Trabajo Fin de Máster

Ramificación y acotación para la resolución de problemas de optimización polinómica

Alejandro Barros González

Máster en Técnicas Estadísticas

Curso 2025-2026

Propuesta de Trabajo Fin de Máster

Título en galego: Ramificación e acotación para a resolución de problemas de optimización polinómica
Título en español: Ramificación y acotación para la resolución de problemas de optimización polinómica
English title: Branch and bound for solving polynomial optimization problems
Modalidad: Modalidad B
Autor/a: Alejandro Barros González, Universidad de Vigo
Director/a: Brais González Rodríguez, Universidad de Vigo
Tutor/a: Julio González Díaz, CITMAga
Breve resumen del trabajo: Este Trabajo de Fin de Máster estudia la aplicación de técnicas de reformulación y linealización a problemas de optimización polinómica con variables enteras y binarias. El trabajo se desarrolla en el marco de RAPOSa, analizando el tratamiento de productos de variables binarias, comparando distintas formulaciones y evaluando computacionalmente su efecto sobre el rendimiento del solver.

Don Brais González Rodríguez, Profesor Distinguido de la Universidad de Vigo, y don Julio González Díaz, Investigador vinculado de CITMAga informan que el Trabajo Fin de Máster titulado

Ramificación y acotación para la resolución de problemas de optimización polinómica

fue realizado bajo su dirección por don Alejandro Barros González para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal. Además, don Brais González Rodríguez y don Alejandro Barros González

$\surd$ sí $\square$ no

autorizan a la publicación de la memoria en el repositorio de acceso público asociado al Máster en Técnicas Estadísticas.

En Vigo, a 3 de junio de 2026.

El director:
Don Brais González Rodríguez

GONZALEZ RODRIGUEZ BRAIS - 44661382M
Digitally signed by GONZALEZ RODRIGUEZ BRAIS - 44661382M
Date: 2026.06.03 16:58:37 +02'00'

El tutor:
Don Julio González Díaz

Firmado por GONZALEZ DIAZ JULIO - ***3267** el día 03/06/2026 con un certificado emitido por AC FNMT Usuarios

El autor:
Don Alejandro Barros González

BARROS GONZALEZ, ALEJANDRO (AUTENTICACIÓN)
Firmado digitalmente por BARROS GONZALEZ, ALEJANDRO (AUTENTICACIÓN)
Fecha: 2026.06.03 16:43:27 +02'00'

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,...sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen

Resumen en español

En este trabajo se estudia la aplicación de técnicas de reformulación y linealización a problemas de optimización polinómica con variables enteras y binarias, dentro del marco de RAPOSa, un solver desarrollado por investigadores de la Universidad de Santiago de Compostela y de la Universidad de Vigo. En primer lugar, se presentan los fundamentos de la técnica RLT, incluyendo la construcción de variables auxiliares, restricciones bound factor, criterios de ramificación y extensiones al caso entero. A continuación, se analiza la reformulación de productos de variables binarias a partir de patrones de linealización, siguiendo las ideas de Elloumi y Verchère (2023). Estas reformulaciones se incorporan posteriormente a problemas polinómicos generales, comparando el ajuste de las relajaciones continuas y enteras con el de la formulación RLT original.

La parte computacional del trabajo se centra en evaluar el efecto de estas reformulaciones dentro de RAPOSa. Para ello, se genera una nueva batería de problemas con distintos números de variables binarias y se comparan varias configuraciones del solver, atendiendo a métricas como el tiempo de resolución y el número de nodos explorados. Finalmente, se proponen posibles mejoras del método, incluyendo nuevas estrategias de ramificación, un tratamiento más específico de los productos que combinan variables binarias y no binarias, y una reducción adicional del número de restricciones bound factor.

English abstract

This work studies the application of reformulation and linearization techniques to polynomial optimization problems with integer and binary variables, within the framework of RAPOSa, a solver developed by researchers from the University of Santiago de Compostela and the University of Vigo. First, the fundamentals of the Reformulation-Linearization Technique (RLT) are presented, including the construction of auxiliary variables, bound-factor constraints, branching criteria, and extensions to the integer case. Next, the reformulation of products of binary variables is analysed through linearization patterns, following the ideas of Elloumi and Verchère (2023). These reformulations are then incorporated into general polynomial optimization problems, comparing the strength of the continuous and integer relaxations with that of the original RLT formulation.

The computational part of the work focuses on evaluating the effect of these reformulations within RAPOSa. To this end, a new test set of problems with different numbers of binary variables is generated, and several configurations of the solver are compared using metrics such as solving time and the number of explored nodes. Finally, possible improvements to the method are proposed, including new branching strategies, a more specific treatment of products involving binary and non-binary variables, and an additional reduction in the number of bound-factor constraints.

