Optimización de la Recolección de Residuos en la Zona Sur de la Ciudad de Buenos Aires

Flavia Bonomo[*]
Guillermo Durán[**]
Federico Larumbe[***]
Javier Marenco[****]

Resumen

En este trabajo presentamos una propuesta para optimizar el recorrido de los camiones encargados de la recolección de los contenedores de residuos en la zona sur de la ciudad de Buenos Aires (Argentina), utilizando técnicas de programación matemática. Se describe el proceso de depuración de los datos disponibles y los pasos realizados para aplicar herramientas de programación lineal entera para este problema. Se obtienen propuestas que minimizan la distancia total recorrida por cada camión y muestran una disminución de los trayectos de entre un 10 % y un 40 % con respecto a los recorridos actuales. Además, estos nuevos trayectos disminuyen sustancialmente el desgaste final de los camiones.

Palabras Clave: Programación Entera, Recolección de Residuos, Ruteo de Vehículos.

[*]Departamento de Computación, FCEyN, Universidad de Buenos Aires y CONICET, Argentina
[**]Departamento de Ingeniería Industrial, FCFM, Universidad de Chile; Departamento de Matemática, FCEyN, Universidad de Buenos Aires y CONICET, Argentina
[***]Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina
[****]Instituto de Ciencias, Universidad Nacional de General Sarmiento, Argentina

1. Introducción

La recolección de residuos puede dividirse en 3 grandes tipos: domiciliaria, comercial e industrial [5]. La recolección domiciliaria consiste en atender fundamentalmente casas particulares. Un mismo camión puede atender en una misma ruta desde 100 hasta 1000 casas cada día. La frecuencia del servicio varía según las ciudades, aunque en general las rutas suelen repetirse una vez todos los días. La recolección comercial consiste en atender shoppings, restaurants o edificios de oficinas. Cada ruta comercial puede atender entre 60 y 400 clientes, con 2 o 3 visitas al depósito en cada día. Cada cliente puede tener una ventana de tiempo prefijada para ser visitado. La recolección industrial consiste en atender fábricas, obras y edificios en construcción. La diferencia principal con la recolección comercial es que los contenedores a ser vaciados suelen ser 4 o 5 veces más grandes, y en muchos casos sólo un contenedor puede ser atendido por vez, lo que convierte al problema en uno de ruteo de vehículos muy diferente al de la recolección comercial. Una buena recopilación sobre artículos de ruteo de camiones para recolección de residuos aparece en [7].

Los objetivos en este tipo de problemas suelen ser diversos: minimizar el número de camiones, minimizar la distancia recorrida, conseguir rutas compactas o balancear la carga entre los distintos camiones.

Existen en la literatura varios trabajos que muestran la aplicación de sistemas basados en técnicas de programación matemática para la recolección de residuos en diferentes ciudades del mundo. Algunos ejemplos son el de Chicago, en los Estados Unidos [4], el de una importante ciudad en Taiwan [2] y el de Lisboa, en Portugal [10]. La mayor parte de los artículos que muestran la solución de problemas reales de recolección de residuos consiste en la implementación de heurísticas, debido a la dificultad de los problemas.

En la Ciudad de Buenos Aires se encuentra en implementación un nuevo sistema para la recolección de residuos domiciliarios, que consiste en ubicar contenedores distribuidos por la ciudad, para que los vecinos depositen en estos contenedores las bolsas de residuos. Cada contenedor tiene 1000 litros de capacidad. La intención de este proyecto de la Ciudad es reemplazar la deposición de residuos en cestos individuales por el uso de estos contenedores generales, contribuyendo así a la higiene general y a la eficiencia en la recolección.

A los efectos de la recolección de residuos, la ciudad se divide en 6 zonas (figura 1). En cada zona mostrada en la figura, se ven pintadas con un color oscuro los lugares donde el nuevo sistema se encuentra actualmente implementado. Las zonas 1 a 4 y la zona 6 tienen asignada una empresa cada una, cuya responsabilidad es gestionar la recolección de residuos en esa área. La zona 5

está asignada al Ente de Higiene Urbana (EHU), un organismo del Gobierno de la Ciudad de Buenos Aires con las mismas responsabilidades.

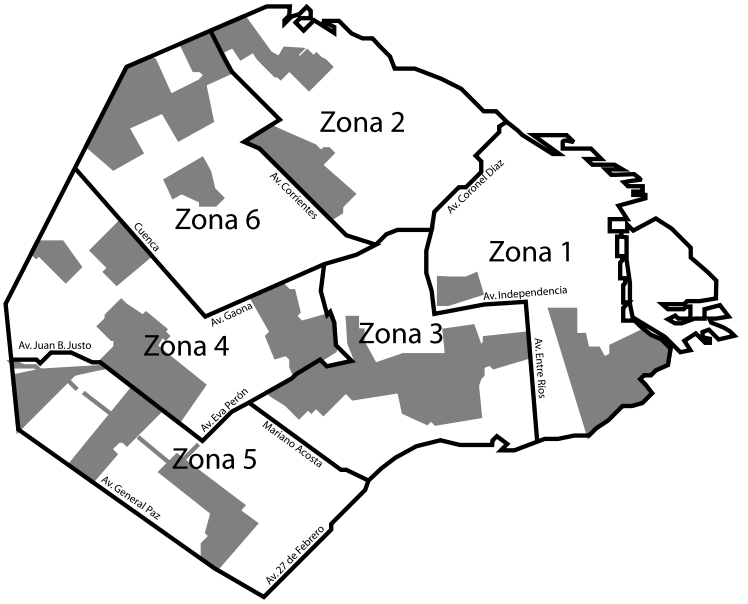


Figura 1: Zonas para la recolección de residuos en Buenos Aires.

