Introducción al algoritmo de generación de columnas.

Gabriel González Domínguez

Índice general

Abstract

This is just a little part of my master's degree's final work. In here you can find a brief introduction to column generation algorithm with a couple of applications. In one of the problem solved with this procedure there is the explained code just to try in your own computer.

1 Generación de columnas

1.1 Introducción

Sea un problema de programación entera máx$\{cx : x \in X\}$ con una región factible X que puede escribirse como la intersección de dos o más conjuntos con estructura $X = \cap_{k=0}^{K} X^k$ para un $K \geq 1$. Tómese el caso en el que las restricciones tienen la forma:

$$A^1x^1 + A^2x^2 + \cdots + A^Kx^K = b \tag{1.1}$$

$$D^1x^1 \leq d_1 \tag{1.2}$$

$$\vdots$$

$$D^Kx^K \leq d_K \tag{1.3}$$

$$x^1 \in Z_+^{n_1}, \ldots, x^K \in Z_+^{n_K} \tag{1.4}$$

de manera que los conjuntos $X^k = \{x^k \in Z_+^{n_k} : D^kx^k \leq d_k\}$ son independientes para $k = 1, \ldots, K$, y sólo las restricciones $\sum_{k=1}^{K} A^kx^k = b$ asocian entre sí los diferentes conjuntos de variables.

Dada la función objetivo máx $\sum_{k=1}^{K} c^kx^k$, dos herramientas que explotan con éxito esta estructura son la generación de cortes, con la cual se trata de validar desigualdades para los subconjuntos X^k, $k = 1, \ldots, K$; y la relajación lagrangiana, con la que se dualizan las restricciones $\sum_{k=1}^{K} A^kx^k = b$ para obtener el problema dual:

$$\min_u L(u), \tag{1.5}$$

donde

$$L(u) = \text{máx}\{\sum_{k=1}^{K}(c^k - uA^k)x^k + ub : x^k \in X^k \text{ para } k = 1, \ldots, K\}, \tag{1.6}$$

y el cálculo de $L(u)$ se divide en K subproblemas distintos:

$$L(u) = \sum_{k=1}^{K} \text{máx}\{(c^k - uA^k)x^k \,:\, x^k \in X^k\} + ub. \tag{1.7}$$

A continuación, se presenta una tercera herramienta que explota la forma de programación entera anterior. El procedimiento esencialmente resuelve un problema equivalente de la siguiente forma:

$$\text{máx}\{\sum_{k=1}^{K} \gamma^k \lambda^k \,:\, \sum_{k=1}^{K} B^k \lambda^k = \beta, \lambda^k \geq 0 \text{ entero para } k = 1, \dots K\} \tag{1.8}$$

donde cada matriz B^k tiene un gran número de columnas, una por cada punto factible en X^k, y cada vector λ^k contiene las correspondientes variables. Todas las demostraciones de los resultados y proposiciones que aparecen en este capítulo se encuentran en el libro publicado por *Wolsey* [2] en 1998.

Véase, por ejemplo, una formulación alternativa de este tipo para el problema de ubicación de instalaciones no capacitadas. Las localizaciones $j = 1, \dots, n$ se corresponden con los índices $k = 1, \dots, K$. Si el depósito j satisface la demanda del conjunto cliente S, entonces se establece $\lambda_S^j = 1$ por cada subconjunto no vacío $S \subseteq M$ de clientes. Esto conduce a la siguiente formulación:

$$\text{mín} \sum_{j \in N} \sum_{S \neq \phi} \left(\sum_{i \in S} c_{ij} + f_j \right) \lambda_S^j \tag{1.9}$$

$$\sum_{j \in N} \sum_{S \neq \phi,\, i \in S} \lambda_S^j = 1 \text{ para } i \in M \tag{1.10}$$

$$\sum_{S \neq \phi} \lambda_S^j \leq 1 \text{ para } j \in N \tag{1.11}$$

$$\lambda_S^j \in \{0, 1\} \text{ para } \phi \neq S \subseteq M, j \in N. \tag{1.12}$$

El coste λ_S^j es el coste de abrir el depósito j y atender a los clientes en S desde el depósito j. Las restricciones (1.10) imponen que se atienda a cada cliente, mientras que las restricciones (1.11) aseguran que al deposito j se le asigna como mucho un subconjunto de clientes. En la práctica las restricciones (1.12) son típicamente innecesarias.

Luego los problemas que se desean resolver son problemas enteros con un número enorme de variables, donde las columnas a menudo se describen implícitamente como los vectores de incidencia de ciertos subconjuntos de un conjunto que recorre

subconjuntos clientes. Estas grandes formulaciones de un problema entero se denominan problemas maestros. A continuación, se considera cómo resolver la relajación de los problemas maestros y comparar esta relajación con la obtenida por la relajación lagrangiana, o por el uso de planos de corte. Finalmente, se considera qué hacer cuando la solución del problema lineal no es entera, y se debe recurrir a la enumeración lo que lleva a un problema entero de generación de columnas o a algoritmos de ramificación y acotación.

1.2 Reformulación de Dantzig-Wolfe de un problema entero

Considérese el problema entero de la forma:

$$x = \text{máx}\{\sum_{k=1}^{K} c^k x^k \ : \ \sum_{k=1}^{K} A^k x^k = b, \ x^k \in X^k \text{ para } k = 1, \dots, K\} \tag{1.13}$$

donde $X^K = \{x^k \in Z_+^{n_k}; D^k x^k \leq d_k\}$ para $k = 1, \dots, K$. Asumiendo que cada conjunto X^k contiene un enorme aunque finito conjunto de puntos $\{x^{k,t}\}_{t=1}^{T_k}$, se tiene que

$$X^k = \left\{ x^k \in R^{n_k} \ : \begin{array}{c} x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{x,t}, \\ \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \\ \lambda_{k,t} \in \{0,1\} \text{ para } t = 1, \dots, T_k \end{array} \right\}. \tag{1.14}$$

Sustituyendo ahora por x^k se llega al problema entero maestro equivalente:

$$z = \text{máx} \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t}) \lambda_{k,t} \tag{1.15}$$

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b \tag{1.16}$$

$$\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \text{ para } k = 1, \dots, K \tag{1.17}$$

$$\lambda_{k,t} \in \{0,1\} \text{ para } t = 1, \dots, T_K \text{ y } k = 1, \dots, K. \tag{1.18}$$