Introducción

En los últimos años se ha producido un avance notable en el diseño e implementación de algoritmos de optimización global para problemas de programación entera mixta no lineal, habitualmente conocidos como problemas MINLP (*Mixed Integer Nonlinear Programming*). Este progreso se ha visto reflejado tanto en el desarrollo de nuevos métodos teóricos como en la incorporación de estas técnicas a solvers de propósito general. Herramientas originalmente orientadas a la programación lineal o cuadrática entera mixta, como Gurobi o Xpress, han comenzado a incluir funcionalidades para tratar problemas MINLP, mientras que otros solvers especializados, como BARON, Couenne, SCIP, LINDO Global u Octeract, se han consolidado como referencias dentro del ámbito de la optimización global. Este crecimiento responde, por un lado, a la amplia aplicabilidad de los modelos MINLP en campos como la ingeniería, la logística, la energía o la planificación, y, por otro, a los avances recientes en capacidad computacional y en técnicas de relajación, ramificación y acotación.

Una característica común a muchos de estos métodos es que no trabajan directamente con el problema original, sino que construyen transformaciones o reformulaciones que permiten obtener relajaciones más sencillas de resolver. Estas relajaciones se integran posteriormente en esquemas de tipo *branch-and-bound*, donde proporcionan cotas inferiores y superiores que permiten descartar regiones del espacio de búsqueda y avanzar hacia la certificación de la optimalidad global. En este contexto, las reformulaciones desempeñan un papel esencial: una formulación adecuada puede mejorar la calidad de las cotas, reducir el tamaño del árbol de búsqueda y hacer que la resolución computacional del problema sea mucho más eficiente.

Dentro de esta familia de técnicas, una de las reformulaciones más relevantes es la *Reformulation-Linearization Technique* (RLT). Esta metodología, utilizada ya en el trabajo de Sherali y Tuncbilek (1992), permite construir un algoritmo de optimización global para problemas de optimización polinómica continua. La idea central consiste en introducir nuevas variables asociadas a productos de variables originales y reemplazar progresivamente las expresiones polinómicas por restricciones lineales adicionales. De este modo, el problema no lineal inicial se transforma en una relajación lineal que puede resolverse de forma eficiente y que, integrada dentro de un esquema de ramificación y acotación, permite aproximar el óptimo global con garantías.

A partir de esta idea inicial, distintas contribuciones posteriores han estudiado cómo adaptar y mejorar la técnica RLT para hacerla aplicable a clases de problemas más generales. En particular, una extensión natural consiste en considerar problemas de optimización polinómica con variables enteras, lo que permite conectar el marco continuo original con el ámbito más amplio de la programación no lineal entera mixta. Esta extensión introduce nuevas dificultades, ya que el algoritmo debe controlar simultáneamente las violaciones asociadas a las identidades RLT y las posibles violaciones de integralidad de las variables. Por ello, surgen distintas estrategias: resolver relajaciones continuas e incorporar la integralidad dentro del esquema de ramificación, o bien trabajar directamente con relajaciones enteras que mantengan parte de la estructura discreta del problema.

En esta línea se enmarca el desarrollo de RAPOSa, un solver de optimización global para problemas polinómicos desarrollado por investigadores de la Universidad de Santiago de Compostela y de la Universidad de Vigo. RAPOSa implementa un esquema basado en la técnica RLT y permite estudiar de forma práctica el comportamiento de distintas reformulaciones, relajaciones y estrategias de ramificación. El presente Trabajo de Fin de Máster se sitúa en esta misma línea y se centra en estudiar

cómo mejorar el tratamiento de las variables enteras y, especialmente, de las variables binarias, analizando distintas reformulaciones y estrategias de ramificación dentro de RAPOSa. En particular, el estudio de las reformulaciones binarias desarrollado en el Capítulo 2 se apoya de forma esencial en las ideas propuestas por Elloumi y Verchère (2023), que proporcionan un marco para comparar distintas reformulaciones de productos de variables binarias mediante patrones de linealización.

La memoria se estructura en cuatro capítulos.

En el Capítulo 1 se introducen los fundamentos teóricos sobre los que se apoya el resto del trabajo. En primer lugar, se presentan los conceptos básicos necesarios para formular problemas de optimización polinómica. A continuación, se describe la técnica de reformulación y linealización, conocida como *Reformulation-Linearization Technique* (RLT), siguiendo el algoritmo propuesto por Sherali y Tuncbilek (1992). Se construye la relajación lineal asociada al problema polinómico original mediante la introducción de variables RLT y de restricciones bound factor, y se estudian algunos resultados que justifican su validez, como la acotación de las variables auxiliares y el criterio de ramificación basado en la violación de las identidades RLT. Posteriormente, se comentan mejoras del procedimiento, en particular el uso de los denominados J-sets, que permiten reducir el número de restricciones generadas. La parte final del capítulo se dedica a la extensión del algoritmo al caso de problemas con variables enteras, distinguiendo entre dos enfoques: resolver la relajación continua y gestionar explícitamente las violaciones de integralidad, o bien resolver directamente una relajación entera. Finalmente, se presenta RAPOSa como implementación computacional del algoritmo basado en RLT.