La ubicación de los contenedores está prefijada de antemano, y los contenedores que recolecta el EHU están agrupados en 4 sub-zonas. Se tienen 4 camiones de recolección de contenedores. Cada camión tiene una sub-zona asignada, y hace dos recorridos iguales, uno por la mañana y otro por la tarde. El recorrido se inicia en el EHU, se dirige al primer contenedor, recolecta todos sus contenedores asignados, descarga en el depósito los residuos y vuelve al EHU. Aproximadamente, el camión tarda 3 minutos en recolectar un contenedor y limpiar el entorno donde se encuentra. A la mañana, los camiones salen a las 7 hrs. del EHU y vuelven aproximadamente a las 15 hrs. A la tarde, salen a las 18 hrs. Las sub-zonas tienen 47, 133, 134 y 161 contenedores, respectivamente.

En un camión entran 15000 kg. de residuos. A la mañana los camiones recolectan entre 10000 kg. y 15000 kg., y a la tarde cargan entre 2500 kg. y 5000 kg. Hay un camión de limpieza que pasa cada 10 días detrás del camión de recolección y limpia los contenedores vacíos. En el área de descarga, se compactan los residuos y un camión con mayor capacidad los lleva al lugar final de deposición, ubicado en el conurbano de la ciudad.

El propósito del presente trabajo es mejorar la ruta de cada camión con respecto a las rutas hoy existentes, de modo que cada camión parta del EHU, recolecte todos los contenedores de su sub-zona, vaya al depósito y vuelva al

EHU. Los objetivos consisten en minimizar la distancia recorrida y disminuir el desgaste de los camiones, en ese orden. Cuando el camión se encuentra cargado, existe un desgaste considerable de varias de sus partes, con lo cual es conveniente que no recorra una gran distancia con demasiado peso.

Cabe destacar que los empleados cobran por jornada laboral de medio tiempo, por lo que la optimización no influye en el costo laboral, aunque permite realizar la misma tarea recorriendo menos distancia, provocando un menor desgaste de los camiones y permitiendo un mejor tráfico en la ciudad.

Con los datos disponibles actualmente, no es posible estimar apropiadamente los tiempos de recorrido en las calles y avenidas que deben recorrer los camiones. Por este motivo, no se plantea en este trabajo la optimización de los tiempos de recorrido, y en cambio se trabaja con la distancia total como objetivo principal.

Para cuantificar el desgaste del camión a lo largo del recorrido, utilizamos el *trabajo* realizado por el camión. El trabajo en un tramo se calcula como el producto de la distancia recorrida por la fuerza realizada en esa dirección. La unidad básica de trabajo en el Sistema Internacional es el Newton por metro, que se denomina Joule (J). En cada tramo del itinerario entre dos contenedores consecutivos, el trabajo que realiza el camión es el producto de la distancia por el peso de la carga del camión. El trabajo total realizado es la suma de los trabajos de cada tramo. Hasta lo mejor de nuestro conocimiento, no hay trabajos en la literatura de recolección de residuos donde se plantee la disminución del trabajo realizado por los camiones.

En este artículo describimos la aplicación de herramientas computacionales para la resolución efectiva de este problema por medio de técnicas de programación lineal entera, para los cuatro camiones del EHU en las sub-zonas actualmente cubiertas por contenedores. Se describen las herramientas utilizadas y el trabajo computacional de depuración e interpretación de los datos. Como se verá en la sección 5, se obtuvieron recorridos muy eficientes desde el punto de vista de los objetivos planteados, dado que se consiguió disminuir la distancia y el trabajo realizado por los camiones entre un 10 % y un 45 %, con respecto a los recorridos actuales.

Este trabajo está organizado de la siguiente forma. En la sección 2 definimos una representación natural del mapa de la ciudad por medio de grafos. En la sección 3 mostramos la forma de reducir nuestro problema al Problema del Vendedor Viajero. En la sección 4 se describe brevemente la implementación del programa desarrollado para realizar tareas de procesamiento del mapa, cálculo de itinerarios óptimos y visualización de los resultados. En la sección 5 se presentan los resultados obtenidos, y el trabajo se cierra en la sección 6 con las conclusiones obtenidas y los próximos pasos.

2. Modelo del mapa

En esta sección se describen los datos disponibles para la realización del estudio, y se define en detalle cómo representar el mapa por medio de un grafo, con la intención de que este modelo permita calcular recorridos en vehículo por la ciudad. Esta representación es la base de todo el trabajo de implementación dado que, además del cálculo de recorridos, permite validar la información procesada y depurar los datos incorrectos.

La Unidad de Sistemas de Información Geográfica (USIG) del Gobierno de la Ciudad de Buenos Aires proveyó el mapa de la ciudad en formato Shape (SHP), uno de los formatos standard para la representación de mapas e información geográfica. El archivo contiene una base de datos con cada cuadra de la ciudad. Para cada cuadra se tiene la ubicación de sus esquinas, el nombre de la calle, la altura inicial, la altura final y el sentido. Por otro lado, se conoce la posición de los semáforos en la ciudad.

La información sobre los semáforos contiene un punto geográfico por cada uno de ellos, que indica aproximadamente dónde se encuentra el mismo. Si hay dos calles que se intersecan en el punto p y hay un semáforo en la intersección, entonces en la base de datos de semáforos hay un punto cercano a p. En nuestro modelo, necesitamos saber si en la intersección de dos calles cualesquiera hay un semáforo. Para eso, buscamos si hay un punto en dicha base de datos cuya distancia al punto de intersección de las dos calles sea menor que un cierto valor D. Realizamos varias pruebas con distintos valores de este parámetro y encontramos que $D = 3{,}5$ metros detecta correctamente los semáforos.

2.1. Construcción del grafo y cálculo de recorridos

Describimos a continuación una representación natural del mapa de la ciudad por medio de un grafo, de manera tal que esta representación permita calcular distancias recorridas en vehículo. Dado que necesitamos tener en cuenta el sentido de circulación en las cuadras, trabajamos con un grafo dirigido. Los nodos se definen como las posiciones significativas donde puede ubicarse un vehículo, dados por los extremos de cada cuadra y las posiciones de los contenedores.

