Algoritmos Basados en Programación Lineal Entera para Problemas de Optimización Combinatoria

Trabajo Práctico

Nazareno Faillace

Paper: "A column generation approach to the heterogeneous fleet vehicle routing problem" por Eunjeong Choi y Dong-Wan Tcha.

Introducción

Este informe tiene como objetivo exponer e implementar el trabajo realizado por los autores del paper y discutir sus resultados. La primera sección está dedicada a presentar los conceptos necesarios para entender el algoritmo propuesto por los autores. A grandes rasgos, se presentan las definiciones de modelo de Programación Lineal, de su problema dual, de Programación Lineal Entera y se explica en qué consisten los método de generación de columnas y Branch & Bound. La segunda sección está destinada a presentar el problema, explicar el algoritmo desarrollado por los autores y se discuten los resultados obtenidos. Finalmente en la tercera sección se presenta la implementación llevada a cabo por el autor del informe.

1. Conceptos básicos

Programación Lineal

Comencemos con una definición imprescindible para formular un modelo de Programación Lineal:

Definición 1 (Función lineal)**.** Sean $c_0, c_1, \ldots, c_n \in \mathbb{R}$ y sea $f\colon \mathbb{R}^n \to \mathbb{R}$ definida como $f(x_1, \ldots, x_n) = c_0 + \sum_{j=1}^{n} c_j x_j$, decimos que f es una función lineal.

Sean f una función lineal y $b \in \mathbb{R}$, diremos que la expresión $f(x_1, \ldots, x_n) = b$ es una ecuación lineal. Por otro lado, denominaremos desigualdades lineales a las expresiones de la forma $f(x_1, \ldots, x_n) \leq b$ y $f(x_1, \ldots, x_n) \geq b$. Así pues, un problema de programación lineal consiste en hallar el máximo (o el mínimo) de una función lineal sujeta a un conjunto de restricciones lineales. La función a optimizar es denominada función objetivo. Dado que el término independiente de una función lineal no influye en dónde se realiza su óptimo y, en el caso de las ecuaciones e desigualdades se puede pasar restando, por simplicidad consideraremos que las funciones lineales tienen término independiente nulo. De aquí en adelante abreviaremos programación lineal con PL.

En la literatura hay diferentes versiones de lo que es la escritura estándar de un problema de PL. En esta

sección, para un problema con n variables y m restricciones, usaremos la siguiente formulación estándar:

$$\begin{array}{lrcl} \text{máx} & \displaystyle\sum_{j=1}^{n} c_j x_j & & \\ s.a: & \displaystyle\sum_{j=1}^{n} a_{1j} x_j & \leq & b_1 \\ & \vdots & & \vdots \\ & \displaystyle\sum_{j=1}^{n} a_{mj} x_j & \leq & b_m \\ & x_1 & \geq & 0 \\ & \vdots & & \vdots \\ & x_n & \geq & 0 \end{array}$$

Considerando la siguiente notación:

$$c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

con un poco de abuso de notación, podemos escribir sucintamente la formulación estándar del problema de la siguiente manera:

$$\begin{array}{lrcl} \text{máx} & c^T x & & \\ s.a: & Ax & \leq & b \\ & x & \geq & 0 \end{array} \tag{PL}$$

Definición 2 (Solución factible)**.** Sea $\bar{x} \in \mathbb{R}^n$ fijo, diremos que $\bar{x}$ es solución factible de (PL) si cumple que $A\bar{x} \leq b$ y que $\bar{x}_j \geq 0$ para cada $j = 1, \dots, n$. Denominaremos región de soluciones factibles a $S = \{x \in \mathbb{R}^n : Ax \leq b \ \wedge \ x \geq 0\}$.

Definición 3 (Solución óptima)**.** Dado (PL) y sea $x^* \in S$, diremos que x^* es solución óptima de (PL) si y sólo si $c^T x^* \geq c^T x \ \ \forall x \in S$. En ese caso, denominaremos valor óptimo a $c^T x^*$.

Definición 4 (Problema infactible)**.** Dado (PL), se lo denomina infactible si $S = \emptyset$.

A continuación, presentaremos una serie de definiciones y enunciaremos teoremas que brindan características muy importantes para resolver problemas de PL. Las demostraciones de los teoremas se omiten en pro de no extender demasiado el informe.

Definición 5. Sea $T \subset \mathbb{R}^n$, diremos que T es un conjunto convexo si dados dos puntos cualesquiera $x, y \in T$, el segmento que los une está contenido en T. En otras palabras T es convexo si, dados $x, y \in T$, vale que $z = \lambda x + (1 - \lambda y) \in T \quad \forall \lambda \in [0, 1]$. En la Figura 1 se muestran ejemplos de un conjunto convexo y un conjunto no convexo.

Teorema 1. Sea S la región de soluciones factibles de (PL), si $S \neq \emptyset$, entonces S es un conjunto convexo.

Definición 6 (Poliedro)**.** Sea $T \subset \mathbb{R}^n$ un conjunto formado por la intersección de subespacios afines, se dice que T es un poliedro. Notar que, como cada una de las restricciones de (PL) es una desigualdad lineal, cada una define un subespacio afín, por lo tanto, S es un poliedro.

Definición 7 (Cápsula convexa)**.** Sea $T \subset \mathbb{R}^n$ diremos que su cápsula convexa es el menor conjunto convexo que lo contiene y lo notaremos $e(T)$. Si T es un conjunto finito de puntos, $e(T)$ se denomina polítopo. En la Figura 2 se exhibe un ejemplo.

(a) Conjunto convexo

(b) Conjunto no convexo

Figura 1: Ejemplo de un conjunto convexo y uno no convexo. En (a) podemos tomar dos puntos cualesquiera y el segmeneto que los une estará incluido en el conjunto. En (b) el segmento que une a x con y (línea roja) no está incluido en el conjunto.

(a) $T = \{x, y, w, z\} \subset \mathbb{R}^2$

(b) $e(T)$

Figura 2: Conjunto finito en $\mathbb{R}^2$ y su polítopo.

Definición 8 (Conjunto convexo acotado)**.** Sea $T \subset \mathbb{R}^n$ convexo, T se dice no acotado si $\forall x \in T$ existe $d \in \mathbb{R}^n$, $d \neq 0$ tal que $x + \lambda d \in T \ \ \forall \lambda > 0$. De lo contrario, se dice que T es acotado.

Definición 9 (Punto extremo)**.** Sean $T \subset \mathbb{R}^n$ y $x \in T$, decimos que x es un punto extremo o vértice de T si x no puede escribirse como combinación lineal convexa de otros dos puntos de T. En otras palabras, x es punto extremo si no existen $y, z \in T$ tales que $x = \lambda z + (1 - \lambda)y$ para algún $\lambda \in (0, 1)$.

Teorema 2. Sea $T \subset \mathbb{R}^n$ convexo y acotado, entonces tiene una cantidad finita de puntos extremos $E = \{x^{(1)}, \ldots, x^{(p)}\}$ y $T = e(E)$.

Teorema 3. Si (PL) admite solución óptima, entonces el valor óptimo se alcanza en alguno de los puntos extremos del poliedro factible S. Si el valor óptimo se alcanza en varios vértices de S, entonces se alcanza en cualquier punto que sea combinación lineal convexa de ellos.

Se puede demostrar que cualquier problema de Programación Lineal pertenece a una de estas categorías:

- Infactible (el conjunto de soluciones factibles es vacío)
- Con única solución óptima
- Con infinitas soluciones óptimas
- No acotado

En la siguiente subsección hablaremos de SIMPLEX, un algoritmo que permite resolver cualquier problema de Programación Lineal factible con al menos una solución óptima.

SIMPLEX

El método simplex fue ideado y desarrollado por Dantzig en 1947 como herramienta para resolver problemas de Programación Lineal vinculados a la organización de recursos de la Fuerza Aérea estadounidense. Desde aquel entonces, el campo de la Programación Lineal ha ido adquiriendo más importancia, no sólo por los impactos positivos de sus aplicaciones prácticas, sino también por el renovado interés que otorgaba a problemas teóricos de distintas ramas de la computación y de la matemática.