En el Capítulo 2 se estudia la reformulación de productos de binarias, siguiendo de cerca las ideas propuestas por Elloumi y Verchère (2023). El capítulo comienza revisando la linealización estándar de productos de variables binarias y mostrando cómo un problema polinómico puramente binario puede reformularse como un problema lineal entero mixto. A partir de ahí se introducen los patrones de linealización, que permiten representar distintas formas de descomponer y sustituir productos de binarias mediante variables auxiliares. Después se recogen resultados sobre el ajuste de estas formulaciones, en particular la dominancia de ciertos patrones de linealización frente a la linealización estándar. En la segunda parte del capítulo, estas ideas se extienden a problemas de optimización polinómica generales, en los que pueden aparecer simultáneamente variables binarias y variables no binarias. Para ello, se reformula únicamente la parte binaria de los monomios y posteriormente se aplica RLT al problema resultante. Este capítulo concluye comparando el ajuste de las relajaciones continuas y enteras, y mostrando que la formulación base de RLT puede ser más ajustada que las formulaciones obtenidas tras aplicar patrones de linealización, aunque estas últimas puedan resultar computacionalmente más ventajosas.

En el Capítulo 3 se presentan los resultados computacionales. En primer lugar, se justifica la necesidad de construir una nueva batería de problemas, ya que muchas de las baterías habituales están formadas principalmente por problemas cuadráticos, en los que las distintas linealizaciones de binarias pueden coincidir con la formulación base de RAPOSa y, por tanto, no permiten evaluar adecuadamente el efecto de las reformulaciones propuestas. A continuación, se describe el procedimiento de generación de instancias, detallando los parámetros con los que funciona. Posteriormente, se detalla el entorno de pruebas y las métricas utilizadas para comparar las distintas configuraciones, como los tiempos de resolución o el número de nodos explorados. La parte central del capítulo analiza varias baterías de problemas con distintos números de variables binarias, comparando la formulación base con distintas reformulaciones de binarias y diferentes enfoques de ramificación. Finalmente, se validan algunas de las conclusiones mediante representaciones gráficas de los árboles de ramificación, lo que permite interpretar visualmente cómo evolucionan las cotas y qué tipo de variables se seleccionan para realizar la ramificación en cada estrategia.

En el Capítulo 4 se recogen posibles líneas de mejora del algoritmo basado en la técnica RLT a partir de las observaciones teóricas y computacionales realizadas en los capítulos anteriores. En primer lugar, se estudia una posible linealización de la parte no binaria por la parte binaria, con el objetivo de aprovechar mejor la estructura de los monomios que combinan variables continuas y binarias. Para dicha linealización, veremos que es posible probar resultados análogos a los de la segunda sección del Capítulo 2. Después se plantean nuevas estrategias de ramificación, en particular, la posibilidad de no

priorizar de forma rígida un tipo concreto de violación y de controlar la profundidad de la ramificación entera. También se analiza con más detalle el papel de las variables auxiliares que sustituyen a productos de binarias, prestando atención a su integralidad en las soluciones factibles del problema original, a su posible uso en técnicas de ajuste de cotas y a estrategias de ramificación múltiple que aprovechen la estructura de los patrones de linealización. Por último, se propone una reducción adicional del número de restricciones bound factor más allá de los J-sets, mediante la construcción de lo que se denominan súper J-sets. Esta idea consiste en agrupar monomios mediante mínimos comunes múltiplos para cubrir varios conjuntos de restricciones con un número menor de restricciones bound factor, formulando el problema como un modelo entero de cobertura y proponiendo también una heurística para hacerlo computacionalmente más manejable.

Capítulo 1

Programación polinómica y *Reformulation Linearization Technique*

En este capítulo empezaremos presentando conceptos básicos necesarios para describir los problemas de programación polinómica. A continuación, presentaremos un algoritmo para su resolución basado en la técnica de reformulación y linealización (*RLT-based algorithm*) presentado en Sherali y Tuncbilek (1992). Este algoritmo tiene la ventaja de garantizar la obtención de un óptimo global, aspecto especialmente relevante si consideramos que los problemas de programación polinómica suelen ser no convexos y, por tanto, los métodos de optimización local no aseguran una solución óptima en general.

Para el desarrollo teórico nos guiaremos por González-Rodríguez (2022), donde se detallan los resultados clave que sustentan la validez del algoritmo. Finalmente, presentaremos RAPOSa, un solver implementado en C++ basado en dicho algoritmo. Mencionaremos algunas ideas que se abordan en González-Rodríguez et al. (2022) sobre la implementación, y dedicaremos el final del capítulo a la extensión del algoritmo a variables enteras, apoyándonos en lo recogido en González-Díaz et al. (2024).

1.1. Preliminares

Con el fin de definir adecuadamente los problemas de programación polinómica, comenzamos presentando brevemente algunos conceptos básicos sobre polinomios.