Si un vehículo está en una intersección de cuadras, se marca como dos posiciones diferentes cuando está en una u otra cuadra de la intersección. Por otro lado, en las cuadras doble-mano, la posición del vehículo cambia si está en una mano de la cuadra o en la contramano. Luego, la posición del vehículo se define como la cuadra, la mano y las coordenadas geográficas. Existe un arco entre dos nodos que representan dos posiciones si un vehículo puede pasar

directamente de una a otra. Por esta razón, un camino en el grafo representa un recorrido válido de un vehículo en la ciudad. El peso de un arco es la distancia en metros entre las posiciones que representan sus nodos extremos.

En una intersección de dos calles doble-mano, tenemos entonces 8 posiciones: mano y contramano de cada una de las 4 cuadras. Si no hay giros prohibidos, estos puntos se conectan de forma que se pueda girar para un lado y para el otro. Sin embargo, cuando hay un semáforo en la intersección, el vehículo no puede girar a la izquierda. En la figura 2 se ve este caso, mostrando los giros que un vehículo que va por la Av. Rivadavia puede realizar hacia Boyacá. Notar que los 8 puntos que aparecen en la figura tienen las mismas coordenadas geográficas, lo que varía es la cuadra y/o la mano.

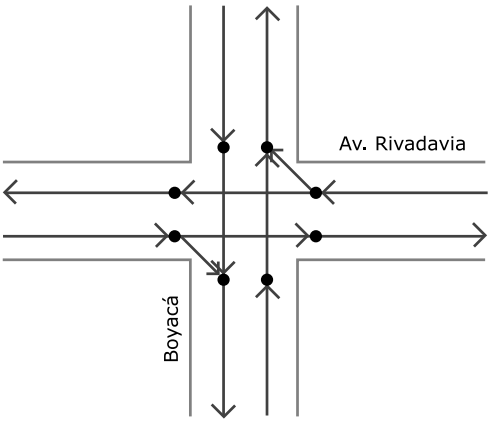


Figura 2: Arcos de giros de Av. Rivadavia a Boyacá con semáforo.

Formalizamos a continuación la definición del grafo $G = (V, A)$ que modela los recorridos en vehículo. Sea $C = \{(s_1, p_1), \ldots, (s_n, p_n)\}$ el conjunto de cuadras de la ciudad, donde $s_i \in S = \{$*creciente, decreciente, doble-mano*$\}$ y $p_i = \left(q_{i_1}, \ldots, q_{i_{t_i}}\right)$ es el conjunto de puntos geográficos de la cuadra i, para $i = 1, \ldots, n$. Los puntos q_{i_1} y $q_{i_{t_i}}$ corresponden a las esquinas, y los puntos $q_{i_2}, \ldots, q_{i_{t_i-1}}$ corresponden a los contenedores ubicados en la cuadra. Los puntos de cada cuadra están ordenados en el sentido creciente de la altura de la cuadra. Si el sentido es creciente, el vehículo puede transitar de q_{i_1} a q_{i_2}, de q_{i_2} a q_{i_3} y así hasta $q_{i_{t_i}}$. Si el sentido es decreciente, el vehículo puede transitar de $q_{i_{t_i}}$ a $q_{i_{t_i-1}}$, de $q_{i_{t_i-1}}$ a $q_{i_{t_i-2}}$ y así hasta q_{i_1}. Si el sentido es doble-mano, el vehículo puede transitar en ambos sentidos.

Sea $Q_i = \left\{q_{i_1}, \ldots, q_{i_{t_i}}\right\}$ el conjunto de puntos de la cuadra c_i (es decir, el vector p_i considerado como un conjunto), y definimos M_i como el conjunto de manos por las que se puede circular en la cuadra:

$$M_i = \begin{cases} \{creciente\} & \text{si } s_i = creciente \\ \{decreciente\} & \text{si } s_i = decreciente \\ \{creciente,\ decreciente\} & \text{si } s_i = doble\text{-}mano \end{cases}$$

Para cada cuadra i, el conjunto de nodos del grafo que origina la cuadra i es $V_i = \{c_i\} \times Q_i \times M_i$, dados por las distintas posiciones que puede ocupar un camión en la cuadra en función de sus puntos geográficos y las manos de la cuadra. El conjunto de nodos de G está dado por $V = \bigcup_{i=1}^{n} V_i$.

Decimos que dos nodos (c_i, q_{i_k}, m) y (c_j, q_{j_l}, m') son *consecutivos* si y sólo si $i = j$, $m = m'$, $m = creciente \Rightarrow l = k + 1$ y $m = decreciente \Rightarrow l = k - 1$. Decimos que un nodo $(c_i, q_{i_k}, m) \in V$ es un *extremo de salida* de la cuadra $c_i = (s_i, p_i)$ con $p_i = \left(q_{i_1}, \ldots, q_{i_{t_i}}\right)$ si y sólo si $m = creciente \Rightarrow k = t_i$ y $m = decreciente \Rightarrow k = 1$. Decimos que un nodo $(c_i, q_{i_k}, m) \in V$ es un *punto de entrada* a la cuadra c_i si es una esquina y no es un extremo de salida de c_i.

Dados dos nodos $v_1 = (c_i, q_{i_k}, m) \in V$ y $v_2 = (c_j, q_{j_l}, m') \in V$, decimos que hay un *giro* de v_1 a v_2 si $q_{i_k} = q_{j_l}$, v_1 es extremo de salida de c_i y v_2 es punto de entrada a c_j. Para definir el ángulo de este giro, consideramos $v_1' = (c_i, q_{i_x}, m) \in V$ y $v_2' = (c_j, q_{j_y}, m') \in V$ tales que v_1' y v_1 son consecutivos y v_2 y v_2' son consecutivos. El *ángulo de giro* es el ángulo entre la semirrecta con origen q_{i_k} y sentido de q_{i_x} a q_{i_k} a y la semirrecta con origen en q_{i_k} y que contiene q_{j_y}. Consideramos que este ángulo está comprendido en el intervalo $(-\pi, \pi]$, con lo cual los ángulos positivos corresponden a giros la izquierda y los ángulos negativos corresponden a giros hacia la derecha.