Recordar que en el Teorema 3 hemos visto que si (PL) admite solución óptima, entonces el valor óptimo se alcanza en, por lo menos, uno de los vértices de su poliedro factible. El método simplex los recorre de manera inteligente en búsqueda de una solución óptima. El primer paso consiste en estandarizar el problema. Debemos lograr que el problema lineal quede escrito de alguna de las siguientes dos maneras:

$$\begin{array}{lrl} \text{mín} & c^T x & \\ s.a: & Ax = & b \\ & x \geq & 0 \end{array} \qquad\qquad \begin{array}{lrl} \text{máx} & c^T x & \\ s.a: & Ax = & b \\ & x \geq & 0 \end{array}$$

con $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ y con $b \in \mathbb{R}^m$ un vector tal que $b_i \geq 0 \; \forall i = 1, \ldots, m$.Como el problema tratado en el paper es un problema de minimización, trabajaremos con la primera de las estandarizaciones. Para estandarizar un problema, se pueden seguir los siguientes lineamientos:

- Si debemos cambiar el objetivo, basta con cambiarle el signo a la función objetivo:

$$\text{máx } c^T x \; \rightarrow \text{mín } -c^T x$$

- Si algún b_i es negativo, multiplicamos ambos lados de la igualdad/desigualdad por -1:

$$\text{si } b_i < 0: \quad \sum_j a_{ij} x_j \leq b_i \rightarrow -\sum_j a_{ij} x_j \geq -b_i$$

$$\text{si } b_i < 0: \quad \sum_j a_{ij} x_j \geq b_i \rightarrow -\sum_j a_{ij} x_j \leq -b_i$$

- Agregamos variables *slack* w_i para transformar las desigualdades en igualdades:

$$\sum_j a_{ij} x_j \leq b_i \; \rightarrow \sum_j a_{ij} x_j + w_i = b_i$$

$$\sum_j a_{ij} x_j \geq b_i \; \rightarrow \sum_j a_{ij} x_j - w_i = b_i$$

  Se agregan también las restricciones:

$$w_i \geq 0 \; \forall i$$

- Si una variable x_j es libre (es decir, $x_j \in (-\infty, +\infty)$) agregamos dos variables no negativas x_j^+ y x_j^- y reemplazamos cada ocurrencia de x_j por $x_j^+ - x_j^-$
- Si una variable x_j es negativa, introducimos la variable no negativa $\tilde{x}_j$ al modelo, y reemplazamos cada ocurrencia de x_j por $-\tilde{x}_j$

Veamos un ejemplo, a la izquierda el problema original y a la derecha el problema estandarizado:

$$\begin{array}{rrcl} \text{máx} & x_1 - x_2 + x_3 & & \\ \text{s.a:} & x_1 + 2x_2 - x_3 \leq & 3 \\ & x_1 - x_2 - x_3 \leq & -2 \\ & x_1 - x_2 \geq & 10 \\ & x_1 \geq & 0 \\ & x_2 \leq & 0 \end{array} \qquad \begin{array}{rrcl} \text{mín} & -x_1 - \tilde{x}_2 - x_3^+ + x_3^- & & \\ \text{s.a:} & x_1 - 2\tilde{x}_2 - x_3^+ + x_3^- + w_1 & = & 3 \\ & -x_1 - \tilde{x}_2 + x_3^+ - x_3^- - w_2 & = & 2 \\ & x_1 + \tilde{x}_2 - w_3 & = & 10 \\ & x_1, \tilde{x}_2, x_3^+, x_3^-, w_1, w_2, w_3 & \geq & 0 \end{array}$$

Ambos problemas de optimización son equivalentes. A partir de la solución óptima del problema estandarizado se puede recuperar la solución óptima del problema original teniendo en cuenta que $x_3 = x_3^+ - x_3^-$ y que $x_2 = -\tilde{x}_2$.

En lo que resta de esta sección trabajaremos entonces con el siguiente problema:

$$\begin{array}{rrcl} \text{mín} & c^T x & & \\ s.a: & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{PLS}$$

Por simplicidad de notación incluímos a las variables slack entre las variables x.

Utilizaremos la siguiente notación:

- $A_{.j} \rightarrow$ columna j de la matriz
- $A_{i.} \rightarrow$ fila i de la matriz
- $a_{ij} \rightarrow$ coeficiente de la fila i columna j de A.

Se asume que $rg(A) = m$ (de lo contrario, se pueden eliminar las filas redundantes), por lo tanto, existen m columnas linealmente independientes de A. Llamemos B a la matriz formada por ellas. Notar que, como las columnas de B son linealmente independientes, B es inversible. Denominaremos matriz residual a la matriz formada por las columnas de A que no están en B, y la notaremos R.

Definición 10. (Matriz básica) Sea B una matriz formada por m columnas linealmente independientes de A, diremos que es una matriz básica.

Definición 11. (Matriz Primal factible) Diremos que una matriz básica B es matriz primal factible si $B^{-1}b \geq 0$

Definición 12. (Solución básica factible) Dada B una matriz primal factible y R la matriz residual, la solución básica factible asociada a B es $x \in \mathbb{R}^n$ tal que $x_B = B^{-1}b$ y $x_R = 0$, donde x_B y x_R son los vectores con las componentes de x asociadas a B y a R respectivamente. Las componentes de x_B son denominadas variables básicas y las de x_R variables no básicas.

Para fijar ideas, veamos un ejemplo. Tenemos el siguiente problema estandarizado:

$$\begin{array}{rrcl} \text{mín} & -x_1 - x_2 - x_3 + x_4 & & \\ \text{s.a:} & x_1 - 2x_2 - x_3 + x_4 + x_5 & = & 3 \\ & -x_1 - x_2 + x_3 - x_4 - x_6 & = & 2 \\ & x_1 + x_2 - x_7 & = & 10 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 & \geq & 0 \end{array} \tag{PLS1}$$

En este caso, se tiene que:

$$A = \begin{pmatrix} 1 & -2 & -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & -1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 3 \\ 2 \\ 10 \end{pmatrix}$$

La elección más sencilla de una matriz básica es tomar las últimas tres columnas de A, en ese caso:

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad R = \begin{pmatrix} 1 & -2 & -1 & 1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Sin embargo, B no es una matriz primal factible, pues:

$$B^{-1}b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 10 \end{pmatrix} = \begin{pmatrix} 3 \\ -2 \\ -10 \end{pmatrix} \ngeq 0$$

Por otro lado, si tomamos $B = [A_{\cdot 1}|A_{\cdot 3}|A_{\cdot 5}]$, B resulta matriz primal factible:

$$B = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} -2 & 1 & 0 & 0 \\ -1 & -1 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}$$

$$B^{-1}b = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \\ 10 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \\ 5 \end{pmatrix} \geq 0$$

Por lo tanto, x_1, x_3, x_5 son variables básicas mientras que x_2, x_4, x_6, x_7 son variables no básicas.Además, tenemos la siguiente solución básica factible: $(10, 0, 12, 0, 5, 0, 0)$. Enunciamos el siguiente teorema que relaciona a las soluciones básicas factibles con los vértices del poliedro factible:

Teorema 4. Un punto $\bar{x}$ es solución básica factible $\iff$ $\bar{x}$ es punto extremo del poliedro factible.

Por lo tanto, si el problema es acotado, sabemos que alguna de las soluciones básicas factibles es una solución óptima. Sin embargo, enumerarlas y evaluarlas a todas resulta poco práctico, dado que se tienen $\binom{n}{m}$ potenciales soluciones básicas factibles. El algoritmo SIMPLEX resuelve este problema eficazmente analizando los vértices del poliedro factible de manera sistemática: partiendo de una solución básica factible, si esta no resulta óptima, se pasa a la solución básica factible adyacente[1] con mejor valor en la función objetivo. Se repite el procedimiento hasta encontrar un óptimo o hasta que se detecte que el problema es no acotado. Para pasar de una solución básica factible a una adyacente, una de las variables básicas pasa a ser no básica (sale de la base) y una de las variables no básicas pasar a ser básica (entra a la base).

Para entender cómo se determina si una solución básica factible es óptima y, en caso negativo, cómo se elige cuál variable sale de la base y cuál entra, presentamos la siguiente definición:

Definición 13. (Costo reducido) Sea x una solución básica factible, B la matriz primal factible asociada, el costo reducido de la variable x_j se define como:

$$\bar{c}_j^T = c_j^T - c_B^T B^{-1} A_{\cdot j}$$

donde c_B son los coeficientes de x_B en la función objetivo.

Observación 1. Los costos reducidos de las variables básicas son nulos,

Observación 2. El vector de costos reducidos de las variables no básicas se pueden calcular como:

$$\bar{c}_R^T = c_R^T - c_B^T B^{-1} R$$

[1]Dos soluciones básicas factibles son adyacentes si el segmento que las une corresponde a una arista del poliedro factible

Veamos por qué el vector de costos reducidos aporta la información que necesitamos. Supongamos, sin pérdida de generalidad, que B está compuesta por las primeras m columnas (de lo contrario, cambiamos el orden de las columnas en A y el de sus correspondientes componentes en x). De esta manera, $A = [B|R]$ y se sigue:

$$\begin{aligned} Ax = b \iff [B|R][x_B|x_R]^T = b \iff Bx_B + Rx_R = b \iff B^{-1}Bx_B + B^{-1}Rx_R = B^{-1}b \iff \\ \iff x_B + B^{-1}Rx_R = B^{-1}b \Rightarrow x_B = B^{-1}b - B^{-1}Rx_R \end{aligned}$$

Examinando la función objetivo:

$$c^T x = c_B^T x_B + c_R^T x_R = c_B^T(B^{-1}b - B^{-1}Rx_R) + c_R^T x_R = c_B^T B^{-1}b + \underbrace{(c_R^T - c_B^T B^{-1}R)}_{\bar{c}_R^T} Rx_R$$

Por lo tanto, el vector de costos reducidos nos indica cómo variaría la función objetivo si se aumentara el valor de alguna de las variables no básicas (que actualmente son nulas). Por lo tanto, si alguna componente de $\bar{c}_R^T$ es negativa, significa que si aumenta la correspondiente variable no básica, el valor de la función objetivo será menor. De lo contrario, si todas sus componentes son no negativas, el valor de la función objetivo no puede mejorar y se ha alcanzado el óptimo. Los criterios se resumen a continuación:

Criterio de optimalidad. Sea x^* solución básica factible de (PLS), si $\bar{c}_R^T \geq 0$, x^* es una solución óptima.