Retomando el problema de ubicación de instalaciones no capacitadas, supóngase que se parte de la formulación débil:

$$\text{mín} \sum_{i\in M}\sum_{j\in N} c_{ij}x_{ij} + \sum_{j\in N} f_j y_j \tag{1.19}$$

$$\sum_{j\in N} x_{ij} = 1 \text{ para } i \in M \tag{1.20}$$

$$\sum_{i\in M} x_{ij} \leq m y_j \text{ para } j \in N \tag{1.21}$$

$$x \in B^{|M|\times|N|},\ y \in B^{|N|}. \tag{1.22}$$

Donde:

$$X^k = \{(x_{1k}, \ldots, x_{mk}, y_k) : \sum_{i\in M} x_{ik} \leq m y_k,\ x_{ik} \in B^1 \text{ para } i \in M,\ y_k \in \{0,1\}\}. \tag{1.23}$$

Los puntos en X^k son $\{(x_S^k, 1)\}_{S\subseteq M}$ donde x_S^k es el vector de incidencia de $S \subseteq M$, y $(0,0)$ con variables asociadas λ_S^k, ν^k que conducen al problema maestro entero:

$$\text{mín} \sum_{j\in N}\left[\sum_{S\neq\phi}\left(\sum_{i\in S} c_{ij} + f_i\right)\lambda_S^j + f_j\lambda_\phi^j\right] \tag{1.24}$$

$$\sum_{j\in N}\sum_{S\neq\phi,\, i\in S} \lambda_S^j = 1 \text{ para } i \in M \tag{1.25}$$

$$\sum_{S\neq\phi} \lambda_S^j + \lambda_\phi^j \nu^j = 1 \text{ para } j \in N \tag{1.26}$$

$$\lambda_S^j \in \{0,1\} \text{ para } S \subseteq M,\ j \in N, \nu^j \in \{0,1\} \text{ para } j \in N. \tag{1.27}$$

Obsérvese que $f_j \geq 0$, luego ν^j es dominante sobre la variable λ_ϕ^j y λ_ϕ^j puede eliminarse. Si se coge ν^j como variable de holgura en (1.11) entonces las formulaciones (1.9)-(1.12) y (1.24)-(1.27) son idénticas.

1.3 Resolución del problema maestro

Para resolver la relajación lineal del problema maestro entero se utiliza un algoritmo de generación de columnas llamado problema maestro de programación lineal.

$$z^{PML} = \text{máx} \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{k,t}) \lambda_{k,t} \tag{1.28}$$

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b \tag{1.29}$$

$$\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \text{ para } k = 1, \dots, K \tag{1.30}$$

$$\lambda_{k,t} \geq 0 \text{ para } t = 1, \dots, T^k, \ k = 1, \dots, K \tag{1.31}$$

donde hay una columna $\begin{pmatrix} c^k x \\ A^k x \\ e_k \end{pmatrix}$ para cada $x \in X^k$. Como variables duales asociadas a las restricciones (1.29) se usará $\{\pi_i\}_{i=1}^m$ y para las restricciones (1.30), llamadas restricciones de convexidad, se usarán como variables duales $\{\mu_k\}_{k=1}^K$.

La idea es resolver el problema lineal con el algoritmo simplex primal. Sin embargo, el paso de tasar y elegir una columna que entre en la base debe modificarse debido al enorme número de columnas. En lugar de tasar las columnas una a una, se resuelve el problema de encontrar la columna con el mayor coste reducido, que es en sí mismo un conjunto de K problemas de optimización.

Inicialización. Supóngase que se dispone de un subconjunto de columnas (al menos una por cada k), el cual forma un problema maestro lineal restringido factible:

$$\tilde{z}^{PMLR} = \text{máx } \tilde{c}\tilde{\lambda} \tag{1.32}$$

$$\tilde{A}\tilde{\lambda} = \tilde{b} \tag{1.33}$$

$$\tilde{\lambda} \geq 0 \tag{1.34}$$

donde $\tilde{b} = \begin{pmatrix} b \\ 1 \end{pmatrix}$, el conjunto de columnas disponible genera la submatriz:

$$\tilde{A} = \begin{pmatrix} A^1x^{1,1} & \dots & A^1x^{1,T_1} & A^2x^{2,1} & \dots & A^1x^{1,T_2} & \dots & A^Kx^{K,1} & \dots & A^Kx^{K,T_K} \\ 1 & \dots & 1 & & & & & & & \\ & & & 1 & \dots & 1 & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & 1 & \dots & 1 \end{pmatrix} \tag{1.35}$$

y $\tilde{c}$, $\tilde{\lambda}$ son los costes y variables correspondientes. Al resolver (1.32)-(1.34) se obtiene una solución optima primal $\tilde{\lambda}^*$ y una solución óptima dual $(\pi, \mu) \in R^m \times R^K$.

Factibilidad primal. Cualquier solución factible para el problema (1.32)-(1.34) es factible para el problema (1.28)-(1.31). En particular, $\tilde{\lambda}^*$ es solución factible de (1.28)-(1.31), y por tanto $\tilde{z}^{PML} = \tilde{c}\tilde{\lambda}^* = \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k \leq z^{PML}$.

Comprobación de optimalidad para el problema maestro lineal. Es necesario comprobar si (π, μ) es solución factible dual de (1.28)-(1.31). Esto implica comprobar que se cumple $c^k x - \pi A^k x - \mu_k \leq 0$ para cada $x \in X^k$ de cada k, de cada columna. En lugar de examinar cada punto de forma separada, se examinan todos los puntos X^k implícitamente resolviendo el subproblema de optimización:

$$\zeta_k = \text{máx}\{(c^k - \pi A^k)x - \mu_k \,:\, x \in X^k\} \tag{1.36}$$

Criterio de parada. Si $\zeta_k = 0$ para $k = 1, \dots, K$, entonces la solución dual (π, μ) es la solución dual factible del problema maestro lineal y por tanto $z^{PML} \leq \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k$. Como el valor de la solución factible primal $\tilde{\lambda}$ es igual al de su cota superior, $\tilde{\lambda}$ es óptimo para el problema maestro lineal.

Generación de una nueva columna. Si $\zeta_k > 0$ para un k, entonces la columna correspondiente a la solución óptima $\tilde{x}^k$ del subproblema tiene un coste reducido positivo. Introduciendo la columna $\begin{pmatrix} c^k\tilde{x} \\ A^k\tilde{x} \\ e_k \end{pmatrix}$ se obtiene un nuevo problema maestro linear restringido que se puede reoptimizar (por ejemplo con el algoritmo simplex primal).