Para restringir los arcos que corresponden a giros que no puede realizar un vehículo, definimos S como el conjunto de los puntos de salida de las cuadras con semáforos. Dados dos nodos $v_1 = (c_i, q_{i_k}, m) \in V$ y $v_2 = (c_j, q_{j_l}, m') \in V$, decimos que hay un *giro prohibido* de v_1 a v_2 si hay un giro de v_1 a v_2, si $q_{i_k} \in S$ y el ángulo de giro es mayor que $\pi/4$. Dados dos nodos $v_1 \in V$ y $v_2 \in V$, decimos que hay un *giro permitido* de v_1 a v_2 si hay un giro de v_1 a v_2 y no hay un giro prohibido de v_1 a v_2.

Definimos ahora los arcos del grafo considerando si un nodo es consecutivo del otro en la misma cuadra o si se puede girar de un nodo al otro.

$$A = \{(v_1, v_2) \in V \times V : \ v_1 \text{ y } v_2 \text{ son consecutivos, o bien hay un giro permitido de } v_1 \text{ a } v_2\}$$

Como los arcos del grafo corresponden a todos los movimientos permitidos que puede realizar un vehículo, si utilizamos un algoritmo de camino mínimo sobre este grafo, tenemos como resultado un recorrido mínimo que puede realizar un vehículo para ir de una posición a otra en la ciudad.

2.2. Ubicación de los contenedores

Como parte de los datos, contamos con cuatro listas de direcciones de contenedores; cada una corresponde a una sub-zona y las direcciones están ordenadas de la forma en que se recorren actualmente. En la figura 3, pueden verse las cuatro sub-zonas de contenedores actuales.

Estos datos nos permiten calcular la longitud de los itinerarios actuales para comparar con los resultados obtenidos en este trabajo. Aún si no sabemos si el camino que realizan entre cada par consecutivo de contenedores es el mínimo posible, podemos asumirlo y así calcular una cota inferior de la distancia y el trabajo de los itinerarios actuales. Análogamente, podemos asumir que los caminos entre el EHU y el primer contenedor, entre el último contenedor y el depósito, y entre el depósito y el EHU son también mínimos.

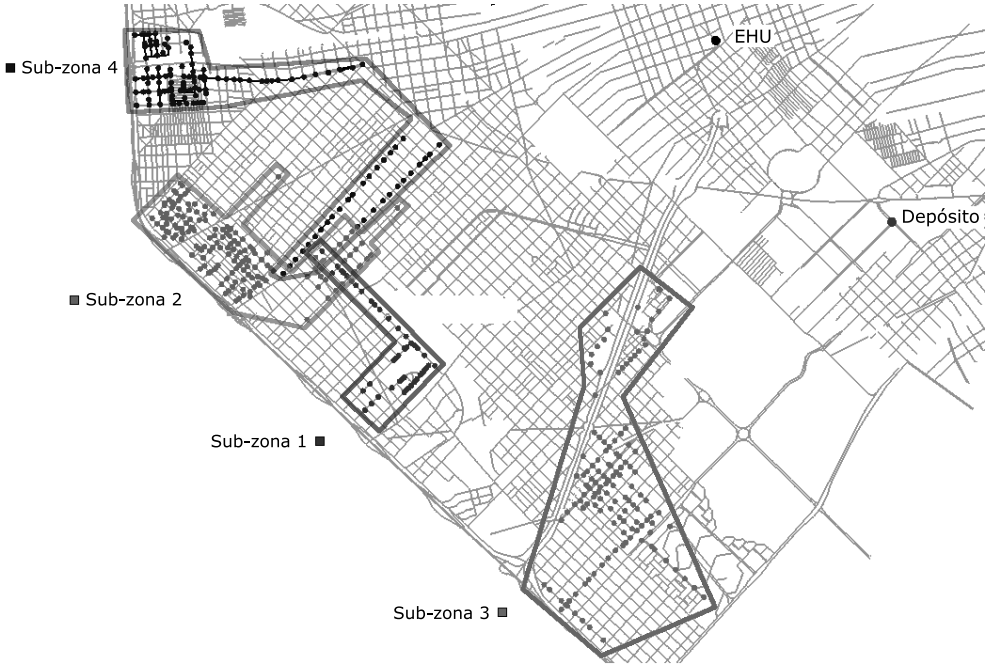


Figura 3: Sub-zonas de contenedores actuales.

Como las listas de contenedores no estaban pensadas para ingresar a un sistema informático, las direcciones no estaban normalizadas. En algunos casos, una dirección se definía por alguna referencia del mapa, por ejemplo nombrando instituciones sin dar su ubicación. Estos casos se corrigieron manualmente en la base de datos. Por otra parte, se detectaron múltiples denominaciones y variaciones de los nombres de las calles, lo cual dificultó su sistematización para el procesamiento automático. Para solucionar este problema, se reemplazó manualmente cada nombre con su correspondiente en la base de datos.

El resultado de esta traducción son listas de direcciones de los contenedores dadas por nombre de la calle y altura o por intersección de calles. Una vez completado este proceso, es necesario traducir cada dirección a una posición del mapa *(punto, cuadra, mano)*, con el objetivo de generar dentro del grafo que representa la ciudad los nodos correspondientes a los contenedores.

Para una dirección por altura, se busca la cuadra de la calle cuyo intervalo [*altura inicial*, *altura final*] incluye la dirección en cuestión. En las cuadras doble-mano, se le asigna la mano que va en sentido creciente de altura o la otra mano según si la altura es impar o par, respectivamente.

Para una dirección denotada por intersección de calles, se buscan las cuadras de ambas calles que se intersecan en un punto. En el caso más simple, hay cuatro cuadras que se intersecan en un punto: dos de una calle y dos de la otra. Es necesario elegir una de las cuatro cuadras para definir la posición del contenedor. El criterio que usó el EHU para confeccionar las listas es que la dirección nombra primero la calle sobre la cual está el contenedor y el mismo se encuentra antes de la intersección. Para resolver las ambigüedades que se presentan cuando la primera calle es doble-mano, entre los dos puntos posibles (mano y contramano) se escoge el de menor coordenada x y, si ambas coordenadas x son iguales, se escoge el de menor coordenada y. Este criterio fue definido arbitrariamente.