Criterio de entrada a la base. Si $\bar{c}_R^T \ngeq 0$, entra a la base la variable no básica con menor costo reducido.

Criterio de salida de la base. Sea x_s la variable que entra a la base, para cada x_i de x_B, por la i-ésima restricción del problema, se tiene que $x_i + (B^{-1}A_{\cdot s})_i x_s = (B^{-1}b)_i$. Por lo tanto, para asegurar que $x_i \geq 0$, x_s puede crecer hasta $\frac{(B^{-1}b)_i}{(B^{-1}A_{\cdot s})_i}$. Teniendo esto en cuenta, sea i_0 el índice donde se realiza $\text{mín}_{i:\,(B^{-1}R)_{is}>0} \left\{ \frac{(B^{-1}b)_i}{B^{-1}A_{\cdot s})_i} \right\}$, la variable que sale de la base es x_{i_0}, es decir, la que le impone la cota más restrictiva al crecimiento de x_s.

Teorema 5. Sea B una base primal factible, si en el punto extremo $\tilde{x}$ todas las variables básicas son positivas, entonces el cambio de base efectuado por SIMPLEX proporciona un punto extremo adyacente a $\tilde{x}$.

Para ver concretamente cómo itera SIMPLEX, veamos el siguiente ejemplo:

$$\begin{aligned} \text{mín} \quad & z = -x_1 - 2x_2 \\ \text{s.a:} \quad & 2x_1 + x_2 + x_3 = 2 \\ & -x_1 + 2x_2 + x_4 = 7 \\ & x_1 + x_5 = 3 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Expresando este problema de manera matricial:

$$\begin{aligned} \text{mín} \quad & c^T x \\ \text{s.a:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Siendo:

$$A = \begin{pmatrix} 2 & 1 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} \quad c = \begin{pmatrix} -1 \\ -2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

Tomaremos como matriz primal factible a B formada por $A{\cdot}1, A{\cdot}2, A{\cdot}5$ dada por:

$$B = \begin{pmatrix} -2 & 1 & 0 \\ -1 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad B^{-1}b = \begin{pmatrix} -\frac{2}{3} & \frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{3} & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix} \geq 0$$

Así, tenemos que:

$$x_B^T = (x_1, x_2, x_5) \quad x_R^T = (x_3, x_4)$$

$$B = \begin{pmatrix} -2 & 1 & 0 \\ -1 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad B^{-1} = \begin{pmatrix} -\frac{2}{3} & \frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{3} & 1 \end{pmatrix} \quad R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$c_B^T = (-1, -2, 0) \quad c_R^T = (0, 0)$$

$$x_B = B^{-1}b = \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix}$$

Calculamos los costos reducidos de las variables no básicas para chequear el criterio de optimalidad:

$$\bar{c}_R^T = c_R^T - c_B^T B^{-1} R = (0,0) - (-1,-2,0) \begin{pmatrix} -\frac{2}{3} & \frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{3} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} = \left(-\frac{4}{3}, \frac{5}{3} \right)$$

Luego, esto indica que la solución básica factible actual no es óptima y que x_3 debe entrar a la base pues es la variable no básica con menor costo reducido positivo. Veamos quién debe salir de la base:

$$B^{-1}A_{\cdot 3} = \begin{pmatrix} -\frac{2}{3} & \frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{3} & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{2}{3} \\ -\frac{1}{3} \\ \frac{2}{3} \end{pmatrix}$$

Ahora calculamos $\dfrac{(B^{-1}b)_i}{(B^{-1}A_{\cdot 3})_i}$ para cada i tal que $(B^{-1}A_{\cdot s})_i > 0$. En este caso, sólo $i = 3$ cumple esto, y la componente de x_B que le corresponde es x_5:

$$\frac{(B^{-1}b)_3}{(B^{-1}A_{\cdot 3})_3} = \frac{2}{\frac{2}{3}} = 3$$

Luego, sale x_5 de la base. Entonces, se remueve $A_{\cdot 5}$ de B y se le agrega $A_{\cdot 3}$:

$$x_B^T = (x_1, x_2, x_3) \quad x_R^T = (x_4, x_5)$$

$$B = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad B^{-1} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 1 & -\frac{1}{2} & \frac{3}{2} \end{pmatrix} \quad R = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$c_B^T = (-1, -1, 0) \quad c_R^T = (0, 0)$$

$$x_B = B^{-1}b = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 1 & -\frac{1}{2} & \frac{3}{2} \end{pmatrix} \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 3 \end{pmatrix}$$

Calculamos los costos reducidos de las variables no básicas y $\bar{b}$:

$$\bar{c}_R^T = c_R^T - c_B^T B^{-1} R = (0,0) - (-2,-1,0) \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 1 & -\frac{1}{2} & \frac{3}{2} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = (1, 2)$$

Como los costos reducidos son todos no negativos, entonces llegamos a una solución óptima:

$$(5, 3, 3, 0, 0)$$

Hay mucho más para contar sobre este método: ¿cómo se elige la base primal factible inicial? ¿SIMPLEX puede ciclar? ¿cómo se detecta la presencia de infinitos óptimos?, entre muchas otras cosas. No obstante, lo que hemos comentado hasta aquí es suficiente para comprender los contenidos del paper. Como observación final, una característica interesante de SIMPLEX es que se trata de un algoritmo de complejidad exponencial; a pesar de ello, resulta muy eficiente en las aplicaciones prácticas.

Problema Dual

A partir de cualquier problema de Programación Lineal se puede definir su problema dual. Como veremos más adelante, el problema dual aporta información importante sobre el problema original (denominado problema primal). Veamos a continuación un problema primal de minimización y su correspondiente problema dual:

$$\begin{array}{lrcl} \text{mín} & c^T x & & \\ \text{s.a:} & Ax & \geq & b \\ & x & \geq & 0 \end{array} \qquad \text{(A)} \qquad\qquad \begin{array}{lrcl} \text{máx} & b^T \pi & & \\ \text{s.a:} & A^T \pi & \leq & c \\ & \pi & \geq & 0 \end{array} \qquad \text{(B)}$$

En general, a partir de un problema lineal, se puede plantear su problema dual siguiendo estos lineamientos:

Primal	Dual
Obj: Minimizar	Obj: Maximizar
i-ésima restricción $\leq$	i-ésima variable ≤ 0
i-ésima restricción $\geq$	i-ésima variable ≥ 0
i-ésima restricción $=$	i-ésima variable libre
j-ésima variable ≥ 0	j-ésima restricción $\leq$
j-ésima variable ≤ 0	j-ésima restricción $\geq$
j-ésima variable libre	j-ésima restricción $=$

Primal	Dual
Obj: Maximizar	Obj: Minimizar
i-ésima restricción $\leq$	i-ésima variable ≥ 0
i-ésima restricción $\geq$	i-ésima variable ≤ 0
i-ésima restricción $=$	i-ésima variable libre
j-ésima variable ≥ 0	j-ésima restricción $\geq$
j-ésima variable ≤ 0	j-ésima restricción $\leq$
j-ésima variable libre	j-ésima restricción $=$

Ahora enunciaremos algunos de los teoremas que relacionan a las soluciones (A) con las de (B):

Teorema 6. (Teorema Fundamental de Dualidad) Si el problema (A) tiene solución óptima x^* entonces (B) tiene solución óptima π^* tal que:

$$c^T x^* = b^T \pi^*$$

Teorema 7. (Teorema de Holgura Complementaria) Sean $x^* = (x_1^*, \cdots, x_n^*)$ solución óptima del primal e $y^* = (y_1^*, \cdots, y_m^*)$ solución óptima del dual, las siguientes son condiciones necesarias y suficientes para la optimalidad simultánea de x^* e y^*:

$$\sum_{i=1}^{m} a_{ij} y_i^* = c_j \quad \text{o} \quad x_j^* = 0 \quad \text{(o ambos)} \quad \forall j = 1, \cdots, n$$

$$\sum_{j=1}^{m} a_{ij} x_j^* = b_i \quad \text{o} \quad y_i^* = 0 \quad \text{(o ambos)} \quad \forall i = 1, \cdots, m$$

Teorema 8. Si el problema (A) tiene solución óptima x^*, al finalizar SIMPLEX el costo reducido de la variable x_j se puede expresar como:

$$\bar{c}_j = c_j - \sum_{i=1}^{m} \pi_i^* a_{ij}$$

donde π^* es la solución óptima del problema dual.

Generación de columnas

El método de generación de columnas (CG) es empleado para resolver problemas de Programación Lineal con tal cantidad de variables que plantearlo explícitamente resulta imposible (por ejemplo, porque existe una cantidad exponencial de ellas). Como hemos visto en el apartado de SIMPLEX, la solución óptima de un problema con n variables y m restricciones ($n > m$), tiene m variables básicas. En el caso en el que $n \gg m$, es muy probable que muchas de las variables nunca entren a la base, por lo tanto, ¿qué utilidad tiene considerarlas en la formulación del problema? Tal vez sería más eficiente comenzar con un conjunto más reducido de variables e ir generando nuevas que mejoren el valor de la función objetivo. Esta es la idea detrás de CG.