Una cota dual (superior). Del subproblema se tiene $\zeta_k \geq (c^k - \pi A^k)x - \mu_k$ para todo $x \in X^k$. O lo que es lo mismo $(c^k - \pi A^k)x - \mu_k - \zeta_k \leq 0$ para todo $x \in X^k$. Luego estableciendo $\zeta = (\zeta_1, \dots, \zeta_K)$, se tiene que $(\pi, \pi + \zeta)$ es solución factible dual del

problema maestro lineal. Luego:

$$z^{PML} \leq \pi b + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta_k. \tag{1.37}$$

Todas estas observaciones conducen directamente a un algoritmo para el problema maestro lineal que termina cuando $\zeta_k = 0$ para $k = 1, \dots, K$. Sin embargo, como en la relajación lagrangiana, es posible terminar antes.

Un criterio de parada alternativo. Si la solución del subproblema $(\tilde{x}^1, \dots, \tilde{x}^K)$ satisface las restricciones $\sum_{k=1}^{K} A^k x^k = b$, entonces $(\tilde{x}^1, \dots, \tilde{x}^K)$ es óptima. Esto es así puesto que $\zeta_k = (c^k - \pi A^k)\tilde{x}^k - \mu_k$ implica que $\sum_k c^k \tilde{x}^k = \sum_k \pi A^k \tilde{x}^k + \sum_k \mu_k + \sum_k \zeta_k = \pi b + \sum_k \mu_k + \sum_k \zeta_k$. Luego la solución factible tiene el mismo valor que la cota superior de z^{PML}.

1.4 Eficacia del problema maestro lineal

¿Cómo de eficaz es la relajación del problema lineal del problema maestro? ¿Hay alguna esperanza de que resuelva el problema original entero?

Proposición 1.1.

$$z^{PML} = \text{máx} \left\{ \sum_{k=1}^{K} c^k x^k \ : \ \sum_{k=1}^{K} A^k x^k = b, x^j \in conv(X^k) \quad \text{para} \quad k = 1, \dots, K \right\}. \tag{1.38}$$

Cuando un problema entero se puede descomponer, dos herramientas para resolverlo son la relajación lagrangiana y los planos de corte. Sea w_L el valor de la solución dual lagrangiana cuando la restricción $\sum_{k=1}^{K} A^k x^k = b$ está dualizada, y sea z^C el valor obtenido cuando se añaden los planos de corte a la relajación del problema lineal entero usando un algoritmo de separación exacto para cada uno de los conjuntos $conv(X^k)$ para $k = 1, \dots, K$. Tanto estos procedimientos como la generación de columnas son de cierta forma equivalentes en tanto en cuanto conducen hasta las mismas cotas de la solución dual.

Proposición 1.2.

$$z^{PML} = w_L = z^C \tag{1.39}$$

Como el subproblema resuelto tanto en el algoritmo de generación de columnas como en la relajación lagrangiana son problemas de optimización sobre X^k, puede visualizarse la generación de columnas como un algoritmo para resolver la relajación lagrangiana en el cual las variables duales π se actualizan resolviendo el problema maestro lineal restringido. Este algoritmo se usa a menudo junto al algoritmo del subgradiante para resolver la relajación lagrangiana que se basa en un procedimiento de actualización mucho más simple.

Por otro lado, si se usan los planos de corte, aunque las cotas obtenidas sean potencialmente las mismas, se resuelven problemas de separación en $conv(X^K)$ en lugar de problemas de optimización. Aunque la complejidad de los problemas de optimización y los problemas de separación en $conv(X^k)$ sea teóricamente la misma, la elección depende de la dificultad relativa para resolver los dos problemas, así como de la convergencia de los algoritmos de generación de columnas y plano de corte en la práctica.

1.5 Resolución del problema del viajante simétrico mediante generación de columnas: un ejemplo ilustrativo

A continuación, se aplica el algoritmo anteriormente expuesto para resolver el problema maestro lineal de un problema en el cual hay sólo un subproblema. Considérese el problema del viajante simétrico que puede formularse como:

$$\text{mín} \left\{ \sum_{e \in E} c_e x_e \; : \; \sum_{e \in \delta(i)} x_e = 2 \text{ para } i \in N, \; x \in X^1 \right\}, \tag{1.40}$$

donde:

$$X^1 = \left\{ x \in Z_+^m \; : \; \begin{array}{c} \sum_{e \in \delta(1)} x_e = 2, \\ \sum_{e \in E(S)} x_e \leq |S| - 1 \text{ para } \phi \subset S \subset N \setminus \{1\}, \\ \sum_{e \in E} x_e = n \end{array} \right\}, \tag{1.41}$$

es el conjunto de vectores de incidencia de 1-árboles.

Escribiendo $x_e = \sum_{t : e \in E^t} \lambda_t$, donde $G^t = (N, E^t)$ es el t-ésimo 1-árbol, las restricciones que afectan al grado se convierten en $\sum_{e \in \delta(i)} x_e = \sum_{e \in \delta(i)} \sum_{t : e \in E^t} \lambda_t = \sum_t d_i^t \lambda_t = 2$

donde d_i^t es el grado del nodo i en el 1-árbol G^t. Por lo tanto, el correspondiente problema maestro lineal es:

$$\text{mín} \sum_{t=1}^{T_1} (cx^t)\lambda_t, \tag{1.42}$$

$$\sum_{t=1}^{T_1} d_i^t \lambda_t = 2 \text{ para } i \in N, \tag{1.43}$$

$$\sum_{t=1}^{T_1} \lambda_t = 1, \tag{1.44}$$

$$\lambda \in R_+^T, \tag{1.45}$$

al que se asocia variables duales $\{u_i\}_{i=1}^n$ en las restricciones de grado, y variables duales μ a las restricciones de convexidad. El subproblema correspondiente es:

$$\zeta_1 = \text{mín} \left\{ \sum_{e \in E} (c_e - u_i - u_j) x_e - \mu \,:\, x \in X^1 \right\}, \tag{1.46}$$

puesto que el coste reducido del 1-árbol $cx^t - \sum_{i \in N} d_i^t u_i - \mu = cx^t - \sum_{i \in N} u_i \sum_{e \in \delta(i)} x_e^t - \mu = \sum_{e \in E} (c_e - u_i - u_j) x_e^t - \mu$, donde x_e^t con $e \in E$ son las variables correspondientes a las aristas del 1-árbol G^t, y $e = (i, j)$ para $e \in E$.

Debido a que se trabaja con 1-árboles, $d_i^t = 2$ para todo t, y por tanto, (1.43) es dos veces (1.44), luego se puede desechar la restricción (1.44).