Como toda base de datos que contiene información de la realidad, el mapa tiene fallas. Se verificaron muchas de las calles que están en las zonas donde hay contenedores para revisar los sentidos. Se revisaron especialmente las que se utilizan en los recorridos óptimos que encontramos. Para eso se comparó con otros mapas digitales y con fotos aéreas, que permitieron validar los sentidos de las calles. Este proceso manual permitió depurar los datos, con el objetivo de contar con información lo más precisa posible para el proceso de optimización.

3. Estrategia de resolución

Dado que el primer objetivo es minimizar la distancia recorrida por cada camión, la opción más natural es transformar el problema de diseñar el recorrido de cada camión en una instancia adecuada del Problema del Vendedor Viajero (TSP). El TSP consiste en encontrar un circuito Hamiltoniano de longitud mínima en un grafo, y es un problema NP-hard. El segundo objetivo es disminuir el trabajo total realizado por el camión (como aproximación del desgaste total). Como las instancias generadas resultan tener una gran cantidad de óptimos alternativos para el TSP, entonces se resuelve muchas veces en forma óptima el TSP utilizando una estrategia no determinística, y se selecciona la solución que realiza el menor trabajo total. Para que este esquema tenga éxito, es necesario contar con una implementación que permita resolver el TSP en

forma óptima y no determinística. En nuestra implementación utilizamos el software *Concorde* [1].

Se describe en esta sección el proceso de generación del grafo y resolución del TSP. En la Sección 3.1 se describe la construcción de una instancia del problema de camino Hamiltoniano mínimo entre dos nodos particulares, que modela nuestro problema. En la Sección 3.3 se describe el uso del software *Concorde*, que resuelve el TSP sobre grafos no dirigidos. Por este motivo, es necesario transformar el problema en un TSP simétrico, proceso que se describe en la Sección 3.4 y la Sección 3.5.

3.1. Construcción de la instancia

Para modelar el problema adecuadamente, construimos un digrafo completo con los contenedores como nodos. El peso de un arco del nodo A al nodo B se define como la distancia del recorrido mínimo en vehículo del contenedor A al contenedor B (Ver Sección 3.2). Agregamos a este grafo el EHU y el Depósito. Agregamos un arco desde el EHU hacia cada contenedor, y desde cada contenedor al depósito. Los pesos de estos arcos también quedan definidos por la distancia del recorrido mínimo de un elemento al otro. Llamamos a este grafo $G_1 = (V_1, A_1)$ y denotamos por $w_1 : A_1 \to \mathbb{R}$ la función de distancia que a cada arco le asocia la distancia del recorrido correspondiente. Puede verse un ejemplo en la figura 4.

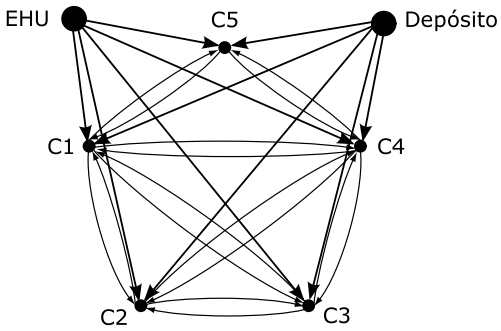


Figura 4: G_1. Grafo con el EHU, cinco contenedores y el depósito.

Nuestro objetivo es buscar un camino Hamiltoniano mínimo en este grafo, que parta del nodo que representa al EHU, pase por todos los nodos que representan contenedores, y termine en el nodo que representa al depósito.

3.2. Algoritmo de camino mínimo

Una vez definido el grafo asociado al mapa de la ciudad aplicamos la variante A^* del algoritmo de Dijkstra [3, 6] para obtener el camino mínimo en

vehículo entre cada par de nodos en el digrafo completo que tiene a los contenedores, al EHU y al depósito como nodos. El problema que surge al aplicar el algoritmo de Dijkstra sin dicha variante es que durante su ejecución el conjunto de nodos pendientes de analizar se expande en una forma similar a un rombo con el punto de origen como centro.

Este comportamiento hace que sea menos eficiente cuanto mayor sea la distancia entre los nodos a analizar. Por ejemplo, para calcular un recorrido mínimo entre dos puntos que dé como resultado 10 km., calculará los caminos mínimos a todos los puntos que estén a menos de 10 km. del origen. En el grafo de la ciudad, este cálculo demora más de una hora realizado con una implementación en SmallTalk en una computadora Intel Dual Core de 1.60 GHz.

Este problema se soluciona usando la variante A*, ya que dicha heurística cambia la forma de elegir el siguiente nodo a analizar. En lugar de tener en cuenta sólo la distancia al origen, también tiene en cuenta la distancia al destino. Para que funcione correctamente, utiliza la propiedad de que la distancia del recorrido es siempre mayor o igual que la distancia euclidiana entre ambos puntos. Luego, en un punto intermedio del recorrido, al cual ya le calculó el recorrido mínimo, se deduce que la distancia del recorrido mínimo del origen al destino es mayor o igual que la suma de la distancia del recorrido del origen al intermedio, más la distancia euclidiana.

Cuando se hace el mismo cálculo, en la misma computadora, de un recorrido de más de 10 km. utilizando la variante A* se reduce el tiempo de ejecución drásticamente a 280 milisegundos, lo cual verifica que la elección de la misma es satisfactoria para este trabajo.

3.3. Utilización de Concorde

Concorde [1] es un programa realizado por David Applegate, Robert Bixby, Vašek Chvátal y William Cook en el Instituto de Tecnología de Georgia (Georgia Institute of Technology), que permite resolver instancias del TSP en forma exacta mediante programación lineal entera. Para esto se le adjunta un paquete para resolver problemas de programación lineal, por ejemplo *QSOpt* (de libre uso y desarrollado por los mismos autores). *Concorde* puede resolver instancias de hasta 1000 elementos en forma eficiente. Por ejemplo, una instancia de nuestro problema con 98 nodos es resuelta en 200 milisegundos en una computadora Intel Dual Core de 1.60 GHz.

