and Ulgiati, S. (2016). A review on circular economy: the expected transition to a balanced interplay of environmental and economic systems. *Journal of Cleaner production*, 114:11–32.

[8] Giusti, L. (2009). A review of waste management practices and their impact on human health. *Waste management*, 29(8):2227–2239.

[9] Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318.

[10] Guerra Talavera, R. and Garcia, T. P. (2008). Canarias: entre el desarrollo turístico y la protección al medio. *Études caribéennes*, online(9-10).

[11] Gunawan, A., Lau, H. C., and Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332.

[12] Han, H. and Ponce Cueto, E. (2015). Waste collection vehicle routing problem: literature review. *Promet-Traffic&Transportation*, 27(4):345–358.

[13] Idrus, Z., Ku-Mahamud, K. R., and Benjamin, A. M. (2017). Waste collection vehicle routing problem benchmark datasets and case studies: A review. *Journal of Theoretical and Applied Information Technology*, 95(5):1048–1062.

[14] Kaza, S., Yao, L., Bhada-Tata, P., and Van Woerden, F. (2018). *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. World Bank Publications.

[15] McKinnon, A., Browne, M., Whiteing, A., and Piecyk, M. (2015). *Green logistics: Improving the environmental sustainability of logistics*. Kogan Page Publishers.

[16] Melián, B., Moreno Pérez, J. A., and Moreno Vega, J. M. (2003). Metaheurísticas: Una visión global. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 7(19):0.

[17] Pearce, D. W. and Turner, R. K. (1990). *Economics of natural resources and the environment*. JHU Press.

[18] Ramos, T. R. P., de Morais, C. S., and Barbosa-Póvoa, A. P. (2018). The smart waste collection routing problem: Alternative operational management approaches. *Expert Systems with Applications*, 103:146 – 158.

[19] Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809.

[20] Vansteenwegen, P., Souffriau, W., and Berghe, Greet Vanden & Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290.

[21] Vansteenwegen, P., Souffriau, W., and Van Oudheusden, D. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10.

Intelligent Optimization for Waste Collection Routing

Yanira García Hernández
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100699845@ull.edu.es

Abstract

The growing negative effects concerning the environment, due to pollution caused by waste generation, have induced the movements of, not only governments, but also, companies and society with the purpose of establishing the necessary activities to control and reverse this situation. One of the activities that has been made with more emphasis is the waste classification and collection. The main goal of this task is to increase the amount of waste that can be recycled. In this work we aim to show the use of mathematical tools for optimizing the routes for waste collection and so maximizing the amount of collected waste to be recycled.

1. Introduction

Climate change is a major challenge to be faced in our time. Among other causes, the generation of waste by human activity [1] and its effect on the environment, plays a key role in this problem. This work deals mainly with modeling a problem of combinatorial optimization -a branch of mathematics belonging to mathematical programming- that improves the collection of waste so that, for example,to maximized the amount of waste to be collected. In this type of optimization problems, apparently easy to solve because they have a space of finite solutions, it is not possible finding an exact solution but trying to find the best possible solution. The key to finding the best solutions is how the way in which the problem is modeled since solutions will depend on this modeling. Our objective is to find a mathematical model to obtain the best routes in waste collection and apply it later to a real case.

2. Waste Collection Routes

In this first chapter, in addition to a review of the current state of the art, we focus on describing a mathematical model that explains the waste collection routes planning and their possible extensions. The basic problem consists of finding the route from a given origin to the waste depot through a selected set of collection points or waste containers such that the total amount of collected waste is maximised. Let m be the number of collection points that combined with the mandatory origin and ending points form the set $I = 0, .., m+1$. Each point has a value (v_i) that is the amount of waste in the container. The travel time between each pair of points (t_{ij}) is known and the total route duration cannot exceed a given time (T_{max}). Our problem can be defined like the known Orienteering Problem (OP) [2].

With the previous notation and the next variables x_{ij} (that estates if the route goes form point i to j) and p_i (that shows position of the point i in the route), the problem is formulated as the following Linear Programming Mathematical Model.

Maximise

$$Max \sum_{i=0}^{m} \sum_{j=1}^{m} v_j x_{ij}, \tag{1}$$

such that:

$$\sum_{j=1}^{m} x_{0j} = \sum_{i=0}^{m-1} x_{im}; \tag{2}$$

$$\sum_{i=0}^{m-1} x_{id} = \sum_{j=1}^{m} x_{dj} \leq 1; d = 1..m \tag{3}$$

$$\sum_{i=0}^{m-1} \sum_{j=1}^{m} t_{ij} x_{ij} \leq T_{max}, \tag{4}$$

$$1 \leq u_i \leq m; i = 1..m \tag{5}$$

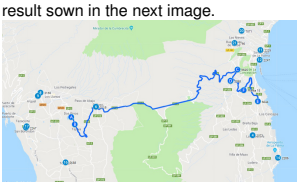
$$u_i - u_j + 1 \leq m(1 - x_{ij}); i, j = 0..m \tag{6}$$

$$u_i \in \mathbb{Z}, x_{ij} \in \{0, 1\}; i, j = 0..m \tag{7}$$

The main if the extensions are obtained by adding time window constraints at collection points (OPTW) and by considering a homogeneous fleet of vehicles to collect waste (TOP) [3]

3. Experiments

In the second chapter we show the implementation of the the mathematical model (OP) and its extensions to be solved by a software package called GUSEK that solves mixed integer programming (MIP) problem. First of all, we make several tests to OP and OPTW with known instances that can be found in the web page of the group of Professor Vansteenwegen [2] Once tests are done and verifying that both programs are executed optimally, the next step consist of doing the same test with real cases. The real case to study is to find the optimal route for waste collection at La Palma island, for this we have all data from collection containers of recyclable waste divided in two type of containers (one for each kind of waste): blue container for paper and yellow container for plastic [4]. Since there are specific vehicles for each kind of recyclable waste, we focus on the collection of only for one, for example, paper. For the other kind of waste, the procedure will be the same, just changing to correspondent data. For paper containers, we have data about numbers, locations and fill rate that play the role of benefit. So, before executing the program we make a previous selections of random nodes due to GUSEK cannot be executed with a big amount of data. For the real case, we do not have data about the time of visit for each node so we only can execute OP. For a specific maximum time for the route we have the result sown in the next image.



The TOP extension can be apply to obtain the collection route during a week if the value of parameters stay the same. However, for the dairy routes of a collection vehicle this model is no suitable because the filling rates for each container keep the amount of waste growing if it is not collected, otherwise the value of the fill level is reset. Thus, an OP iteration for each week day have to be applied updating the values of the collection points.

References

[1] Kaza, S., L. Yao, P. Bhada-Tata, and Van Woerden, 2018. *What a Waste 2.0: A Global Snapshot of Solid WasteManagement to 2050.* World Bank Publications.

[2] Vansteenwegen, P., W. Souriau, and D. Van Oudheusden, 2011. *The orienteering problem: A survey.* European Journal of Operational Research.209(1), 1-10.

[3] Gunawan, A., H. C. Lau, and P. Vansteenwegen, 2016. *Orienteering problem: A survey of recent variants, solution approaches and applications.* European Journal of Operational Research, 255(2), 315-332.

[4] Expósito-Márquez, A., Expósito-Izquierdo, C., Brito-Santana, J., and Moreno-Pérez, J. A. 2018. |em Solving an Eco-efficient Vehicle Routing Problem for Waste Collection with GRASP. In International Symposium on Intelligent and Distributed Computing (pp. 215-224). Springer, Cham.