Definición 1.1. Dado un vector $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$, un monomio es cualquier expresión de la forma $m(\boldsymbol{x}) = \alpha \prod_{j=1}^{n} x_j^{\beta_j}$, con $\alpha \in \mathbb{R}$ y $\beta_j \in \mathbb{N} \cup \{0\}$ para cada $j \in \{1, \ldots, n\}$.

Definición 1.2. El grado de un monomio $m(\boldsymbol{x}) = \alpha \prod_{j=1}^{n} x_j^{\beta_j}$ se define como $\sum_{j=1}^{n} \beta_j$.

Definición 1.3. Dado un vector $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$, un polinomio es la suma de un número finito de monomios; es decir, cualquier expresión de la forma: $\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\beta_{tj}}$, donde $T \subset \mathbb{N}$ es un conjunto de índices, $\alpha_t \in \mathbb{R}$ para todo $t \in T$ y $\beta_{tj} \in \mathbb{N} \cup \{0\}$ para todo $j \in \{1, \ldots, n\}$ y para todo $t \in T$.

Definición 1.4. El grado de un polinomio $\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\beta_{tj}}$ se define como el mayor de los grados de los monomios que lo conforman; es decir, $\max_{t \in T} \{\sum_{j=1}^{n} \beta_{tj}\}$.

A continuación, introducimos el concepto de multiconjunto, que permitirá representar los monomios mediante los índices de las variables que los componen y sus multiplicidades.

Definición 1.5. Dado un conjunto finito N y una aplicación $p : N \to \mathbb{N}$, se denota por multiconjunto al par (N, p), donde la aplicación p denota la multiplicidad de cada elemento. En este caso, se denota su cardinal por $|(N, p)| = \sum_{j \in N} p(j)$.

Dado que el uso de los multiconjuntos en este trabajo será muy limitado, realizaremos un ligero abuso de notación: si $N = \{1, \dots, n\}$, denotaremos el multiconjunto (N, p) como $\{1, \overset{p(1)}{\dots}, 1, \dots, n, \overset{p(n)}{\dots}, n\}$. Además, cuando $p(j) = \delta$ para todo $j \in N$ con $\delta \in \mathbb{N}$, escribiremos simplemente N^δ.

Empleando multiconjuntos, un monomio $m(\boldsymbol{x}) = \alpha \prod_{j=1}^{n} x_j^{\beta_j}$ de grado δ puede expresarse como $m(\boldsymbol{x}) = \alpha \prod_{j \in J} x_j$, donde $J \subset N^\delta$ es el multiconjunto dado por $J = \{1, \overset{\beta_1}{\dots}, 1, \dots, n, \overset{\beta_n}{\dots}, n\}$, entendiendo que $\beta_j = 0$ quiere decir que el elemento j no aparece en J. En adelante, emplearemos el multiconjunto J para referirnos al conjunto de índices con multiplicidad que caracteriza a cada monomio. Análogamente, un polinomio $\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\beta_{tj}}$ de grado δ puede escribirse como $\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j \in J_t} x_j$ donde $J_t \subset N^\delta$ para cada $t \in T$.

Con esto ya estamos en disposición de definir los problemas de optimización polinómica, que constituirán el objeto de estudio del presente trabajo.

Definición 1.6. Un problema de optimización polinómica es un problema de optimización de la forma:

$$\begin{array}{rll} \text{minimizar} & \phi_0(\boldsymbol{x}) & \\ \text{sujeto a} & \phi_r(\boldsymbol{x}) \geq \beta_r, & r = 1, \dots, R_1, \\ & \phi_r(\boldsymbol{x}) = \beta_r, & r = R_1 + 1, \dots, R, \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^n, & \end{array} \tag{$PP(\Omega)$}$$

donde cada $\phi_r(\boldsymbol{x})$ es un polinomio de grado $\delta_r \in \mathbb{N}$, $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : 0 \leq l_j \leq x_j \leq u_j < \infty,\ \forall j \in N\} \subset \mathbb{R}^n$ es un hiperrectángulo que contiene a la región factible y el grado del problema se define como $\delta = \max_{r \in \{0, \dots, R\}} \delta_r$. A partir de ahora se supondrá que $\delta \geq 2$ para que no sea un problema de optimización lineal.

Ejemplo 1.7. Presentamos un ejemplo con el fin de ilustrar los conceptos a medida que se vayan presentando. Un problema de optimización polinómica podría ser:

$$\begin{array}{rl} \text{minimizar} & x_1 x_2^2 - x_1^2 x_2 + 2x_1 \\ \text{sujeto a} & 3x_1 x_2 - x_2 \geq 2 \\ & x_1 - x_2 + x_1 x_2 = 4 \\ & \boldsymbol{x} \in [1, 2] \times [3, 4]. \end{array}$$

Antes de introducir la técnica RLT, mencionamos algunos conceptos de optimización que emplearemos a lo largo del trabajo.