Concorde también permite resolver el TSP mediante la heurística Chained Lin-Kernighan [9]. Esta heurística es muy eficiente y proporciona resultados óptimos o muy cercanos al valor óptimo para instancias pequeñas. Por ejemplo, para la instancia de 98 nodos que consideramos en este trabajo, el circuito obtenido resulta ser el óptimo y el tiempo de ejecución es similar al tiempo de

resolución del algoritmo exacto.

Por otro lado, como *Concorde* utiliza aleatoriedad para obtener una solución, si lo ejecutamos varias veces, se obtienen distintos óptimos alternativos.

3.4. Transformación de camino Hamiltoniano a circuito Hamiltoniano

En la Sección 3.1 modelamos nuestro problema como un problema de camino Hamiltoniano mínimo dirigido. Sin embargo *Concorde* resuelve el TSP, es decir el problema de circuito Hamiltoniano mínimo no dirigido y recibe como entrada una matriz de distancias de un grafo completo. Por lo tanto, debemos traducir nuestro modelo de camino Hamiltoniano mínimo dirigido a un circuito Hamiltoniano mínimo no dirigido. Lo haremos en dos pasos: primero pasaremos de un camino dirigido a un circuito dirigido y luego de éste a un circuito no dirigido.

Calcularemos el camino Hamiltoniano mínimo en función del circuito Hamiltoniano. Necesitamos que en este circuito luego del depósito se ubique el EHU. Es decir, a partir del nodo del EHU, se pase por todos los contenedores, luego por el depósito y nuevamente por el EHU. Además, queremos que el grafo sea completo. Para esto, construimos el grafo completo $G_2 = (V_2, A_2)$ a partir de G_1 tomando $V_2 = V_1$ y $A_2 = V_2 \times V_2$. Definimos una función de peso $w_2 : A_2 \to \mathbb{R} \cup \{+\infty\}$ de la siguiente forma:

$$w_2((v_1, v_2)) = \begin{cases} w_1((v_1, v_2)) & \text{si } (v_1, v_2) \in A_1 \\ 0 & \text{si } v_1\text{=EHU y } v_2\text{=Depósito, o viceversa} \\ +\infty & \text{si } (v_1, v_2) \notin A_1 \end{cases}$$

De esta forma, si no hay un arco entre dos nodos de G_1, agregamos un arco en G_2 con peso infinito y así G_2 es completo. De esta forma, todo camino Hamiltoniano de G_1 que comience en el EHU y termine en el depósito genera un circuito Hamiltoniano en G_2 con distancia finita, y viceversa.

3.5. Transformación de grafo dirigido a grafo no dirigido

En esta sección reduciremos el problema de encontrar un circuito Hamiltoniano mínimo de un grafo dirigido, al problema de encontrar un circuito Hamiltoniano mínimo de un grafo no dirigido [8]. Para esto definimos el grafo $G_3 = (V_3, A_3)$ con un nodo ficticio y un nodo real por cada nodo de G_2. Más formalmente, supongamos que $V_2 = \{v_1, \ldots, v_p\}$ y definamos $F = \{f_1, \ldots, f_p\}$, de modo tal que el nodo f_i es el nodo ficticio asociado a v_i. Definimos $V_3 = V \cup F$ y $A_3 = V_3 \times V_3$. Sea $M \in \mathbb{R}$ un número suficientemente grande, y definimos la función de pesos asociados a las aristas $w_3 : A_3 \to \mathbb{R} \cup \{+\infty\}$ del siguiente modo:

$$w_3(x,y) = \begin{cases} +\infty & \text{si } x \in V \wedge y \in V, \\ +\infty & \text{si } x \in F \wedge y \in F, \\ -M & \text{si } \exists i / x = v_i \wedge y = f_i, \\ w_2(v_i, v_j) & \text{si } \exists i / x = v_i \wedge y = f_j \wedge i \neq j \end{cases}$$

En la figura 5 se presenta un ejemplo donde se define un grafo no dirigido G_3 a partir de un grafo dirigido G_2, con $M = 10000$. En G_3 se tienen 3 nodos reales y 3 nodos ficticios. Las aristas entre nodos reales tienen valor $+\infty$, lo que implica que los caminos mínimos no utilizarán esas aristas. Lo mismo sucede para las aristas entre nodos ficticios. Un camino mínimo en este grafo siempre alternará nodos ficticios con reales. Además, como las aristas entre un nodo real y su correspondiente nodo ficticio tienen el valor $-M$, los caminos mínimos siempre utilizarán esas aristas. Como el camino mínimo no contendrá los arcos con peso infinito y sí contendrá los que tienen peso negativo, a continuación de un nodo real siempre aparecerá su nodo ficticio en un camino mínimo.

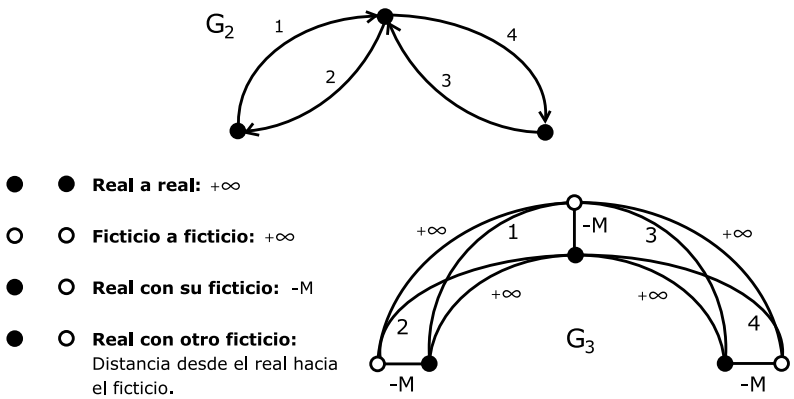
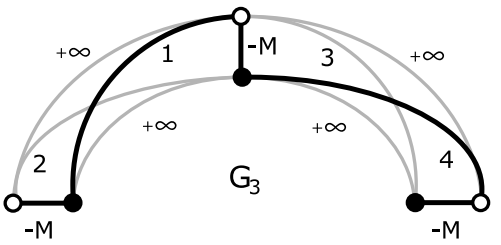


Figura 5: Modelo de grafo dirigido con un grafo no dirigido.