Supongamos que tenemos el siguiente problema de Programación Lineal, al que nos referiremos como Problema Maestro:

$$\begin{aligned} z_{PM}^* = \quad & \text{mín} \quad \sum_{j \in J} c_j x_j \\ & s.a: \quad \sum_{j \in J} a_{ij} x_j \quad \geq \quad b_i \quad \forall\, i \in I \\ & \qquad\qquad x_j \quad \geq \quad 0 \quad \forall\, j \in J \end{aligned} \tag{PM}$$

tal que $|J|$ es demasiado grande como para escribir todas las x_j de manera explícita. CG comienza resolviendo el problema de Programación Lineal con un subconjunto $J' \subset J$ tal que $|J'| \ll |J|$, al que llamaremos Problema Mestro Restringido:

$$\begin{aligned} z_{PMR}^* = \quad & \text{mín} \quad \sum_{j \in J'} c_j x_j \\ & s.a: \quad \sum_{j \in J'} a_{ij} x_j \quad \geq \quad b_i \quad \forall\, i \in I \\ & \qquad\qquad x_j \quad \geq \quad 0 \quad \forall\, j \in J' \end{aligned} \tag{PMR}$$

A continuación, se resuelve (PMR) para obtener su solución óptima x^*. Por el Teorema 7, se obtiene π^*, el óptimo del problema dual de (PMR). Para saber si x^* es solución óptima de (PM), tendríamos que chequear si el costo reducido de cada x_j, $j \in J$ es no negativo. En caso contrario, se toma una variable x_j tal que $\bar{c}_j < 0$ y se la agrega a (PMR). Sin embargo, calcular el costo reducido de todas las variables de (PM) resulta computacionalmente imposible (o al menos muy poco eficiente) debido al tamaño de $|J|$. En su lugar, se resuelve el siguiente problema de optimización auxiliar:

$$z^* = \text{mín}\{\bar{c}_j | j \in J\}$$

es decir, buscamos $j \in J$ con el menor costo reducido. Pueden ocurrir dos situaciones:

- si $z^* < 0$, añadimos $\arg\min_{j \in J} \bar{c}_j$ a J' y se vuelve a resolver (PMR) con el nuevo conjunto J'.
- si $z^* \geq 0$, significa que no hay $j \in J$ tal que $\bar{c}_j < 0$, por lo que x^* cumple con el criterio de optimalidad de (PM).

Recordar que, por Teorema 8, tenemos que $\bar{c}_j = c_j - \sum_{i \in I} \pi_i^* a_{ij}$, por lo tanto, el Problema Auxiliar puede plantearse como:

$$\begin{aligned} z^* = \quad & \text{mín} \quad c_j - \sum_{i \in I} \pi_i^* a_{ij} \\ & s.a: \qquad j \in J \end{aligned} \tag{PA}$$

Las restricciones $j \in J$ dependen de cada problema. En Algoritmo 1 se muestra el procedimiento del algoritmo de Generación de Columnas.

Algoritmo 1: GENERACIÓN DE COLUMNAS

Entrada: Problema Maestro Restringido con un conjunto inicial de variables J'
Salida: x^* óptimo del Problema Maestro

```
mientras z* ≥ 0 hacer
    Resolver (PMR) y obtener x*, π*
    Resolver (PA) y obtener z*
    si z* < 0 entonces
        J' ← J' ∪ {j*} con j* tal que c̄_{j*} = z*
devolver x*
```

Programación Lineal Entera y Branch & Bound

Hasta ahora hemos trabajado con problemas lineales donde las variables pueden tomar valores reales. Cuando a un problema de Programación Lineal se añade la restricción $x \in \mathbb{Z}^n$, es decir que sus variables sólo pueden tomar valores enteros, se transforma en un problema de Programación Lineal Entera:

$$\begin{array}{lrcl} \text{mín} & c^T x & & \\ \text{s.a:} & Ax & \geq & b \\ & x & \geq & 0 \\ & x & \in & \mathbb{Z}^n \end{array} \tag{PLE}$$

En particular, si se requiere que $x \in \{0,1\}^n$, se trata de un problema de Programación Lineal Entera Binaria (PLEB). Tantos los problemas (PLE) como (PLBE) han sido muy estudiados pues muchas aplicaciones prácticas se modelan de esa manera: programación de horarios, armados de fixtures deportivos, problemas de ruteo de vehículos, problemas de asignación, planificación de producción, etc. A diferencia de los problemas de Programación Lineal, para los cuales existen algoritmos con complejidad temporal polinomial, resolver un problema PLE es NP-Hard. Por esta razón, las técnicas para resolver problemas PLE generalmente involucran lo que se conoce como la relajación lineal.

Definición 14. (Relajación lineal) Dado un problema (PLE), se denomina relajación lineal (RL) al problema de Programación Lineal que se obtiene eliminando la restricción $x \in \mathbb{Z}^n$.

Notar que el valor óptimo de la relajación es una cota inferior al óptimo de PLE, pues la región de soluciones factibles de PLE está incluida en la de su relajación lineal. Por otro lado, si el óptimo de la relajación lineal tiene todas sus componentes enteras, entonces también es el óptimo de PLE.

Una idea intuitiva para encarar un problema PLE podría ser resolver su relajación lineal y luego redondear el resultado para que las variables tomen valores enteros. Sin embargo, los ejemplos de la Figura 3 muestran que este procedimiento, en general, puede conducir a resultados incorrectos.

Uno de los algoritmos utilizados en la resolución de problemas PLE es Branch & Bound. Como sólo es mencionado en el paper, presentaremos su idea general. Comenzando con el problema PLE original, que notaremos $PLE^{(0)}$, se resuelve su relajación lineal $RL^{(0)}$ obteniendo su óptimo $x^*_{RL^{(0)}}$. Si $x^*_{RL^{(0)}} \notin \mathbb{Z}^n$,

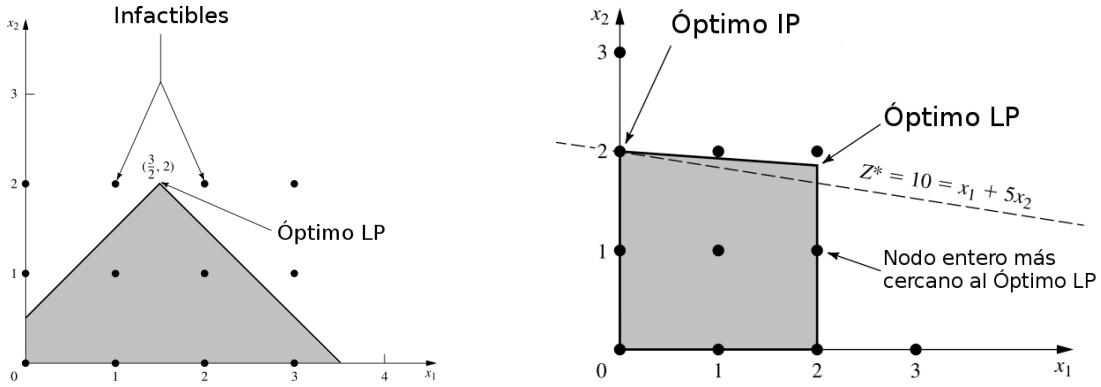

Figura 3: Ejemplos en los que redondear el resultado de la relajación lineal no otorga la solución óptima del problema

se lleva a cabo el proceso de ramificación (*Branch*). Se elije alguna variable x_{j_0} tal que $(x^*_{RL^{(0)}})_{j_0} \notin \mathbb{Z}$ y se consideran dos problemas: $PLE^{(1)}$ que añade a PLE la restricción $x_{j_0} \geq \lceil (x^*_{RL^{(0)}})_{j_0} \rceil$ y $PLE^{(2)}$ que añade a PLE la restricción $x_{j_0} \leq \lfloor (x^*_{RL^{(0)}})_{j_0} \rfloor$. El procedimiento se repite para $PLE^{(1)}$ y $PLE^{(2)}$, y así sucesivamente.

Cuando para algún $k \in \mathbb{N}$, $RL^{(k)}$ tiene solución óptima entera, se compara el valor óptimo con el mejor valor óptimo obtenido proporcionado por una solución entera hasta el momento (valor incumbente, notado $\underline{z}$). Si es menor el nuevo valor óptimo, se actualiza el valor incumbente.

No se ramifica $PLE^{(k)}$ si ocurre alguna de las siguientes situaciones:

- $RL^{(k)}$ es infactible.
- $RL^{(k)}$ tiene solución óptima entera.
- el valor óptimo de $RL^{(k)}$ es mayor o igual que $\underline{z}$ (*Bound*)[2]

El algoritmo termina cuando no se pueden realizar más ramificaciones.

