

Problema de recolección y entrega con restricciones de incompatibilidad

Pablo Factorovich, Isabel Méndez Díaz, Paula Zabala

1) Recolección y entrega

El Problema de Recolección y Entrega es del tipo de ruteo de vehículos y presenta aplicaciones en el **transporte de mercadería punto a punto**. En nuestra versión del problema tenemos n **pares de vértices** (*pick+delivery*) y un **vehículo** cuya tarea consiste en tomar cada elemento de un vértice *pick* y depositarlo en su respectivo *delivery*. El objetivo es encontrar el **camino más corto** que inicia en un depósito origen y termina en otro destino, cumpliendo las n tareas asignadas.



2) Variante estudiada y motivación

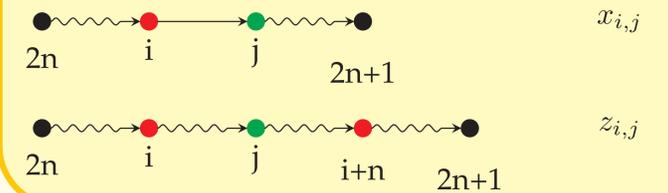
Legislación de distintos países establece que **ciertos elementos no pueden ser transportados simultáneamente por un vehículo** debido a distintos peligros como ser explosión o toxicidad (por ejemplo, en el caso de comida y detergentes). Determinamos un grafo I , donde dos tareas están unidas por un eje si y solo si son incompatibles y, por lo tanto, no pueden estar simultáneamente en el vehículo **pero sí en distintos momentos del recorrido**. Esta variante **no ha sido estudiada aún** en la literatura científica a pesar de ser mencionada en patentes. Tanto el problema original como esta variante son problemas **NP-Completo**s y por lo tanto no se conocen algoritmos polinomiales para resolverlos.

3) Variables

Los vértices *pick* son numerados del 0 al $n - 1$ y los *delivery* del n al $2n - 1$, siendo $i + n$ el *delivery* del *pick* i . El depósito origen se etiqueta como $2n$ y el destino como $2n + 1$ y entre todos constituyen el conjunto V de vértices del grafo G . Para cada par de vértices $i, j \in G$, nos referimos a su distancia como $c_{i,j}$

Definimos los siguientes dos conjuntos de variables **binarias**

- $x_{i,j} = 1 \iff$ el vértice j es visitado inmediatamente después del vértice i en la solución.
- $z_{i,j} = 1 \iff$ en la solución, el vértice j es visitado en algún momento entre i y su *delivery* $i + n$. **Se fija en 0 para i y j incompatibles.**



4) Modelo

minimizar

$$\sum_{i,j} c_{ij} x_{ij}$$

s.a.

$$\sum_{i/(i,j) \in E(G)} x_{ij} = 1 \quad \forall j \in V$$

$$\sum_{j/(i,j) \in E(G)} x_{ij} = 1 \quad \forall i \in V$$

$$x_{i,j} \leq z_{i,j} \quad \forall i \in P, j \in P \cup D, i \neq j$$

$$x_{j,i+n} \leq z_{i,j} \quad \forall i \in P, j \in P \cup D, i \neq j$$

$$x_{ij} + z_{ki} \leq z_{kj} + 1 \quad \forall i \neq j \neq k \in V(G), i \neq k + n$$

$$x_{ij} + z_{kj} \leq z_{ki} + 1 \quad \forall i \neq j \neq k \in V(G), j \neq k + n$$

$$X(S) \leq |S| - 1 \quad \forall S \subset V, |S| > 2, O \notin V$$

5) Branch & cut

Buscamos una solución óptima por medio de un árbol de búsqueda *branch and bound*. Aquí cada nodo hijo representa un subconjunto de las soluciones representadas por su padre siendo disjunto con los subconjuntos representados por sus hermanos. Seguimos la estrategia más usada de ramificar cada nodo tomando una variable binaria y fijandola en 0 y 1 respectivamente para cada uno de sus hijos. Para reducir la búsqueda de nodos es muy importante obtener **buenas cotas superiores e inferiores** del valor de la solución óptima. Proveemos las últimas a través de un buen **modelo** y **cortes** específicos para este. Las cotas superiores se construyen a través de **heurísticas** primales.

6) Cotas Superiores

Las cotas superiores se consiguen a través de una **heurística** para el problema que se basa en las relaciones lineales conseguidas en los distintos nodos (heurística primal). En primer lugar nuestro algoritmo **construye una solución factible** "muy similar" a la relajación lineal. Se realiza **luego una búsqueda local** tomando cada vértice y desplazándolo a cada posición del camino armado en la fase constructiva que lleva a una situación factible.

7) Cotas Inferiores

Las cotas se obtienen aceptando valores en el intervalo $[0, 1]$ para las variables binarias (**relajación lineal**). La calidad de la relajación estará dada por el **modelo** elegido y por restricciones extras (igualdades y desigualdades) que, con el fin de fortalecer el modelo, se agregan en los nodos del árbol *branch & cut* (**cortes**). Fueron identificadas 6 familias de igualdades para el modelo y se probó que no existen otras (que sean linealmente independientes de estas). Por otra parte se encontraron 23 familias de desigualdades que creemos que no pueden ser fortalecidas (**facet**s). Algunos ejemplos de las desigualdades introducidas para fortalecer el modelo incluyen las siguientes familias:

$$X_{i,i+n} + X_{i,j+n} + Z_{i,j} \leq 1 \quad \forall i \in P, j \in P/\{i\}, (i,j) \notin E_I$$

$$Z_{i,j} + Z_{j,k} + Z_{k,i} + X_{i,k} + X_{i+n,j} + X_{j+n,k} + X_{k+n,i} \leq 2$$

$$\forall i \in P, j \in P/\{i\}, k \in P/\{i,j\}, (i,j), (j,k), (k,i) \notin E_I$$

8) Branching

La idea consiste en tener, en cada nodo del árbol, un camino parcial desde un depósito. La variable seleccionada para ramificar es alguna que permite extender un camino ya formado por aquellas variables fijadas en 1 en los nodos ancestros del árbol.

9) Resultados

Realizamos pruebas con distinto grado de incompatibilidad para mostrar el efecto de incluir nuestros desarrollos dentro del algoritmo:

Inc.(%)	Solo modelo		Con cortes		Con cortes + heurística		Con cortes, heur.& branch	
	Time(s)	#Nodes	Time(s)	#Nodes	Time(s)	#Nodes	Time(s)	#Nodes
10	1010,6	20423,64	863,36	16598,60	402,64	3195,84	154,68	1235,6
25	339,08	10153,52	271,92	7519,84	129,44	1844,64	104,88	1133,84
50	16,96	1157,08	19,12	979,24	11,96	407,24	60,88	1269,36
70	0,6	121,94	1,04	108,92	0,52	43,18	8,28	435,12