El peso de las aristas entre un nodo real v_i y uno ficticio f_j con $i \neq j$ es el peso del arco que va de v_i a v_j en G_2. Esto implica que el conjunto de aristas del nodo real v_i en G_3 representa los arcos de salida de v_i en G_2. Recíprocamente el conjunto de aristas del nodo ficticio f_j en G_3 representa los arcos de entrada de v_j en G_2. Entonces, un camino $(f_{i_1}, v_{i_1}, f_{i_2}, v_{i_2}, \ldots, f_{i_k}, v_{i_k})$ en G_3 se interpreta como un camino $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ en G_2. Se puede ver un ejemplo en la figura 6.

Por medio de los procedimientos descritos en esta sección, pasamos entonces de un grafo dirigido G_1 en el que queremos calcular un camino Hamiltoniano mínimo, a un grafo dirigido completo G_2 tal que un circuito Hamiltoniano mínimo de G_2 nos permite calcular el camino Hamiltoniano mínimo



Longitud con arcos ficticios: $-3M + 5$
Longitud sin arcos ficticios: 5

Figura 6: Camino en un grafo no dirigido.

de G_1. Luego pasamos a G_3, un grafo no dirigido completo. Si calculamos el circuito Hamiltoniano mínimo de G_3 podemos obtener fácilmente el circuito Hamiltoniano mínimo de G_2. De esta forma reducimos nuestro problema al Problema del Vendedor Viajero en un grafo completo no dirigido y podemos utilizar *Concorde* para obtener el resultado eficientemente.

4. Descripción de la implementación

Se implementó un programa en el lenguaje de programación *SmallTalk* que realiza tareas de procesamiento del mapa, cálculo de itinerarios óptimos y visualización de los resultados. En este proyecto utilizamos la implementación *VisualWorks NonCommercial, 7.4.1* disponible sobre los sistemas operativos *Microsoft Windows XP/Vista* y *GNU-Linux*.

El programa consta de un conjunto de objetos, de clases y de métodos que modelan el mapa de la ciudad, el grafo de los contenedores, los algoritmos necesarios para calcular los caminos mínimos y los itinerarios mínimos. También incluye interfaces con una base de datos del mapa en *PostgreSQL*, con el programa *Concorde* y con el paquete de visualización gráfica *Takenoko*.

El sistema realiza las siguientes tareas para procesar la información, calcular los itinerarios mínimos y visualizarlos:

1. Lee la base de datos del mapa de la ciudad.
2. Almacena las direcciones de los contenedores de cada itinerario.
3. Para cada una de las 4 sub-zonas realiza los siguientes pasos:
   a) Calcula el recorrido mínimo en vehículo entre cada par de elementos: ente, contenedores y depósito.
   b) Calcula la distancia y el trabajo del itinerario actual.

c) Construye el grafo G_1 con las distancias entre los elementos.

d) Construye el grafo G_2 a partir de G_1 como se define en la Sección 3.4.

e) Construye el grafo completo G_3 a partir de G_2 como se define en la Sección 3.5.

f) Ejecuta *Concorde* con la matriz de distancias de G_3.

g) Interpreta el resultado para construir un itinerario de distancia mínima.

4. Además, para cada itinerario se permite:

a) Calcular su distancia.

b) Calcular su trabajo.

c) Generar una lista con el orden de los contenedores.

d) Generar una lista de las cuadras por las que pasa el camión.

e) Visualizar una imagen del mapa con el recorrido marcado.

f) Visualizar una animación del recorrido del camión sobre el mapa.

5. Resultados y discusión

En esta sección reportamos los resultados obtenidos para las cuatro instancias de nuestro problema, correspondientes a las cuatro sub-zonas, que contienen 47, 133, 138 y 161 contenedores, respectivamente. A su vez, comparamos las distancias del itinerario que utiliza actualmente el EHU con la distancia de un itinerario de distancia mínima. También comparamos el trabajo ejercido en ambos itinerarios.

En las 4 instancias se obtienen a través de Concorde los recorridos mínimos. Utilizando la aleatoriedad del software se realizan múltiples corridas de cada caso, para luego quedarnos con el camino mínimo que da el menor trabajo.

En la siguiente tabla se muestran los resultados calculados para cada uno de los itinerarios. Se reportan la distancia y el trabajo para el itinerario que actualmente realiza el EHU y para el itinerario de distancia mínima (los casos reportados son siempre los que dan el menor trabajo posible dentro de los múltiples recorridos mínimos obtenidos). Se ven importantes mejoras tanto en la distancia, como en el trabajo.

Vemos que hay una diferencia muy grande entre la mejora de la sub-zona 1 y la mejora de la sub-zona 4 que radica en la complejidad de las mismas. La primera tiene menos que un tercio de la cantidad de contenedores de la segunda y sus contenedores están sobre sólo 7 calles distintas. Estas características

		Recorrido actual		Nuestra propuesta		Porcentaje de mejora	
	Contenedores	Distancia	Trabajo	Distancia	Trabajo	Distancia	Trabajo
1	47	27.010	$6,08 \times 10^8$	24.126	$5,50 \times 10^8$	10,68 %	9,52 %
2	133	68.102	$4,77 \times 10^9$	41.752	$2,98 \times 10^9$	38,69 %	38,14 %
3	134	52.270	$4,28 \times 10^9$	39.762	$2,86 \times 10^9$	23,93 %	33,23 %
4	161	61.692	$5,44 \times 10^9$	40.841	$3,11 \times 10^9$	33,80 %	42,80 %

Tabla 1: Tabla de resultados

hacen que un itinerario cercano al mínimo sea bastante intuitivo. En cambio, en la sub-zona 4 hay un área con mucha densidad de contenedores y el método intuitivo no es practicable. Por esta razón, un itinerario de distancia mínima provoca una gran mejora (33,80 %).