2. Trabajo desarrollado en el *paper*

Descripción del problema

El problema de ruteo con flota heterogénea (abreviado HVRP por sus siglas en inglés) consiste en diseñar un conjunto de rutas para una flota heterogénea de vehículos que debe satisfacer la demanda de un conjunto de clientes. Cuando hablamos de flota heterogénea, nos referimos a que se cuenta con distintos tipos de vehículos en cuanto a capacidades, costos fijos y costos variables. En el HVRP se considera que hay una cantidad ilimitada de vehículos de cada tipo. Por otro lado, se requiere que cada cliente sea visitado exactamente una vez, que la ruta de cada vehículo no exceda su capacidad y que comience y termine en el depósito de distribución. El costo de la ruta de cada vehículo viene dado por su costo fijo (es decir, el costo en el que se incurre por utilizarlo) más el costo variable relacionado a la longitud de su viaje. El objetivo es minimizar el total de los costos de las rutas.

Veamos un ejemplo sencillo para fijar ideas. En el Cuadro 1 tenemos los tipos de vehículos con sus respectivas características. Por otro lado, en la Figura 4 se visualizan los clientes que deben visitarse.

[2]Dado que las regiones de soluciones factibles de los subproblemas que aparezcan en la rama están incuidas en la región de soluciones factibles de $RL^{(k)}$, sus valores óptimos nunca serán menores que $\underline{z}$

Tipo de vehículo	Costo fijo	Costo por km.	Capacidad en kg.
1	500	40	1250
2	260	19	600
3	80	11	120

Cuadro 1: Tipos de vehículos y sus características

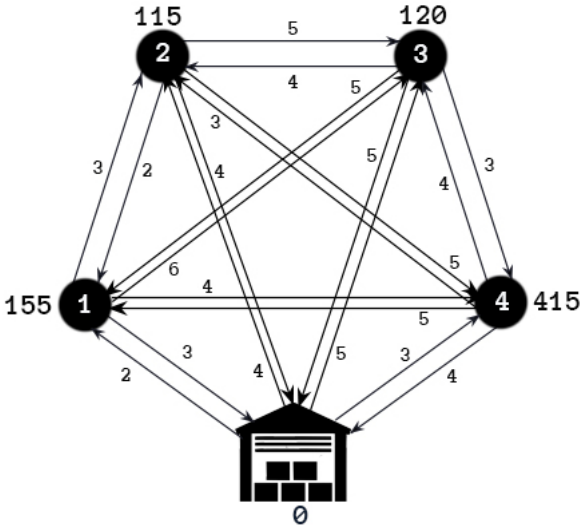

Figura 4: El nodo 0 corresponde al depósito. El peso de cada arista representa la distancia en km. entre los nodos y el peso de cada nodo representa la demanda en kg. del correspondiente cliente.

Una solución factible es contratar un vehículo de tipo 1 y realizar el siguiente recorrido:

$$0 \to 4 \to 1 \to 3 \to 2 \to 0$$

La longitud de esa ruta es de 22 km. y su costo es de $500 + 40 \times 22 = 1380$. Otra solución podría ser contratar un vehículo v_1 del tipo 2 y dos del tipo 3 (v_2 y v_3) y utilizar las siguientes rutas:

$$\begin{array}{lll} v_1: & 0 \to 4 \to 1 \to 0 & \Rightarrow \text{costo: } 469 \\ v_2: & 0 \to 3 \to 0 & \Rightarrow \text{costo: } 135 \\ v_3: & 0 \to 2 \to 0 & \Rightarrow \text{costo: } 168 \end{array}$$

En este caso el costo total es de 772, por lo cual resulta una mejor solución que la sugerida inicalmente. Vale la pena reiterar que en el HVRP se deben llevar a cabo dos decisiones: cuántos vehículos utilizar de cada tipo y qué ruta seguirá cada uno de ellos. Dado que el HVRP resulta ser un problema NP-Hard, los estudios se han enfocado en desarrollar heurísticas en vez de algoritmos que otorguen soluciones exactas. Por ejemplo, se puede formular al HVRP como un problema de Programación Lineal Entera y resolverlo utilizando directamente Branch & Bound. Sin embargo, para instancias con 20 clientes puede tomar horas resolverlo a optimalidad. Esto resulta inconveniente para aplicaciones prácticas, pues una instancia con 20 clientes es considerada pequeña.

Heurística para resolver HVRP

Choi y Tcha proponen una heurística basada en el método de generación de columnas (CG). La idea de utilizar CG es motivada por el éxito del método en otros problemas de ruteo, especialmente en problemas

de ruteo de vehículos con ventanas temporales. HVRP es modelado con un digrafo completo $G = (N, A)$ similar al de la Figura 4: el nodo 0 representa al depósito y luego hay un nodo para cada uno de los clientes, quienes son numerados desde 1 hasta n. Por otro lado A es el conjunto de arcos dirigidos del grafo, cada uno con su correspondiente distancia ℓ_{ij} y se supone que sus distancias satisfacen la desigualdad triangular. En el Cuadro 2 resumimos la notación que se utiliza a lo largo del trabajo.

Notación	Parámetro del problema
q_i	demanda del cliente i, $i \in N - \{0\}$
$T = \{1, \ldots, m\}$	conjunto de tipos de vehículos
b_k	capacidad del tipo de vehículo $k \in T$
f_k	costo fijo del tipo de vehículo $k \in T$
g_k	costo variable por unidad de distancia del tipo de vehículo $k \in T$
c_{ij}^k	costo de recorrer el arco (i, j) con un vehículo de tipo k: $c_{ij}^k = g_k \sum_{(i,j)\in r} \ell_{ij}$
c_r^k	costo de recorrer la ruta r con un vehículo de tipo k: $c_r^k = f_k + \sum_{(i,j)\in r} c_{ij}^k$

Cuadro 2: Notación

Diremos que un par ruta-vehículo (r, k) es factible si la ruta comienza y termina en el depósito y la suma de la demanda de los nodos que visita no excede b_k. Se notará a R como el conjunto de pares ruta-vehículo factibles y como R_k al conjunto de pares ruta-vehículo para el tipo de vehículo k. Notar que $R = \cup_{k\in T} R_k$. Finalmente se define el parámetro $\delta_{ir}^k \in \{0, 1\}$ que vale 1 si y sólo si un vehículo de tipo k visita al cliente i en la ruta r.

Una vez presentada la notación, estamos en condiciones de plantear el Problema Maestro.

x_r^k: vale 1 si y sólo si el par de ruta-vehículo (r, k) es parte de la solución, $x_r^k \in \{0, 1\}$
y_k: cantidad de vehículos de tipo k utilizados, $y_k \in \mathbb{Z}_{\geq 0}$

$$
\begin{aligned}
\text{mín} \quad & \sum_{k\in T} \sum_{r\in R_k} c_r^k x_r^k & & & \\
s.a: \quad & \sum_{k\in T} \sum_{r\in R_k} \delta_{ir}^k x_r^k & \geq & 1 & \forall\, i \in N - \{0\} \\
& \sum_{r\in R_k} x_r^k & = & y_k & \forall\, k \in T \\
& x_r^k & \in & \{0, 1\} & \forall\, k \in T \,\forall\, r \in R_k \\
& y_k & \in & \mathbb{Z}_{\geq 0} & \forall\, k \in T
\end{aligned}
\tag{MP}
$$

La primera restricción asegura que cada cliente es visitado al menos una vez. La segunda y la cuarta aseguran que la cantidad de vehículos utilizados será una magnitud entera, lo cual acelera el proceso de Branch & Bound. En el peor de los casos, para cada $k \in T$ podríamos tener alrededor de 2^n posibles rutas, por lo que la cantidad de variables x es enorme. Por esta razón, consideraremos el Problema Maestro Reducido (abreviado RMP) en el que se trabaja inicialmente con un conjunto $R' \subset R$ tal que $|R'| \ll |R|$. Siguiendo el esquema de CG, a continuación consideramos la relajación lineal de RMP:

$$
\begin{array}{lrcll}
\text{mín} & \displaystyle\sum_{k\in T}\sum_{r\in R'_k} c_r^k x_r^k & & & \\
s.a: & \displaystyle\sum_{k\in T}\sum_{r\in R'_k} \delta_{ir}^k x_r^k & \geq & 1 & \forall\, i \in N-\{0\} \\
& \displaystyle\sum_{r\in R'_k} x_r^k - y_k & = & 0 & \forall\, k \in T \\
& x_r^k & \leq & 1 & \forall\, k \in T \,\forall\, r \in R'_k \\
& x_r^k & \geq & 0 & \forall\, k \in T \,\forall\, r \in R'_k \\
& y_k & \geq & 0 & \forall\, k \in T
\end{array}
\qquad \text{(RMLP)}
$$

Para plantear el dual se ignoran las restricciones del tercer conjunto, es decir, las del tipo $x_r^k \leq 1$. Esto suele hacerse porque cuando las variables están acotadas superiormente, la interpretación del costo reducido se vuelve más compleja. Notar que cuando hablamos de SIMPLEX y de costo reducido en la primera sección del informe, sólo considerábamos que las variables eran acotadas inferiormente. Además, es importante notar que en la solución óptima x^* de RMLP no puede ocurrir que $x_r^{k*} > 1$ para algún $(\bar{r}, k) \in R'_k$. De lo contrario, bastaría reemplazar su valor por 1 y se tendría una solución $\bar{x}$ factible[3] con menor valor en la función objetivo. Por ende, a fines prácticos, podemos eliminar las restricciones $x_r^k \leq 1$. Así, el problema dual de (RMLP) quedaría de la siguiente manera:

$$
\begin{array}{lrcll}
\text{máx} & \displaystyle\sum_{i\in N-\{0\}} \pi_i & & & \\
s.a: & \displaystyle\sum_{i\in N-\{0\}} \delta_{ir}^k \pi_i + \lambda_k & \leq & c_r^k & \forall\, k \in T\, \forall r \in R'_k \\
& \pi_i & \geq & 0 & \forall\, i \in N-\{0\} \\
& \lambda_k & \leq & 0 & \forall\, k \in T
\end{array}
$$

donde utilizamos π para referirnos a las variables del dual correspondientes al primer conjunto de restricciones de (RMLP) y λ para las variables del dual correspondientes al segundo.