Definición 1.8. Dados dos problemas de optimización $PP_1(\Omega_1)$ y $PP_2(\Omega_2)$, se dice que son *equivalentes* cuando existe una correspondencia entre los conjuntos de soluciones factibles y las funciones objetivo asociadas.

Definición 1.9. Dadas dos relajaciones de formulaciones equivalentes, se dice que una es al menos tan ajustada como la otra si la solución óptima de la primera relajación es mayor o igual que la solución óptima de la segunda.

Adicionalmente, conviene aclarar que, cuando hablemos de relajaciones enteras, nos referiremos a reformulaciones del problema en las que las variables enteras mantienen sus restricciones de integralidad. En cambio, cuando dichas variables puedan tomar cualquier valor dentro de su intervalo de cotas, hablaremos de relajación continua.

Una vez introducida la formulación general de los problemas de programación polinómica, el siguiente paso consiste en estudiar cómo pueden abordarse de forma algorítmica. La principal dificultad de

estos problemas reside en la presencia de productos entre variables, que impiden tratarlos directamente mediante técnicas de programación lineal. La técnica RLT parte precisamente de esta observación: en lugar de trabajar con los productos originales, se introducen nuevas variables que los representan y se añaden restricciones lineales que relacionan estas nuevas variables con las variables originales.

De este modo, el problema polinómico se sustituye por una relajación lineal en un espacio de mayor dimensión. Aunque esta reformulación puede aumentar considerablemente el tamaño del problema, permite aprovechar algoritmos de programación lineal y proporciona una estructura adecuada para diseñar un procedimiento de ramificación y acotación con garantías de convergencia global.

1.2. *Reformulation-Linearization Technique* (RLT)

Con esta motivación, presentamos el algoritmo de Sherali y Tuncbilek (1992). Para ello, introducimos primero la relajación lineal asociada al problema polinómico y las variables auxiliares que permiten formularla.

Definición 1.10. Dado un problema de optimización polinómica $PP(\Omega)$, se definen los *bounding factors* como las restricciones dadas por $(x_j - l_j) \geq 0$ y $(u_j - x_j) \geq 0$ para cada $j \in N$.

Definición 1.11. Dado un problema de optimización polinómica $PP(\Omega)$ y un monomio $J \subset N^\delta$ con $2 \leq |J| \leq \delta$, se define la correspondiente identidad RLT como $X_J = \prod_{j \in J} x_j$, donde X_J se denomina variable RLT. Puesto que no afecta a los resultados, será habitual el abuso de notación $x_j = X_{\{j\}}$.

Definición 1.12. Dado un punto $\boldsymbol{x}$ y un problema de optimización polinómica $PP(\Omega)$ de grado δ, se define la extensión de $\boldsymbol{x}$ como $E(\boldsymbol{x}) = (X_J)_{J \subset N^\delta}$, donde $X_J = \prod_{j \in J} x_j$ para cada $J \subset N^\delta$.

Definición 1.13. Dado un polinomio $\phi(\boldsymbol{x})$, se define su linealización $[\phi(\boldsymbol{x})]_L$ como el resultado de reemplazar cada monomio de grado mayor que uno en $\phi(\boldsymbol{x})$ por su variable RLT correspondiente.

Con las anteriores definiciones, podemos formular la relajación lineal asociada al problema $PP(\Omega)$ propuesta en Sherali y Tuncbilek (1992):

$$
\begin{array}{rll}
\text{minimizar} & [\phi_0(\boldsymbol{x})]_L & \\
\text{sujeto a} & [\phi_r(\boldsymbol{x})]_L \geq \beta_r, & r = 1, \ldots, R_1, \\
& [\phi_r(\boldsymbol{x})]_L = \beta_r, & r = R_1 + 1, \ldots, R, \\
& \left[\displaystyle\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \right]_L \geq 0, & \forall J_1 \cup J_2 \subset N^\delta, \\
& & |J_1 \cup J_2| = \delta \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^n. &
\end{array}
\qquad (RLT(PP(\Omega)))
$$

Ejemplo 1.14. Sobre nuestro ejemplo, la reformulación del problema de linealización será de la forma:

$$
\begin{array}{rl}
\text{minimizar} & X_{\{1,2,2\}} - X_{\{1,1,2\}} + 2x_1 \\
\text{sujeto a} & 3X_{\{1,2\}} - x_2 \geq 2 \\
& x_1 - x_2 + X_{\{1,2\}} = 4 \\
& \boldsymbol{x} \in [1,2] \times [3,4],
\end{array}
$$

a la que adicionalmente habrá que añadir las siguientes restricciones bound factor linealizadas:

$$\begin{cases}
-X_{\{1,1,1\}} + 6X_{\{1,1\}} - 12x_1 + 8 \geq 0, \\
X_{\{1,1,1\}} - 5X_{\{1,1\}} + 8x_1 - 4 \geq 0, \\
-X_{\{1,1,1\}} + 4X_{\{1,1\}} - 5x_1 + 2 \geq 0, \\
X_{\{1,1,1\}} - 3X_{\{1,1\}} + 3x_1 - 1 \geq 0, \\
\\
-X_{\{1,1,2\}} + 4X_{\{1,1\}} + 4X_{\{1,2\}} - 16x_1 - 4x_2 + 16 \geq 0, \\
X_{\{1,1,2\}} - 3X_{\{1,1\}} - 4X_{\{1,2\}} + 12x_1 + 4x_2 - 12 \geq 0, \\
X_{\{1,1,2\}} - 4X_{\{1,1\}} - 3X_{\{1,2\}} + 12x_1 + 2x_2 - 8 \geq 0, \\
-X_{\{1,1,2\}} + 3X_{\{1,1\}} + 3X_{\{1,2\}} - 9x_1 - 2x_2 + 6 \geq 0, \\
-X_{\{1,1,2\}} + 4X_{\{1,1\}} + 2X_{\{1,2\}} - 8x_1 - x_2 + 4 \geq 0, \\
X_{\{1,1,2\}} - 3X_{\{1,1\}} - 2X_{\{1,2\}} + 6x_1 + x_2 - 3 \geq 0, \\
\\
-X_{\{1,2,2\}} + 8X_{\{1,2\}} + 2X_{\{2,2\}} - 16x_1 - 16x_2 + 32 \geq 0, \\
X_{\{1,2,2\}} - 7X_{\{1,2\}} - 2X_{\{2,2\}} + 12x_1 + 14x_2 - 24 \geq 0, \\
-X_{\{1,2,2\}} + 6X_{\{1,2\}} + 2X_{\{2,2\}} - 9x_1 - 12x_2 + 18 \geq 0, \\
X_{\{1,2,2\}} - 8X_{\{1,2\}} - X_{\{2,2\}} + 16x_1 + 8x_2 - 16 \geq 0, \\
-X_{\{1,2,2\}} + 7X_{\{1,2\}} + X_{\{2,2\}} - 12x_1 - 7x_2 + 12 \geq 0, \\
X_{\{1,2,2\}} - 6X_{\{1,2\}} - X_{\{2,2\}} + 9x_1 + 6x_2 - 9 \geq 0, \\
\\
-X_{\{2,2,2\}} + 12X_{\{2,2\}} - 48x_2 + 64 \geq 0, \\
X_{\{2,2,2\}} - 11X_{\{2,2\}} + 40x_2 - 48 \geq 0, \\
-X_{\{2,2,2\}} + 10X_{\{2,2\}} - 33x_2 + 36 \geq 0, \\
X_{\{2,2,2\}} - 9X_{\{2,2\}} + 27x_2 - 27 \geq 0.
\end{cases}$$

Este ejemplo evidencia una de las principales consecuencias de aplicar el algoritmo basado en la técnica RLT: aunque el problema resultante es lineal, el número de variables y restricciones puede crecer rápidamente. Por ello, además de estudiar la validez teórica de la reformulación, será importante considerar mecanismos que reduzcan su tamaño o que permitan resolverla de forma más eficiente.

Nótese que, antes de aplicar la linealización, el producto de δ *bounding factors*:

$$F_\delta(J_1, J_2) = \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \geq 0, \ \forall J_1 \cup J_2 \subset N^\delta, \ |J_1 \cup J_2| = \delta, \tag{1.1}$$

es una restricción que se deriva directamente del hecho de que $\boldsymbol{x}$ pertenece a Ω. Cabe destacar, además, que dicha restricción no es estrictamente necesaria para garantizar la convergencia del algoritmo, tal y como se apunta en Sherali y Tuncbilek (1992).

Si lo que nos interesa es que las variables creadas estén acotadas (que es algo que probaremos que se deriva de dichas restricciones) bastaría con extender las cotas impuestas por Ω al producto de variables. Por ejemplo, supongamos $l_1 \leq x_1 \leq u_1$ y $l_2 \leq x_2 \leq u_2$ y que buscamos acotar la variable RLT $X_{\{1,2\}}$, bastaría imponer $l_1 l_2 \leq X_{\{1,2\}} \leq u_1 u_2$ para hacerlo.

No obstante, el uso de las restricciones (1.1) introduce interrelaciones adicionales entre las variables y los productos linealizados, lo que permite obtener una relajación lineal más ajustada. A este conjunto de restricciones lo denotaremos por bound factors. Posteriormente se justificará por qué se considera únicamente el producto de exactamente δ bounding factors y no de un número menor.

Por otra parte, ya que es posible probar que el número de restricciones que componen las restricciones bound factor son

$$\binom{2n + \delta - 1}{\delta},$$

es evidente que dicho número crece de forma exponencial al aumentar el número de variables o el grado del problema, como bien refleja el ejemplo. Sin embargo, se puede demostrar que es posible considerar solo un subconjunto de las mismas para reducir su número, como se verá más adelante.