Considérese una instancia del problema del viajante simétrico con matriz de distancias:

$$c_e = \begin{pmatrix} \cdot & 7 & 2 & 1 & 5 \\ & \cdot & 3 & 6 & 8 \\ & & \cdot & 4 & 2 \\ & & & \cdot & 9 \\ & & & & \cdot \end{pmatrix} \tag{1.47}$$

Se parte de un problema maestro lineal restringido que tiene 7 columnas correspon-

dientes a un recorrido de longitud 28 y seis 1-árboles elegidos arbitrariamente:

$$\begin{array}{lccccccccc}
\text{mín} & 28\lambda_1 & +25\lambda_2 & +21\lambda_3 & +19\lambda_4 & +22\lambda_5 & +18\lambda_6 & +28\lambda_7 & & \\
& 2\lambda_1 & +2\lambda_2 & +2\lambda_3 & +2\lambda_4 & +2\lambda_5 & +2\lambda_6 & +2\lambda_7 & = & 2 \\
& 2\lambda_1 & +2\lambda_2 & +2\lambda_3 & +1\lambda_4 & +1\lambda_5 & +2\lambda_6 & +3\lambda_7 & = & 2 \\
& 2\lambda_1 & +3\lambda_2 & +2\lambda_3 & +3\lambda_4 & +2\lambda_5 & +3\lambda_6 & +1\lambda_7 & = & 2 \\
& 2\lambda_1 & +2\lambda_2 & +3\lambda_3 & +3\lambda_4 & +3\lambda_5 & +1\lambda_6 & +1\lambda_7 & = & 2 \\
& 2\lambda_1 & +1\lambda_2 & +1\lambda_3 & +1\lambda_4 & +2\lambda_5 & +2\lambda_6 & +3\lambda_7 & = & 2 \\
& & & \lambda & \geq & 0. & & & &
\end{array} \tag{1.48}$$

La solución es $\lambda = (0, 0, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ con coste 22,5 y su solución dual es $u = (\frac{151}{8}, \frac{1}{3}, -\frac{5}{3}, \frac{2}{3}, 0)$. La correspondiente matriz de costes reducidos para este subproblema es:

$$\begin{pmatrix}
\cdot & -\frac{87}{8} & -\frac{91}{8} & -\frac{133}{8} & -\frac{111}{8} \\
 & \cdot & \frac{19}{2} & \frac{33}{4} & 9 \\
 & & \cdot & \frac{43}{4} & \frac{15}{2} \\
 & & & \cdot & \frac{41}{4} \\
 & & & & \cdot
\end{pmatrix}. \tag{1.49}$$

El 1-árbol optimal es $x_{14} = x_{15} = x_{24} = x_{25} = x_{35} = 1$ con $\zeta = -\frac{23}{4}$. Por lo que $22,5 + \zeta = 16.75 \leq z^{LP} \leq 22.5$.

Comienza una nueva iteracción introduciendo este 1-árbol como una nueva columna en el problema maestro restringido con coste 22, y grados $(2, 2, 1, 2, 3)$. La solución del nuevo problema lineal es $\lambda = (0, 0, \frac{1}{3}, 0, 0, \frac{1}{3}, 0, \frac{1}{3})$ con coste 20.333 y solucion dual $u = (\frac{65}{6}, \frac{1}{3}, -\frac{5}{3}, \frac{2}{3}, 0)$.

La correspondiente matrix de costes reducidos para el subproblema es:

$$\begin{pmatrix}
\cdot & -\frac{25}{8} & -\frac{43}{8} & -\frac{21}{2} & -\frac{35}{6} \\
 & \cdot & \frac{13}{3} & 5 & \frac{23}{3} \\
 & & \cdot & 5 & \frac{11}{3} \\
 & & & \cdot & \frac{25}{3} \\
 & & & & \cdot
\end{pmatrix}. \tag{1.50}$$

El 1-árbol óptimo es $x_{13} = x_{14} = x_{23} = x_{24} = x_{35} = 1$ con $\zeta = -\frac{14}{3}$. La cota inferior es $20,333 - \frac{14}{3} = 15,667$ y no es tan bueno como el obtenido anteriormente. Por lo tanto, se tiene $16,75 \leq Z^{LP} \leq 30,333$.

De nuevo, se introduce este 1-árbol como una nueva columna en el problema maestro restringido con coste 14 y grados $(2, 2, 3, 2, 1)$. La nueva solución del problema lineal es $\lambda = (0, 0, 0, 0, 0, 0, 0, \frac{1}{2}, \frac{1}{2})$ con coste 18 y solución dual $u = (13, 0, -4, 0, 0)$.

La correspondiente matriz de costes reducidos para el subproblema es:

$$\begin{pmatrix} \cdot & -6 & -7 & -12 & -8 \\ & \cdot & 7 & 6 & 8 \\ & & \cdot & 8 & 6 \\ & & & \cdot & 9 \\ & & & & \cdot \end{pmatrix}. \tag{1.51}$$

El 1-árbol óptimo es $x_{14} = x_{15} = x_{23} = x_{24} = x_{35} = 1$ con $\zeta = -1$. La cota inferior de la solución aumenta en $18 - 1 = 17$. Como este 1-árbol es un tour, del principio de parada alternativo se deduce que es óptimo. De forma alternativa, se puede comprobar que el coste real es 17.

2 Otro ejemplo de aplicación del algoritmo de generación de columnas y su implementación en python

2.1 Resolución problema de corte de material mediante generación de columnas

Para explicar en detalle la idea de la generación de columnas se resuelve a continuación un ejemplo sencillo de problema de corte de material que aparece en el libro publicado por *Winston* [1] en 1990.

Una empresa vende tablas de madera de 3, 5 y 9 metros. Los clientes de la empresa quieren 25 tablas de 3 metros, 20 tablas de 5 metros y 15 tablas de 9 metros. La empresa tiene que cortar tablones de 17 metros y desea malgastar la mínima madera posible. Se formula un programa lineal y se resuelve este mediante generación de columnas parar alcanzar este objetivo.

La empresa debe planear como cortar cada tablón de 17 metros, cada plan corresponde a una forma de cortar una de estas tablones. Así, una variable plan sería cortar el tablón en 3 tablas de 5 metros, lo cual resultaría en desperdiciar 17-15 = 2 metros de madera. Sería tonto cortar un tablón en sólo tablas de 9 y 5 metros cuando se puede cortar en tablas de 9, 5 y 3 metros. En general, cualquier patrón de corte que desperdicie 3 o más metros de madera es desechado. La Tabla1 enumera las formas sensatas de cortar

el tablón de 17 metros.