Luego de resolver (RMLP) utilizando SIMPLEX, se obtienen los óptimos del dual π^* y λ^* y se puede plantear el problema auxiliar para encontrar, si los hubiera, pares ruta-vehículo con costos reducidos negativos:

$$
\begin{array}{ll}
\text{mín} & c_r^k - \displaystyle\sum_{i\in N-\{0\}} \pi_i^* \delta_{ir}^k - \lambda_k^* \\
s.a: & (r,k) \in R
\end{array}
\qquad \text{(SP)}
$$

Es importante notar que la solución óptima de SP está contenida en al menos un R_k, por lo que resolver SP es equivalente a resolver los problemas mutualmente excluyentes SP(1),...,SP(m), donde SP(k) es la versión restringida de SP con R_k en vez de R. Ahora bien, supongamos que fijamos $k \in T$ y que consideramos una ruta $r \in R_k$, $r = (i_0, i_1, \ldots, i_H, i_{H+1})$ con $i_0 = i_{H+1} = 0$. Analicemos la función objetivo de SP(k):

$$
\bar{c}_r^k = \underbrace{c_r^k}_{(1)} - \underbrace{\sum_{i\in N-\{0\}} \pi_i^* \delta_{ir}^k}_{(2)} - \underbrace{\lambda_k^*}_{(3)}
$$

(1) por Cuadro 2 tenemos que $c_r^k = f_k + \displaystyle\sum_{h=1}^{H+1} c_{i_{h-1} i_h}^k$

[3] $\bar{x}$ es factible, pues para cada $i \in \bar{r}$, $(i \neq 0)$, vale que $\delta_{i\bar{r}}^k = 1$ y $1 \leq \bar{x}_{\bar{r}}^k \leq \displaystyle\sum_{k\in T}\sum_{r\in R'_k} \delta_{ir}^k x_r^k$

(2) recordar que $\delta_{ir}^k = 1 \iff i \in \{i_0, i_1, \ldots, i_H, i_{H+1}\}$, por lo tanto, tenemos que

$$\sum_{i \in N - \{0\}} \pi_i^* \delta_{ir}^k = \sum_{h=1}^{H} \pi_{i_h}^*$$

(3) notar que, como el primer conjunto de restricciones de RMLP no abarca al nodo 0, $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$. Definimos por conveniencia (y abusando la notación) $\pi_0^* = \pi_{H+1}^* = \lambda_k^*$

De esta manera, se tiene que:

$$\bar{c}_r^k = f_k + \sum_{h=1}^{H+1} c_{i_{h-1} i_h}^k - \sum_{h=1}^{H} \pi_{i_h}^* - \pi_{H+1}^* = \sum_{h=1}^{H+1} (c_{i_{h-1} i_h}^k - \pi_{i_h}^*) + f_k$$

Por lo tanto, el costo reducido de a ruta r no es más que el costo de recorrerla en un grafo $\bar{G}_k = (N, \bar{A}_k)$ para el vehículo de tipo k donde el costo de cada arco (i, j) viene dado por $c_{ij}^k - \pi_j^*$. Así pues, resolver SP(k) consiste en hallar la ruta de costo mínimo en $\bar{G}_k$ que satisfaga que la suma de demandas de sus nodos no exceda b_k. Esta es prácticamente la definición del problema conocido como Problema de Camino Mínimo con Restricciones de Recursos. En nuestro caso, el recurso es la capacidad del vehículo. Este tipo de problema es fuertemente NP-hard. Por lo tanto, probablemente resulte poco conveniente, por ejemplo, tratar de resolver SP(k) modelándolo como un problema de Programación Lineal Entera y utilizando Branch & Bound.

Entonces, los autores proponen encarar el problema desde la Programación Dinámica. Se define $F_k(S, i)$ como el valor del camino de costo mínimo en $\bar{G}_k$ que comienza en el depósito, visita cada nodo de $S \in N - \{0\}$ exactamente una vez y termina en el nodo i. Para cada estado (S, i) podemos obtener una ruta factible añadiendo el arco $(i, 0)$ y su costo será $F_k(S, i) + \bar{c}_{i0}^k + f_k$. Las ecuaciones de recurrencia vienen dadas por:

$$F_k(\emptyset, 0) = 0 \tag{1}$$

$$F_k(S, j) = \min_{(i,j) \in \bar{A}_k} \left\{ F_k(S - \{j\}, i) + \bar{c}_{ij}^k, \text{ tal que } \sum_{i \in S} q_i \leq b_k \right\} \quad \forall S \subseteq N - \{0\},\ \forall j \in S \tag{2}$$

La segunda ecuación indica que para obtener el valor del camino de costo mínimo visitando los nodos de S hasta j, basta encontrar el mínimo costo de los caminos mínimos de sus posibles predecesores. De esta manera, la solución óptima para SP(k) se obtiene calculando:

$$\min_{S \subseteq N - \{0\}} \min_{j \in S} \{F_k(S, j) + \bar{c}_{j0}^k + f_k\}$$

El problema es que la cantidad de estados (S, i) es exponencial (tenemos 2^n elecciones posibles de S) y por lo tanto, no resulta práctico utilizar directamente esas ecuaciones de recurrencia. En estos casos, se suele aplicar lo que se conoce como una relajación del espacio de los estados. Esto quiere decir que se mapean un conjunto de estados (S, i) en un nuevo estado (q, i): para $S \subseteq N - \{0\}$ se define $Q(S)$ como la suma de las demandas de los nodos que lo componen, $Q(S) = q = \sum_{j \in S} q_j$. En particular, $Q(S - \{j\}) = Q(S) - q_j$. Redefinimos entonces $F_k(q, i)$ como el valor del camino de costo mínimo desde el depósito hasta el nodo $i \in N - \{0\}$ que acumula una carga q. Entonces se cambia la segunda ecuación de recurrencia por:

$$F_k(q, j) = \min_{(i,j) \in \bar{A}_k} \{F_k(q - q_j, i) + \bar{c}_{ij}^k\} \quad \forall j \in N - \{0\}\ \forall\ q_j \leq q \leq b_k \tag{3}$$

Ahora bien, notemos que al usar el mapeo ya no se pide que se visite exactamente una vez a cada cliente. Por lo tanto, se resuelve una versión de SP(k) sin dicha restricción, que llamaremos $\overline{\text{SP}}(k)$. Dado que este

problema tiene menos restricciones, el valor óptimo de $\overline{\text{SP}}(k)$ es una cota inferior para el valor óptimo de $\text{SP}(k)$. Además, los autores proponen una mejora para que la cota sea más ajustada: considerar rutas que no tengan 2-ciclos, es decir, ciclos de dos nodos $i \to j \to i$. Esto se puede lograr manteniendo el mejor y el segundo mejor camino para cada estado (q, i). Sea $F_k^1(q, i)$ el valor de camino de costo mínimo desde el depósito hasta el nodo i con carga q. Sea $\theta_k^w(q, i)$ el predecesor de i en el camino con costo $F_k^w(q, i)$ y sea $F_k^2(q, i)$ el valor del segundo mejor camino hasta el nodo i tal que $\theta_k^2(q, i) \neq \theta_k^1(q, i)$. Para k y q fijos, se define $\psi_k^q(i, j)$ como el valor del camino de costo mínimo desde el depósito hasta el nodo j que pasa por i y no tiene 2-ciclos. Entonces, se tiene que:

$$\psi_k^q(i,j) = \begin{cases} F_k^1(q - q_j, i) + \bar{c}_{ij}^k & \text{si } \theta_k^1(q - qj, i) \neq j \\ F_k^2(q - q_j, i) + \bar{c}_{ij}^k & \text{en caso contrario} \end{cases}$$