Con el fin de establecer la convergencia del algoritmo RLT a un óptimo global del problema original, presentamos tres resultados preliminares. En primer lugar, ponemos de manifiesto que el problema $RLT(PP(\Omega))$ es una relajación lineal de $PP(\Omega)$.

Lema 1.15. *Sean $v[PP(\Omega)]$ y $v[RLT(PP(\Omega))]$ los óptimos de $PP(\Omega)$ y $RLT(PP(\Omega))$ respectivamente. Entonces, se tiene que $v[RLT(PP(\Omega))] \leq v[PP(\Omega)]$. Además, si la solución óptima de $RLT(PP(\Omega))$ es de la forma $(\bar{\boldsymbol{x}}, E(\bar{\boldsymbol{x}}))$, entonces $\bar{\boldsymbol{x}}$ es la solución óptima de $PP(\Omega)$.*

Demostración. Sea $\phi_0[PP(\Omega)](\boldsymbol{x})$ la función objetivo de $PP(\Omega)$ y $\phi_0[RLT(PP(\Omega))](\boldsymbol{x}, \boldsymbol{X})$ la función objetivo de $RLT(PP(\Omega))$. Sea $\bar{\boldsymbol{x}}$ una solución factible del problema original, entonces $(\bar{\boldsymbol{x}}, E(\bar{\boldsymbol{x}}))$ es una solución factible de la linealización, con lo cual la región factible de $PP(\Omega)$ está contenida en la de $RLT(PP(\Omega))$. Además, se tiene que $\phi_0[PP(\Omega)](\bar{\boldsymbol{x}}) = \phi_0[RLT(PP(\Omega))](\bar{\boldsymbol{x}}, E(\bar{\boldsymbol{x}}))$, luego $v[RLT(PP(\Omega))] \leq v[PP(\Omega)]$.

Por otra parte, en caso de que la solución de $RLT(PP(\Omega))$ sea de la forma $(\bar{\boldsymbol{x}}, E(\bar{\boldsymbol{x}}))$, entonces $\bar{\boldsymbol{x}}$ es una solución factible de $PP(\Omega)$, y como $\phi_0[PP(\Omega)](\bar{\boldsymbol{x}}) = \phi_0[RLT(PP(\Omega))](\bar{\boldsymbol{x}}, E(\bar{\boldsymbol{x}}))$, entonces $\bar{\boldsymbol{x}}$ ha de ser la solución óptima de $PP(\Omega)$. □

Una vez justificada la conveniencia de emplear las restricciones bound factor surgía la cuestión de por qué tomar δ bounding factors y no menos. En el siguiente lema se prueba que los productos que involucran menos bounding factors se derivan del producto de δ de ellos, lo que permite reducir el número de restricciones construidas sin afectar a lo ajustada que es la formulación.

Lema 1.16. *Supongamos $l_j < u_j$ para todo $j \in J_1 \cup J_2$, con $J_1 \cup J_2 \subset N^\delta$. Entonces, todas las restricciones de la forma $[F_{\delta'}(J_1, J_2)]_L \geq 0$ con $J_1 \cup J_2 \subset N^\delta$, $|J_1 \cup J_2| = \delta'$ y $1 \leq \delta' < \delta$ están implicadas por las restricciones $[F_\delta(J_1, J_2)]_L \geq 0$ con $J_1 \cup J_2 \subset N^\delta$, $|J_1 \cup J_2| = \delta$.*

Demostración. Para cualquier δ', con $1 \leq \delta' < \delta$, $j \in N$ y $J_1 \cup J_2 \subset N^\delta$, $|J_1 \cup J_2| = \delta$, consideremos las restricciones de la forma $[F_{\delta'+1}(J_1 \cup \{j\}, J_2)]_L \geq 0$ y $[F_{\delta'+1}(J_1, J_2 \cup \{j\})]_L \geq 0$. Entonces, se verifica que:

$$\begin{aligned}[F_{\delta'+1}(J_1 \cup \{j\}, J_2)]_L + [F_{\delta'+1}(J_1, J_2 \cup \{j\})]_L &= [(x_j - l_j)F_{\delta'}(J_1, J_2)]_L + [(u_j - x_j)F_{\delta'}(J_1, J_2)] \\ &= [x_j F_{\delta'}(J_1, J_2)]_L - l_j[F_{\delta'}(J_1, J_2)]_L \\ &+ u_j[F_{\delta'}(J_1, J_2)]_L - [x_j F_{\delta'}(J_1, J_2)]_L \\ &= (u_j - l_j)[F_{\delta'}(J_1, J_2)]_L.\end{aligned}$$

Puesto que $u_j - l_j > 0$, se tiene que $[F_{\delta'}(J_1, J_2)]_L \geq 0$ está implicado por las otras 2 restricciones. Por tanto, aplicando inducción queda probado el resultado. □

Observación 1.17. *Nótese que el anterior resultado no es cierto cuando $l_j = u_j$. En efecto:*

$$[F_{\delta'+1}(J_1 \cup \{j\}, J_2)]_L = [(x_j - l_j)F_{\delta'}(J_1, J_2)]_L = -[(u_j - x_j)F_{\delta'}(J_1, J_2)]_L = -[F_{\delta'+1}(J_1, J_2 \cup \{j\})]_L.$$

De esta forma, al sumar ambas restricciones, en lugar de tener $k[F_{\delta'}(J_1, J_2)]_L \geq 0$, con $k > 0$, origina una restricción de la forma $0 \geq 0$.