Se define ahora:

$$x_i = \text{número de tablas de 17 metros cortadas de acuerdo con la combinación } i \quad (2.1)$$

y se formula el programa lineal:

$$\text{sobrante} + \text{demanda total} = \text{longitud del tablón} \quad (2.2)$$

dado que

$$\text{Demanda total de los usuarios} = 25(3) + 20(5) + 15(9) = 310 \text{ metros} \quad (2.3)$$

$$\text{Largo total del tablón a cortar} = 17(x_1 + x_2 + x_3 + x_4 + x_5 + x_6) \quad (2.4)$$

se escribe

$$\text{sobrante (metros)} = 17x_1 + 17x_2 + 17x_3 + 17x_4 + 17x_5 + 17x_6 - 310 \quad (2.5)$$

Luego la función objetivo de la empresa es:

$$\text{mín } z = 17x_1 + 17x_2 + 17x_3 + 17x_4 + 17x_5 + 17x_6 - 310 \quad (2.6)$$

Lo cual es equivalente a minimizar:

$$\text{mín } z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \quad (2.7)$$

Esto significa que la empresa puede minimizar el sobrante de madera minimizando el número de tablones de 17 metros.

La empresa está sujeta a las siguientes restricciones:

	3 metros	5 metros	9 metros	Sobrante
Comb. 1	5	0	0	2
Comb. 2	4	1	0	0
Comb. 3	2	2	0	1
Comb. 4	2	0	1	2
Comb. 5	1	1	1	0
Comb. 6	0	3	0	2

Cuadro 2.1: Número de tablas de cada tipo por tablón.

- **Restricción 1.** Se deben cortar al menos 25 tablas de 3 metros. Lo cual se traduce en:

$$5x_1 + 4x_2 + 2x_3 + 2x_4 + x_5 \geq 25 \tag{2.8}$$

- **Restricción 2.** Se debe cortar al menos 20 tablas de 5 metros. Lo cual se traduce en:

$$x_2 + 2x_3 + x_5 + 3x_6 \geq 20 \tag{2.9}$$

- **Restricción 3.** Se debe cortar al menos 15 tablas de 9 metros. Lo cual se traduce en:

$$x_4 + x_5 \geq 15 \tag{2.10}$$

Los coeficientes de x_i de la restricción para las tablas de k metros es el número de tablas de k metros producidas con la combinación i.

Es necesario que x_i tome valores enteros. En problemas con grandes demandas, resolviendo el problema de corte de stock como un problema lineal y redondeando por arriba las variables fraccionarias se puede obtener una solución casi óptima. Este procedimiento no siempre alzanca la mejor solución entera. Se tiene el siguiente problema lineal:

$$\text{mín } z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \tag{2.11}$$
$$\tag{2.12}$$

sujeto a

$$5x_1 + 4x_2 + 2x_3 + 2x_4 + x_5 \geq 25 \tag{2.13}$$
$$x_2 + 2x_3 + x_5 + 3x_6 \geq 20 \tag{2.14}$$
$$x_4 + x_5 \geq 15 \tag{2.15}$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \tag{2.16}$$

x_1 sólo aparece en (2.13) y x_6 sólo en (2.14), con lo cual, x_1 y x_6 se pueden utilizar como variables básicas iniciales para estas restricciones. Para (2.15) de 9 metros no hay variables básicas iniciales. Para evitar añadir una variable artificial sólo en el caso de está restricción, se define una combinación 7 en la que se corta una única tabla de 9 metros. x_7 será nula en la solución óptima pero su inclusión en la base inicial

evita usar la M Grande o el método simplex de dos fases. La columna para x_7 en las restricciones del problema lineal será

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.17}$$

y se añadirá un término x_7 a la función objetivo. Se puede usar ahora $BV = \{x_1, x_6, x_7\}$ como base inicial. Luego se tiene:

$$B_0 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

$$B_0^{-1} = \begin{bmatrix} \frac{1}{5} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

$$\tag{2.20}$$

Luego

$$c_{BV} B_0^{-1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{5} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{5} & \frac{1}{3} & 1 \end{bmatrix} \tag{2.21}$$

Si se tasan las variables no básicas, puede verse qué variables entran en la base. Sin embargo, en un problema de corte de stock a gran escala puede haber miles de variables y tasar cada variable no básica puede ser una tarea tediosa. Es en este tipo de situaciones donde la generación de columnas es útil. Puesto que se está minimizando, se desea encontrar una columna que tase positivamente (tenga un coeficiente positivo en la fila 0). En el problema de corte de stock, cada columna o variable , representa una combinación de cortes para el tablón: una variable se especifica con 3 números: a_3, a_5 y a_9. Donde a_i es el número de tablas de i metros que resultan de cortar el tablón de 17 metros. Por ejemplo, la variable x_2 viene dada por $a_3 = 4$, $a_5 = 1$ y $a_9 = 0$. La idea de la generación de columnas es buscar de forma eficiente una columna con una tasación favorable (positiva en un problema de minimizar y negativa si se quiere maximizar). Para la base de la que ahora se dispone, especificando a_3, a_5 y a_9 se tasa como

$$c_{BV} B_0^{-1} \begin{bmatrix} a_3 \\ a_5 \\ a_9 \end{bmatrix} - 1 = \frac{1}{5} a_3 + \frac{1}{3} a_5 + a_9 - 1 \tag{2.22}$$

Se debe elegir a_3, a_5 y a_9 para que sean enteros no negativos y no sobrepase los 17 metros de madera.

Toda combinación de a_3, a_5 y a_9 debe cumplir:

$$3a_3 + 5a_5 + 9a_9 \leq 17 \quad (a_3 \geq 0, a_5 \geq 0, a_9 \geq 0; a_3, a_5, a_9 \text{ entero}) \tag{2.23}$$

Se encuentra la combinación que mejor tasa resolviendo el siguiente problema de mochila

$$\text{máx } z = \frac{1}{5}a_3 + \frac{1}{3}a_5 + a_9 - 1 \tag{2.24}$$

$$\text{sujeto a} \quad 3a_3 + 5a_5 + 9a_9 \leq 17 \tag{2.25}$$

$$a_3, a_5, a_9 \geq 0; a_3, a_5, a_9 \text{ enteros} \tag{2.26}$$

Dado que (2.25) es un problema de mochila (sin restringir las variables a 0-1), se puede resolver con un procedimiento de ramificación y acotación, reflejado en la Figura 2.1.