Con respecto a los tiempos de resolución, la mayor parte la insume el cálculo de recorridos mínimos entre cada par de elementos. En la sub-zona 4, demora 40 minutos la primera vez y sólo 7 segundos cuando los recorridos ya están calculados. Si agregamos o movemos un contenedor del recorrido, sólo hará falta calcular 162 recorridos, lo que insumiría alrededor de 24 segundos según cuál sea la posición nueva del contenedor. El tiempo que demora el cálculo de los caminos mínimos no sólo depende de la cantidad de caminos, sino también de la longitud de los mismos. Cuanto más dispersos están los contenedores, habrá caminos más largos y, por lo tanto, este cálculo demorará más.

Los resultados nos permiten concluir que el problema fue abordado en forma satisfactoria ya que la distancia y el trabajo totales se redujeron en forma muy importante. En el caso de contar con información de las velocidades de las calles, se esperaría una mejora similar en la duración de los itinerarios.

6. Conclusiones y próximos pasos

En esta aplicación vemos niveles de mejora significativos en los recorridos de los camiones. La distancia de los itinerarios se reduce hasta un 39 % y el trabajo, aunque no es la variable que optimiza el modelo, también se redujo hasta un 43 % por tener a la distancia como uno de sus factores.

Una buena parte del trabajo realizado consistió en modelar el grafo e implementar el algoritmo de camino mínimo, considerando todos los detalles para producir recorridos en vehículo válidos en el mapa de la ciudad. La interfaz gráfica para producir imágenes y animaciones es un elemento relevante de la implementación, dado que permite visualizar la información de diversas formas, característica fundamental al tener gran cantidad de información. Tanto la interfaz gráfica como la capa de datos fueron abstraídas en objetos pro-

pios del sistema, por lo cual las partes que cambiarían en caso de elegir otras tecnologías están bien localizadas.

Una posible continuación del proyecto sería desarrollar un software que permita el cálculo de itinerarios en tiempo real. Al ocurrir eventos como cortes de calles, congestiones o manifestaciones, se podrían recalcular los itinerarios en el momento. La eficiencia de los algoritmos utilizados permite esta posibilidad en forma directa.

También podría agregarse un módulo de localización por GPS que permita ingresar al programa la posición de los camiones y, mediante dispositivos móviles, indicar a los conductores los próximos contenedores a recolectar. Estos agregados hacen que el sistema sea escalable a la hora de agregar nuevos contenedores y vehículos.

Para disminuir aún más el desgaste de los vehículos, podría considerarse el problema de optimizar el trabajo total realizado a lo largo del itinerario. Para esto, se pueden diseñar heurísticas que busquen itinerarios alternativos partiendo de la solución óptima del TSP.

El proyecto de recolección de residuos por medio de contenedores prevé agregar contenedores anualmente para llegar a un contenedor por cuadra en toda la ciudad. Por esta razón, será necesario agregar nuevas zonas con sus respectivos itinerarios. El problema de definir la zona asignada a cada camión es un nuevo problema que resulta muy interesante. La solución a este problema puede utilizar la suma de las distancias de los itinerarios mínimos como forma de validar la zonificación.

Si al mapa de la ciudad le agregamos la información de las velocidades de circulación promedio en las cuadras, podemos saber cuánto demora el camión en transitar una cuadra. De esta forma, podemos utilizar el programa para calcular un itinerario de tiempo mínimo. Para esto, sería necesario modificar la implementación del algoritmo A^* para que utilice otra función de cota inferior, como por ejemplo la distancia euclídea multiplicada por la velocidad máxima de circulación.

Como conclusión general, se puede afirmar que las herramientas de investigación de operaciones propuestas deberían ser de suma utilidad si se aplican para implementar un sistema que organice la recolección de residuos de la Ciudad.

Agradecimientos: A Bill Cook, por sus múltiples sugerencias acerca del uso de Concorde. A Daniel Iglesias, Jorge Takashashi y Julián Dunayevich, por proveernos de toda la información necesaria de la Ciudad de Buenos Aires. A Florencia Fernández Slezak, por su colaboración en la realización de este trabajo. Parcialmente financiado por los proyectos UBACyT X069 y X606, ANPCyT PICT-2007-0518 y PICT-2007-0533 (Argentina), FONDECyT 1080286 e Instituto de Ciencias Milenio "Sistemas Complejos de Ingeniería" (Chile).

Referencias

[1] Applegate D. L., Bixby R. E., Chvatal V., and Cook W.J. Who's Who in Networks: The Key Player. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, New Jersey, 2006.

[2] Chang N., Lu H., and Wei L.. GIS technology for vehicle routing and scheduling in solid waste collection systems. *Journal of Environmental Engineering*. Vol 123: 901-933. 1997.

[3] Dijkstra E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*. Vol 1(1): 269-271. 1959.

[4] Eisenstein D. and Iyer A. Garbage collection in Chicago: a dynamic scheduling model. *Numerische Mathematik*. Vol 43: 922-933. 1997.

[5] Golden B., Assad A., and Wasil E.. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. *In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia, PA: SIAM*. 245-286. 2002.

[6] Hart P. E., Nilsson N. J., and Raphael B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*. Vol 4(2): 100-107. 1968.

[7] Kim B., Kim S., and Sahoo S. Waste collection vehicle routing problem with time windows. *Computers and Operations Research*. 3624-3642. 2006.

[8] Kumar R. and Li H. On Asymmetric Tsp: Transformation to Symmetric Tsp and Performance Bound. Manuscript. 1996.

[9] Martin O., Otto S. W., and Felten E. W. Large-step Markov Chains for the Traveling Salesman Problemws. *Complex Systems*. Vol 5: 299-326. 1996.

[10] Mourao M. and Almeida M. Almeida, Lower-bounding and heuristic methods for a refuse collection vehicle routing problem. *European Journal of Operational Research*. Vol 121: 420-434. 2000.