En el primer caso, como el predecesor de i en el camino mínimo desde el depósito no es j, se puede agregar el arco (i, j) a la ruta para obtener el valor de la ruta mínima hasta j que pasa por i. El segundo caso contempla la situación en la que j es predecesor de i en el camino con costo mínimo hasta i y, por lo tanto, hay que utilizar $F_k^2(q - q_j, i)$ para evitar 2-ciclos (recordar que, por definición, en este caso $\theta_k^2(q, i) \neq j$). Ahora se pueden definir las ecuaciones de recurrencia para calcular $F_k^w(q, i)$ con $w = 1, 2$:

$$F_k^1(0,0) = 0 \tag{4}$$

$$F_k^1(q,j) = \min_{(i,j)\in \bar{A}_k} \{\psi_k^q(i,j)\}, \quad \forall j \in N - \{0\} \; \forall \, q_j \leq q \leq b_k \tag{5}$$

$$F_k^2(q,j) = \min_{(i,j)\in \bar{A}_k} \{\psi_k^q(i,j) \colon i \neq \theta_k^1(q,j)\}, \quad \forall j \in N - \{0\} \; \forall \, q_j \leq q \leq b_k \tag{6}$$

y la solución de $\overline{\text{SP}}(k)$ sin 2-ciclos viene dada por:

$$\min_{j \in N - \{0\}} \min_{q_j \leq q \leq b_k} \{F_k^1(q,j) + \bar{c}_{j0}^k + f_k\}$$

Como mencionamos antes, la relajación del espacio de los estados no prohibía que un cliente fuera visitado más de una vez. Por eso, los autores proponen modificar (MP) para reflejar este cambio. Consideran el siguiente problema:

$$\begin{aligned} \min \quad & \sum_{k \in T} \sum_{r \in P_k} c_r^k x_r^k & & & \\ s.a: \quad & \sum_{k \in T} \sum_{r \in P_k} \gamma_{ir}^k x_r^k & \geq & 1 & \forall \, i \in N - \{0\} \\ & \sum_{r \in P_k} x_r^k & = & y_k & \forall \, k \in T \\ & x_r^k & \in & \{0,1\} & \forall \, k \in T \, \forall \, r \in P_k \\ & y_k & \in & \mathbb{Z}_{\geq 0} & \forall \, k \in T \end{aligned} \tag{CP}$$

donde P es el conjunto de pares ruta-vehículo factibles con rutas que pueden contener ciclos (i.e. pueden visitar a un cliente más de una vez) y P_k el subconjunto de P para el vehículo de tipo k. Dado un par factible (r, k), γ_{ir}^k denota la cantidad de veces que el cliente $i \in N - \{0\}$ es visitado por el vehículo k en la ruta r.

La relajación lineal de (CP), que notaremos CLP, puede ser resuelta con Generación de Columnas, siendo $\overline{\text{SP}}(k)$ el problema auxiliar. Notar que una solución óptima de CP es una solución óptima de MP. En otras palabras, las rutas óptimas de CP son óptimas para MP. Demostrémoslo.

Supongamos que, en la solución óptima x^* de CP forma parte de la solución cierto par (r, k) tal que r tiene exactamente un ciclo:

$$r = (0, i_1, \ldots, \underbrace{i_s, \ldots, i_t, i_s}_{\text{ciclo}}, i_v, \ldots, 0)$$

Dado que las longitudes de los arcos cumplen la desigualdad triangular, vale que $c^k_{i_t i_v} \leq c^k_{i_t i_s} + c^k_{i_s i_v}$. Así, sea $\bar{r}$ la ruta resultante de extraer la segunda visita a i_s:

$$\bar{r} = (0, i_1, \ldots, i_s, \ldots, i_t, i_v, \ldots, 0)$$

tenemos que su costo es menor que el de r. Además, es claro que la suma de la demanda de los clientes de $\bar{r}$ no supera b_k, pues (r, k) es un par factible y en $\bar{r}$ se visitan más clientes que en r. Ahora bien, si en x^* reemplazamos r por $\bar{r}$ obtenemos una solución factible de CP con menor valor en la función objetivo. Absurdo, pues x^* era solución óptima. El absurdo proviene de suponer que una ruta con un ciclo puede formar parte de la solución óptima. Un argumento análogo se puede usar para mostrar que una ruta con más de un ciclo tampoco puede formar parte de la solución óptima de CP. De esta manera, se asegura que cada cliente será visitado exactamente una vez.

Recordar que para llevar a cabo el método de Generación de Columnas, preferentemente debemos contar con un conjunto inicial de soluciones factibles para el problema restringido $RCLP$. Es decir, debemos contar con un conjunto $P' \subset P$ sobre el cual se resolverá la relajación lineal de CP. Estas soluciones factibles son inicializadas de la siguiente manera: sea $\omega(q)$ el tipo de vehículo con la capacidad más pequeña capaz de transportar carga q, se considera P' el conjunto de pares (r, k) tales que cada cliente $i \in N - \{0\}$ es atendido exclusivamente por un vehículo de tipo $\omega(q_i)$.

Finalmente, los autores introducen un preprocesamiento que permite eliminar tipos de vehículos que no formarán parte de la solución óptima. La idea es considerar $\bar{Z}$ una cota superior para el mínimo de HVRP, que viene dada por cualquier solución factible del problema. Sea $k^* \in T$, consideramos HVRP(k^*) como el problema HVRP con la restricción de que debe utilizarse al menos un vehículo de tipo k^*. Ahora bien, como resolver HVRP(k^*) es tan complejo como resolver HVRP, se considera $\underline{Z(k^*)}$ una cota inferior de HVRP(k^*). Entonces, si $\underline{Z(k^*)}$ es mayor que $\bar{Z}$, resulta que k^* no forma parte de la solución óptima y puede ser descartado. Para calcular $\underline{Z(k^*)}$ se toma en cuenta que el costo fijo de utilizar un vehículo es gran parte del costo total de sus operaciones. Por lo tanto, los autores plantean el siguiente modelo:

$$\begin{aligned}
\underline{Z(k^*)} = \quad & \text{mín} && \sum_{k \in T} f_k y_k && \\
& \text{s.a:} && \sum_{k \in T} b_k y_k && \geq \quad \sum_{i \in N - \{0\}} q_i \\
& && y_{k^*} && \geq \quad 1 \\
& && y_k && \in \quad Z_{\geq 0} \; \forall \, k \in T
\end{aligned}$$

Las variables y_k indican cuántos vehículos de tipo k serán utilizados. La primera restricción asegura que la flota de vehículos tiene la capacidad necesaria para satisfacer la demanda de todos los clientes mientras que la segunda impone la utilización de al menos un vehículo de tipo k^*. En Algoritmo 2 se encuentra el pseudocódigo del preprocesamiento.

Algoritmo 2: PREPROCESAMIENTO

Entrada: T: conjunto de tipos de vehículos
Salida: Conjunto de vehículos que podrían formar parte de la solución óptima

```
1 k* ← m
2 Calcular una cota superior Z̄
3 mientras k* > 0 hacer
4 |   Calcular Z(k*)
5 |   si Z(k*) > Z̄ entonces
6 |   |   T ← T − {k*}
7 |   k* ← k* − 1
8 devolver T
```

Para concluir esta sección, en Algoritmo 3 se resume el procedimiento propuesto por los autores para hallar buenas soluciones al HVRP.

Algoritmo 3:

```
1  Realizar preprocesamiento
2  Construir CLP inicial con P'
3  Resolver CLP
4  Resolver SP sin 2-ciclos
5  si se generó una nueva columna entonces
6  |  Añadir columna a CLP
7  |  Ir a 3
8  en otro caso
9  |  si la solución es entera entonces
10 |  |  PARAR
11 |  en otro caso
12 |  |  aplicar Branch & Bound a CP para hallar solución entera
13 |  |  PARAR
```

Resultados computacionales y comentarios

Los autores utilizan doce instancias de las veinte presentadas por Golden et al[4]. Las instancias que fueron excluidas empleaban distancias que no cumplían con la desigualdad triangular, y, como hemos demostrado, esa es una característica necesaria para que una solución óptima de CP sea óptima para MP.

A partir de esas doce instancias, crean tres grupos para probar el algoritmo. El primer grupo se emplea para medir su desempeño: se utilizan las doce instancias con los costos fijos de Golden et al. y los costos variables de Taillard[5]. En el segundo grupo, las doce instancias son consideradas con costos variables fijados en 1 para todos los tipos de vehículos. De esta manera, se pretende poder comparar con el algoritmo de Golden et al., el cual trata el HVRP con costos fijos y sin costos variables. En el tercer grupo, en las doce instancias el costo fijo de cada tipo de vehículo es nulo para poder comparar el algoritmo con el de Taillard y otros investigadores, que trataban el HVRP con costo variable y sin costo fijo.

Al ver las tablas comparativas en el paper, podemos apreciar que el algoritmo desarrollado por los autores alcanza la mejor solución conocida en la mayoría de las instancias. Si bien en las tablas se registran los tiempos de ejecución, es una medida que no ofrece un parámetro preciso pues las pruebas de los algoritmos se ejecutaron en computadoras con distintos procesadores. Los autores tienen esto en cuenta y por eso basan el éxito de su algoritmo en que ha alcanzado el mejor valor conocido en la mayoría de las instancias, dominando a los algoritmos desarrollados por los demás investigadores.