En la Figura 2.1, para resolver el problema 6 primero se establece $a_5 = 1$(dado que $a_5 \geq 1$)

En la Figura 2.1 puede verse que la solución óptima del problema lineal (2.25) es $z = \frac{8}{15}$, $a_3 = a_5 = a_9 = 1$. Esto corresponde a la combinación 5 y la variable x_5. Así, x_5 se tasa en $\frac{8}{15}$ y al entrar x_5 en la base, el sobrante de madera disminuye. Para introducir x_5 en la base se procede como sigue:

$$\text{columna } x_5 \text{ en el cuadro actual } = B_0^{-1}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}\frac{1}{5} & 0 & 0\\ 0 & \frac{1}{3} & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}\frac{1}{5}\\ \frac{1}{3}\\ 1\end{bmatrix} \tag{2.27}$$

$$\text{Parte derecha del cuadro actual } = \begin{bmatrix}\frac{1}{5} & 0 & 0\\ 0 & \frac{1}{3} & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}25\\20\\15\end{bmatrix} = \begin{bmatrix}5\\ \frac{20}{3}\\ 15\end{bmatrix} \tag{2.28}$$

Así queda $BV(1) = \{x_1, x_6, x_5\}$. Aplicando el producto de la inversa se obtiene:

$$B_1^{-1} = E_0 B_0^{-1} = \begin{bmatrix}1 & 0 & -\frac{1}{5}\\ 0 & 1 & -\frac{1}{3}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\frac{1}{5} & 0 & 0\\ 0 & \frac{1}{3} & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{5} & 0 & -\frac{1}{5}\\ 0 & \frac{1}{3} & -\frac{1}{3}\\ 0 & 0 & 1\end{bmatrix} \tag{2.29}$$

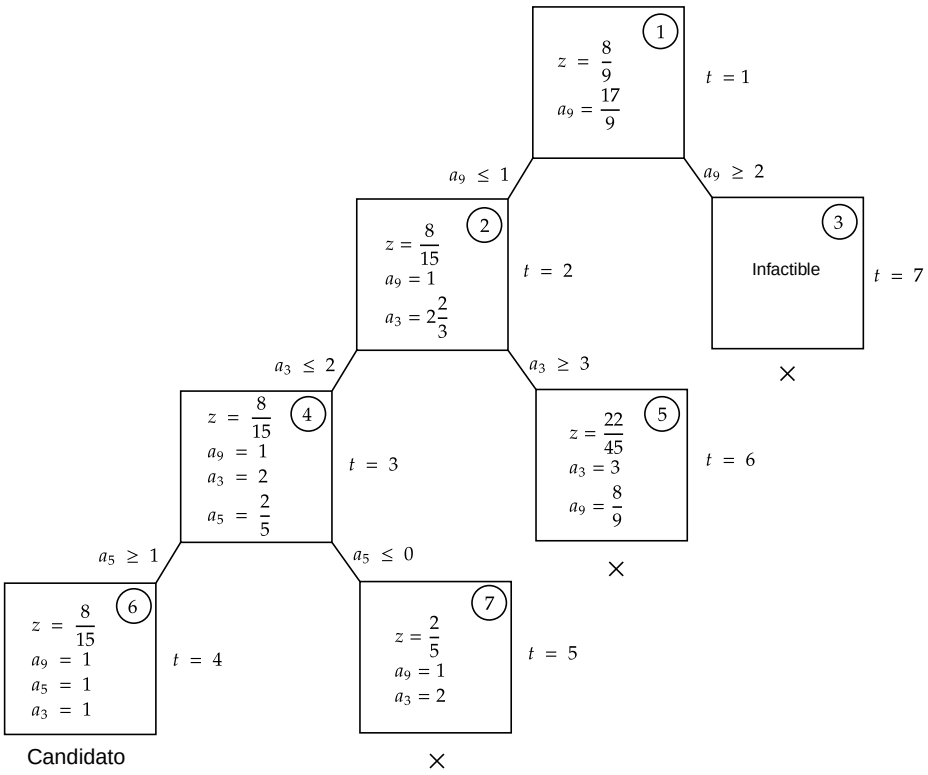

Figura 2.1: Ramificación y acotación en el árbol del problema lineal (2.25)

Ahora

$$c_{BV}\boldsymbol{B}_1^{-1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{5} & 0 & -\frac{1}{5} \\ 0 & \frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{5} & \frac{1}{3} & \frac{7}{15} \end{bmatrix} \tag{2.30}$$

Con el nuevo precio de referencia ($c_{BV}\boldsymbol{B}_1^{-1}$), se puede usar generación de columnas de nuevo para determinar si alguna combinación debe de incluirse en la base. Con los precios de referencia actuales

$$\begin{bmatrix} \frac{1}{5} & \frac{1}{3} & \frac{7}{15} \end{bmatrix} \begin{bmatrix} a_3 \\ a_5 \\ a_9 \end{bmatrix} - 1 = \frac{1}{5}a_3 + \frac{1}{3}a_5 + \frac{7}{15}a_9 - 1 \tag{2.31}$$

Para el cuadro actual, el procedimiento de generación de columnas conduce al si-

guiente problema:

$$\text{máx } z = \frac{1}{5}a_3 + \frac{1}{3}a_5 + \frac{7}{15}a_9 - 1 \tag{2.32}$$

$$\text{sujeto a} \quad 3a_3 + 5a_5 + 9a_9 \leq 17 \tag{2.33}$$

$$a_3, a_5, a_9 \geq 0;\ a_3, a_5, a_9 \text{ enteros} \tag{2.34}$$

El árbol de ramificación y corte para (2.33) aparece en la Figura 2.2. La combinación que mejor tasa es $a_3 = 4$, $a_5 = 1$ y $a_9 = 0$ (combinación 2). Tendrá un coeficiente $\frac{2}{15}$ en la fila 0. Luego se introduce x_2 en la base, con el cuadro actual la columna para x_2 es:

$$B_1^{-1}\begin{bmatrix}4\\1\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{5} & 0 & -\frac{1}{5}\\ 0 & \frac{1}{3} & -\frac{1}{3}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}4\\1\\0\end{bmatrix} = \begin{bmatrix}\frac{4}{5}\\ \frac{1}{3}\\ 0\end{bmatrix} \tag{2.35}$$

El lado derecho del cuadro actual es:

$$B_1^{-1}b = \begin{bmatrix}\frac{1}{5} & 0 & -\frac{1}{5}\\ 0 & \frac{1}{3} & -\frac{1}{3}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}25\\20\\15\end{bmatrix} = \begin{bmatrix}2\\ \frac{5}{3}\\ 0\end{bmatrix} \tag{2.36}$$

Así queda $BV(2) = \{x_2, x_6, x_5\}$. Aplicando el producto de la inversa se obtiene:

$$E_1 = \begin{bmatrix}\frac{5}{4} & 0 & 0\\ -\frac{5}{12} & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2.37}$$

Luego

$$B_2^{-1} = E_1 B_1^{-1} = \begin{bmatrix}\frac{5}{4} & 0 & 0\\ -\frac{5}{12} & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\frac{1}{5} & 0 & -\frac{1}{5}\\ 0 & \frac{1}{3} & -\frac{1}{3}\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{4} & 0 & -\frac{1}{4}\\ -\frac{1}{12} & \frac{1}{3} & -\frac{1}{4}\\ 0 & 0 & 1\end{bmatrix} \tag{2.38}$$

El nuevo conjunto de precios de referencia viene dado por

$$c_{BV}B_2^{-1} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}\frac{1}{4} & 0 & -\frac{1}{4}\\ -\frac{1}{12} & \frac{1}{3} & -\frac{1}{4}\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{6} & \frac{1}{3} & \frac{1}{2}\end{bmatrix} \tag{2.39}$$

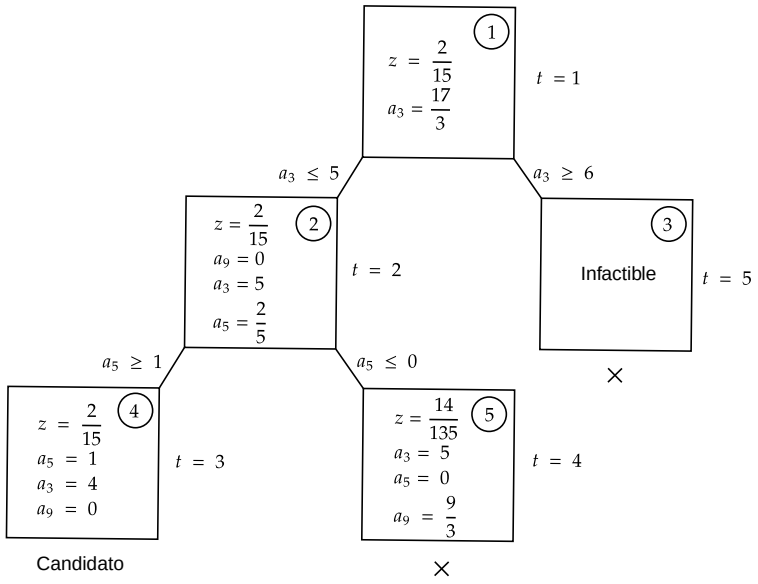

Figura 2.2: Ramificación y acotación en el árbol del problema lineal (2.33)

Para estos precios de referencia, una combinación especificada por a_3, a_5 y a_9 se tasará como $\frac{1}{6}a_3 + \frac{1}{3}a_5 + \frac{1}{2}a_9 - 1$. Así, el procedimiento de generación de columnas requiere la resolución del siguiente problema:

$$\text{máx } z = \frac{1}{6}a_3 + \frac{1}{3}a_5 + \frac{1}{2}a_9 - 1 \tag{2.40}$$

$$\text{sujeto a} \quad 3a_3 + 5a_5 + 9a_9 \leq 17 \tag{2.41}$$

$$a_3, a_5, a_9 \geq 0; \;\; a_3, a_5, a_9 \text{ enteros} \tag{2.42}$$

El valor óptimo para (2.40) es $z = 0$. Esto significa que ninguna combinación tasa convenientemente, por lo que, la solución actual debe de ser la solución óptima. Para conocer los valores de las variables de la base en la solución óptima, se halla el lado derecho de el cuadro actual:

$$B_2^{-1}b = \begin{bmatrix} \frac{1}{4} & 0 & -\frac{1}{4} \\ -\frac{1}{12} & \frac{1}{3} & -\frac{1}{4} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 25 \\ 20 \\ 15 \end{bmatrix} = \begin{bmatrix} 5/2 \\ 5/6 \\ 15 \end{bmatrix} \tag{2.43}$$

Luego la solución óptima para el problema de corte de stock es $x_2 = \frac{5}{2}, x_6 = \frac{5}{6}, x_5 = 15$. Redondeando por arriba x_2 y x_6 se obtiene la solución $x_2 = 3$, $x_6 = 1$, $x_5 = 15$.

2.2 Implementación del problema de corte de material

Generalizando el problema 2.11-2.16 se tiene el siguiente problema de programación lineal entera:

$$\text{mín} \sum_{p} c_p x_p \tag{2.44}$$

$$\text{sujeto a} \quad \sum_{p} n_{pj} x_p \geq d_j \qquad \forall j \in PI \tag{2.45}$$

donde los índices:

- $p \in PA$ es el conjunto de todos los posibles patrones de corte.
- $j \in PI$ es el conjunto de piezas de diferentes tamaños a trocear.

Los parámetros:

- c_p coste asociado al patrón de corte p.
- n_{pj} es el número de piezas de tipo j que se generan al cortar el patrón p.
- d_j es la demanda total de piezas de tipo j.

La variable:

- $x_p \in \mathbb{Z}^+$ cantidad de veces que se corta la tabla original siguiendo el patrón p.

Para resolver este problema se usa Python 3.8.8 y Gurobi Optimizer version 9.1.2. El código procede de un vídeo de youtube salvo algunas modificaciones hechas por mí.

Gurobi no permite manejar el branch-and-price, por lo que no puede resolver un problema de programación entera mixta de manera óptima mientras se van agregando variables con costes reducidos negativos sobre la marcha.

Lo que se hace es resolver la relajación de tu modelo con generación de columnas. Una vez hecho esto, te planteas resolver el problema con, supón, 114 millones de variables, pero implícitamente fijas muchas de ellas en cero (simplemente no las agregas al modelo). Este modelo con esas variables "fijadas a cero" se denomina problema maestro

restringido (RMP). Se resuelve este problema y si todas las variables fijadas a cero (al no tenerlas en el modelo) tienen costes reducidos no negativos (para un problema de minimización), entonces la solución RMP también es óptima para el problema completo.

Se parte generando un conjunto inicial de patrones:

```
class InitialPatternsGenerator:
    def __init__(self, nbItems):
        columns = ['PatternCost', 'PatternFill']
        patterns = pd.DataFrame(index = range(nbItems), columns = columns)
        self.patternDf = patterns
        self.nbItems = nbItems

    # Original patterns: Use each raw only for i cutted roll, regardless the
    # size
    def generateBasicInitialPatterns(self):
        self.patternDf['PatternCost'] = 1
        self.patternDf['PatternFill'] = [np.where(self.patternDf.index == j, 1, 0)
                                         for j in range(self.nbItems)]
        return self.patternDf
```

Por ejemplo, partiendo de tablas de tamaño 17 se desean 25 tablas de tamaño 3, 50 de tamaño 5 y 15 de tamaño 9:

```
data = [[3, 25], [5, 20], [9, 15]]
inputDf = pd.DataFrame(data, columns=['Size', 'Amount'])
originalBoardSize = 17
patternGenerator = InitialPatternsGenerator(len(inputDf))
patternDf = patternGenerator.generateBasicInitialPatterns()
# The size of the original boards is 17.
# Requeriments:
#    Size  Amount
# 0     3      25
# 1     5      20
# 2     9      15
# Cost and Patterns
#    PatternCost  PatternFill
# 0            1   [1, 0, 0]
# 1            1   [0, 1, 0]
# 2            1   [0, 0, 1]
```

El cualquier patrón es unitario pero podría cambiarse el código para que cada patrón tuviese un coste de acuerdo con algún criterio. Los patrones de corte inicial como se puede ver consisten en una pieza por cada tabla, que son patrones realmente malos.