Personalmente, el conjunto de pruebas que han utilizado me parece robusto para comparar el algoritmo que desarrollaron con los que se habían presentado previamente. Sin embargo, dada la simplicidad con la que se pueden armar nuevas instancias de prueba, creo que hubiese sido interesante ampliar el conjunto del primer grupo. Es decir, probar el algoritmo en instancias con mayor cantidad de nodos y medir cómo escala su desempeño. Incluso podría llevarse a cabo un análisis para ver cómo afecta al tiempo de solución la variedad de tipos de vehículos. Más aún, hoy en día creo que analizar instancias con más nodos permitiría ver qué tan bien se desempeña para una aplicación como, por ejemplo, reparto de productos de *e-commerce* desde un depósito.

[4] Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. Computers & Operations Research, 11(1), 49–66. doi:10.1016/0305-0548(84)90007-8

[5] Taillard, E. (1999). A heuristic column generation method for the heterogeneous fleet VRP. RAIRO - Operations Research, 33(1), 1-14. doi:10.1051/ro:1999101

Siguiendo con las reflexiones sobre la aplicación a instancias reales, creo que sería interesante probar el algoritmo con clientes ubicados en un plano. En las instancias de pruebas utilizadas por los autores, se toman coordenadas en $\mathbb{R}^2$ para los puntos y se utiliza la distancia euclídea, entonces las distancias entre clientes y hasta el depósito son simétricas. Uno podría preguntarse qué sucedería si se ubican a los clientes en el plano de una ciudad y las distancias son las distancias del camino mínimo que los une. En este caso, la matriz de distancia no es necesariamente simétrica pero sigue valiendo la desigualdad triangular. Sea ℓ_{ij} la distancia del camino mínimo en un plano entre los nodos i y j. Si para algún $i, k \in N$ existiera $j \in N,\ j \neq i, k$ tal que no se cumpliera la desigualdad triangular, es decir $\ell_{ik} > \ell_{ij} + \ell_{jk}$. Entonces el camino que resulta de unir el camino mínimo de i a j y el de j a k tiene menor longitud que el camino mínimo de i a k, lo cual es absurdo.

Para saciar la curiosidad que me provocaron esos interrogantes, decidí implementarlo y probarlo en las instancias que utilizaron en el paper y generar algunas nuevas.

Implementación

Llevé a cabo la implementación en Python 3.7 utilizando las librerías de CPLEX para resolver los problemas de programación lineal y programación lineal entera. Corrí las instancias en una computadora con procesador Intel i5 `@3.00GHz` con 16 GB de RAM. Comencé armando las doce instancias en la que los autores realizaron las pruebas de desempeño de su algoritmo. Si bien los parámetros de los vehículos se encuentran en la Tabla 1 del paper, tuve que recurrir al paper de Taillard para encontrar los datos sobre la ubicación y la demanda de los clientes. Cada una de las instancias se encuentran en la carpeta `Instancias` con el nombre `golden_taillard_t.txt`, donde t corresponde al número de identificación utilizado en el paper ($3-6$ y $13-20$).

La mayor dificultad de la implementación se me presentó en el proceso de resolución de $\overline{\text{SP}}(k)$. Los autores mencionan que se puede resolver utilizando el método de etiquetado de nodos desarrollado por Desrochers, pero la referencia apunta a un artículo al que no pude acceder. Después de una larga búsqueda, hallé el método explicado en otro trabajo de Desrochers [6]. Sin embargo, muy probablemente debido a mi pobre implementación, el método demoraba mucho en encontrar la ruta con menor costo reducido. Si bien terminé optando por otro método para resolver $\overline{\text{SP}}(k)$, la implementación del método de Desrochers se encuentra en la función `pulling_algorithm` del archivo `choi_tcha_algorithm.py`.

El método que utilicé para resolver $\overline{\text{SP}}(k)$ aparece explicado en la tesis de doctorado de Petersen [7]. La idea es utilizar etiquetas que representan rutas parcialmente recorridas y que comienzan en el nodo 0. Cada etiqueta, de ser posible, es extendida a otro nodo, generando una nueva etiqueta. Este procedimiento se repite hasta que no queden etiquetas. En particular, en mi implementación, cada etiqueta L_v tiene los siguientes atributos:

- v: nodo final del camino desde el depósito
- q: carga acumulada a lo largo del camino
- *pred*: predecesor del nodo en el camino
- *path*: secuencia de nodos del camino
- *reduced_cost*: costo reducido de la ruta que resulta de agregarle al camino el arco $(v, 0)$

Notaremos como $q(L_v)$ a la carga acumulada a lo largo del camino de L_v y $pred(L_v)$ al nodo predecesor a v en el camino de L_v

[6] Desrochers, Martin & Desrosiers, Jacques & Marius, Maxmilian. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research. 40. 342-354. 10.1287/opre.40.2.342.

[7] Petersen, B. (2011). Shortest Paths and Vehicle Routing. DTU Management. PhD thesis No. 9.2011

Extender una etiqueta significa que se busca generar nuevas etiquetas a partir de extender el camino a un nodo vecino. Sea u un nodo vecino de v, se puede generar una etiqueta L_u a partir de una etiqueta L_v si se cumplen las siguientes condiciones:

- $q(L_v) + q_u \leq b_k$
- $pred(L_v) \neq u$

La primera condición busca asegurar que la capacidad del vehículo no sea excedida en la ruta que se obtiene del camino de L_u y la segunda elimina la posibilidad de obtener caminos con 2-ciclos. Si L_u es generada, es agregada a $\mathcal{L}_u$ el conjunto de etiquetas cuyo nodo final es u si no es dominada por ninguna otra etiqueta de ese conjunto. Sean L_u y L'_u dos etiquetas, diremos que L_u domina a L'_u si y sólo si $reduced\_cost(L_u) < reduced\_cost(L'_u)$. En el Algoritmo 4 se muestra el procedimiento que permite hallar una solución de $\overline{\text{SP}}(k)$.

Algoritmo 4: Algoritmo para resolver $\overline{\text{SP}}(k)$

```
L_0 ← inicializar la etiqueta correspondiente al depósito (v = 0, q = 0)
PQ.encolar(L_0)
mientras PQ ≠ ∅ hacer
    L ← primera etiqueta en la cola PQ
    para cada nodo u al que se puede extender L hacer
        L_u ← etiqueta generada a partir de la extensión de L
        si L_u no es dominada por ninguna etiqueta de L_u entonces
            Encolar L_u a PQ
            D ← conjunto de etiquetas de L_u dominadas por L_u Eliminar de L_u y de PQ cada
              etiqueta de D
sol ← etiqueta con menor costo reducido de ∪_{v∈N} L_v
devolver sol
```

Por otra parte, noté que, para que el proceso de generación de columnas funcione correctamente, es importante indicarle a CPLEX que no realice el preprocesamiento al momento de resolver CLP.

Como mencioné en el apartado anterior, me pareció interesante probar el problema en instancias con más nodos. Para eso generé cuatro nuevas instancias: dos con 200 nodos y dos con 500. Dichas instancias tienen el nombre `custom_t.txt` con $t \in \{1, 2, 3, 4\}$. Para las instancias 1 y 3 utilicé los vehículos de `golden_taillard_4` mientras que en 2 y 4 utilicé los de `golden_taillard_13`. En el Cuadro 3 se muestran los resultados obtenidos. En dicho cuadro, n indica la cantidad de clientes de la instancia, LB la cota inferior proporcionada por CLP al finalizar el procedimiento de generación de columnas, *# cols* es la cantidad de columnas generadas, *Valor* es el mejor valor de la función objetivo hallado para CP y *Tiempo* la cantidad de tiempo consumida por la totalidad del algoritmo para resolver la instancia.

Lamentablemente los resultados de mi implementación señalan que las cotas inferiores obtenidas por CLP son superiores a las obtenidas en el paper. Si bien revisé el código varias veces, no logré individualizar la causa del error. Lo positivo es que, incluso para las nuevas instancias con mayor cantidad de clientes, el algoritmo logra obtener muy buenas soluciones en poco tiempo. En particular, esto indicaría que es muy conveniente utilizarlo en aplicaciones prácticas.

Instancia	n	LB	# cols	Valor	Tiempo (segs.)
golden_taillard_3.txt	20	1156,53	26	1171,34	0,39
golden_taillard_4.txt	20	7087,49	37	8287,49	0,55
golden_taillard_5.txt	20	1375,62	23	1401,14	0,43
golden_taillard_6.txt	20	7305,67	34	9331,09	0,48
golden_taillard_13.txt	50	3070,16	40	3078,46	2,11
golden_taillard_14.txt	50	11015,08	98	14531,03	6,44
golden_taillard_15.txt	50	2989,80	140	3105,42	7,09
golden_taillard_16.txt	50	3387,53	136	3603,56	5,88
golden_taillard_17.txt	75	2188,99	428	2594,38	78,9
golden_taillard_18.txt	75	3351,57	201	3423,68	40,47
golden_taillard_19.txt	100	10313,28	307	13416,5	90,3
golden_taillard_20.txt	100	4563,72	513	4916,39	114,88
custom_1.txt	200	96046,82	463	96564,6	192,01
custom_2.txt	200	17215,15	625	19483,26	452,39
custom_3.txt	500	383771,91	603	383815,92	1068,87
custom_4.txt	500	38789,84	1487	40340,24	5462,55

Cuadro 3: Resultados obtenidos para las instancias de prueba