El problema maestro coincide con 2.45. Se genera una variable para cada uno de los distintos patrones con cota inferior 0 y cota superior $\sum_j d_j$ que coincide con una solución en la que sólo apareciese un único patrón de corte. El correspondiente código es:

```
class MasterProblem:
    def __init__(self, patternDf, inputDf):
        self.patternCost = patternDf['PatternCost'].values
        self.pattern= patternDf[ 'PatternFill'].values
        self.amount = inputDf['Amount'].values
        self.model = gb.Model("MasterProblem")
        self.patternsIndex = patternDf.index.values

    def buildModel(self):
        self.generateVariables()
        self.generateConstraints()
```

```
        self.generateObjective()
        self.model.update()

    def generateVariables(self):
        # PatternUseVar (p) \in [0, total Amount]: Number of times the pattern 'p'
        # is used in the solution.
        self.patternUseVar = self.model.addVars(self.patternsIndex, lb = 0,
                                                ub = sum(self.amount),
                                                vtype = gb.GRB.INTEGER,
                                                name="PatternUseVar")

    def generateConstraints(self):
        # It must be cutted, at least as many pieces as specified in the input data
        for i in range (len(self.patternsIndex)):
            self.model.addConstr(gb.quicksum(
                self.pattern[p][i] * self.patternUseVar[p]
                for p in self.patternsIndex) >= self.amount[i],
                'C'+str(i))

    def generateObjective(self):
        # Minimize the total cost of the used rolls
        self.model.setObjective(gb.quicksum (
            self.patternUseVar[p] * self.patternCost[p]
            for p in self.patternsIndex),
            sense = gb.GRB.MINIMIZE)
```

Una vez resuelto el problema maestro se busca aquel patrón de corte que en mayor medida mejore la solución. Para esto se resuelve el master problem relajado ($x_p \in \mathbb{R}$) y se obtiene a partir de él las variables duales π_i asociadas a las restricciones. Estos son los costes reducidos que señalan cuánto afecta a la función objetivo el nuevo patrón de corte. Se resuelve el problema relajado con:

```
    def solveRelaxedModel(self):
        # Relaxed integer variables to continuous variables
        self.relaxedModel = self.model.relax()
        self.relaxedModel.optimize()
```

El coste reducido es el atributo "Pi" de las restricciones:

```
    def getDuals(self):
        return self.relaxedModel.getAttr("Pi", self.model.getConstrs())
```

A continuación, se resuelve el siguiente subproblema de mochila:

$$\text{máx} \sum_j \pi_j y_j \tag{2.46}$$

$$\text{sujeto a} \quad \sum_j t_j y_j \leq T \qquad \forall j \in PI \tag{2.47}$$

donde los parámetros:

- T es el tamaño original de la tabla.
- t_j es el tamaño de la pieza cortada.
- π_j es el coste reducido correspondiente a la restricción de cumplimiento de demanda del problema maestro para las piezas de tamaño j.

La variable:

- $y_j \in \mathbb{Z}^+$ cantidad de piezas de tamaño j en el nuevo patrón.

El correspondiente código es:

```
class Subproblem:
    def __init__(self, inputDf, originalBoardSize, duals):
        self.patternCost = patternDf['PatternCost'].values
        self.pattern = patternDf['PatternFill'].values
        self.amount = inputDf['Amount'].values
        self.pieceSize = inputDf['Size'].values
        self.originalBoardSize = originalBoardSize
        self.duals = duals
        self.model = gb.Model('Subproblem')
        self.piecesIndex = inputDf.index.values

    def buildModel(self):
        self.generateVariables()
        self.generateConstraints()
        self.generateObjective()
        self.model.update()

    def generateVariables(self):
        # PiecesInPatternVar(p) \in [0, amount(piece)]: Number of times the piece
        # p is cutted in the new pattern
        self.piecesInPatternVar = self.model.addVars(self.piecesIndex, lb = 0,
                                                     ub = self.amount,
                                                     vtype = gb.GRB.INTEGER,
                                                     name = 'PiecesInPatternsVar')

    def generateConstraints(self):
        # The size of the raw Roll limits the pieces that van be cutted
        self.model.addLConstr(gb.quicksum(
            self.piecesInPatternVar[p] * self.pieceSize[p]
            for p in self.piecesIndex) <= self.originalBoardSize,
            'RollSizeCt')

    def generateObjective(self):
        # Maximize de profit of the new pattern
        self.model.setObjective(gb.quicksum(
            self.piecesInPatternVar[p] * self.duals[p]
            for p in self.piecesIndex),
            sense = gb.GRB.MAXIMIZE)
```

Para saber si el patrón generado en el subproblema mejora o no la solución en el problema maestro se debe cumplir que $c_j - \pi < 0$. Para obtener el nuevo patrón del subproblema basta con conocer las nuevas variables del problema creado:

```
    def getNewPattern(self):
        return self.model.getAttr("X", self.model.getVars())
```

Para añadir el nuevo patrón al problema maestro se crea un objeto columna y al añadir la nueva variable Gurobi permite reflejar cómo afecta la nueva variable a la función objetivo así como especificar cuál es la nueva columna, de esta manera al añadir la nueva variable no resuelve desde cero sino que parte de la solución óptima anterior (aquí también se van añadiendo los nuevos patrones al dataframe para obtener la solución al final):

```
    def addColumn(self, objective, newPattern):
        ctName = ('PatternUseVar[%s]' %len(self.model.getVars()))
```

```
        newColumn = gb.Column(newPattern, self.model.getConstrs())
        patternDf.loc[len(patternDf.index)] = [1, [int(abs(x))
                                                   for x in newPattern]]
        self.model.addVar(
            vtype = gb.GRB.INTEGER,
            lb = 0, obj = objective,
            column = newColumn,
            name = ctName)
        self.model.update()
```

Una vez ya no se obtiene mejor solución para el problema relajado, se resuelve el problema sin relajar:

```
    def solveModel(self, timelimit = None, GAP = 0.0):
        self.model.setParam('TimeLimit', timelimit)
        self.model.setParam('MIPGap', GAP)
        self.model.optimize()
```

Bibliografía

[1] Winston, W. L. *Introduction to Mathematical Programming: Applications and Algorithms.* Brooks/Cole, 1990.

[2] Wolsey, L. A. *Integer Programming.* Wiley-Interscience, 1998.