La anterior consideración no se tiene en cuenta en González-Rodríguez (2022) ni Sherali y Tuncbilek (1992), donde se afirma el resultado del Lema 1.16 para $u_j - l_j \geq 0$. En general esto no tiene impacto en los problemas continuos, que son los abordados en Sherali y Tuncbilek (1992), ya que la situación $u_j - l_j = 0$ sólo puede producirse cuando se parte de una variable con valor fijado, caso poco habitual en el que además podría eliminarse dicha variable del problema y reemplazarse por su valor, o bien cuando se ramifica en una variable por una de sus cotas, algo que, como veremos en el Lema 1.19, no tiene sentido hacer.

A partir del Lema 1.16 podemos derivar el siguiente resultado sobre la acotación de las variables de la relajación lineal del problema.

Corolario 1.18. *En el problema $RLT(PP(\Omega))$ todas las variables RLT están acotadas inferior y superiormente debido a las restricciones bound factor. Con lo cual, todas las variables de este problema están acotadas y entonces, de ser un problema factible, tiene un óptimo finito.*

Demostración. Consideremos X_J una variable con $J \subset N^\delta$ y probemos que X_J está acotada inferior y superiormente. Lo haremos por inducción.

Supongamos $|J| = 2$, donde $J = \{j_1, j_2\}$. En virtud del Lema 1.16, las restricciones bound factor $[F_2(J_1, J_2)]_L \geq 0$ con $J_1 = J$ y $J_2 = \emptyset$ están implicadas por las que aparecen en el problema $RLT(PP(\Omega))$. Por lo tanto, se verifica que:

$$[F_2(J_1, J_2)]_L = [(x_{j_1} - l_{j_1})(x_{j_2} - l_{j_2})]_L = X_J - l_{j_2}x_{j_1} - l_{j_1}x_{j_2} + l_{j_1}l_{j_2} \geq 0,$$

con lo cual $X_J \geq l_{j_2}x_{j_1} + l_{j_1}x_{j_2} - l_{j_1}l_{j_2}$ y está acotada inferiormente.

Si consideramos una bound factor de la misma forma $[F_2(J_1, J_2)]_L \geq 0$, tomando ahora $J_1 = \{j_1\}$ y $J_2 = \{j_2\}$, que también se deducen de las presentes en el problema $RLT(PP(\Omega))$, se tiene:

$$[F_2(J_1, J_2)]_L = [(x_{j_1} - l_{j_1})(u_{j_2} - x_{j_2})]_L = -X_J + u_{j_2}x_{j_1} + l_{j_1}x_{j_2} - l_{j_1}u_{j_2} \geq 0,$$

con lo que $X_J \leq u_{j_2}x_{j_1} + l_{j_1}x_{j_2} - l_{j_1}u_{j_2}$, así que está acotada superiormente.

Probado el caso base, consideremos que el resultado se verifica para $|J| < \delta'$ y vamos a probarlo para $|J| = \delta'$, con $\delta' \leq \delta$. Considerando $J = \{j_1, \ldots, j_{\delta'}\}$, ya sabemos que $[F_{\delta'}(J_1, J_2)]_L \geq 0$ con $J_1 = J$ y $J_2 = \emptyset$ está implicado por $RLT(PP(\Omega))$. Así, se verifica que:

$$[F_{\delta'}(J_1, J_2)]_L = \left[\prod_{j \in J}(x_j - l_j)\right]_L = X_J + f(\boldsymbol{x}, \boldsymbol{X}) \geq 0,$$

donde $f(\boldsymbol{x}, \boldsymbol{X})$ es una combinación lineal de variables RLT asociadas a monomios de grado menor que δ'. Por la hipótesis de inducción podemos afirmar que están acotadas y así se tiene que $X_J \geq -f(\boldsymbol{x}, \boldsymbol{X})$; por tanto, X_J está acotada inferiormente.

Tomando ahora $J_1 = J \setminus \{j_{\delta'}\}$ y $J_2 = \{j_{\delta'}\}$, de nuevo es implicada por las restricciones de $RLT(PP(\Omega))$, y tenemos que:

$$[F_{\delta'}(J_1, J_2)]_L = \left[(u_{j_{\delta'}} - x_{j_{\delta'}}) \prod_{j \in J_1}(x_j - l_j)\right]_L = -X_J + f(\boldsymbol{x}, \boldsymbol{X}) \geq 0,$$

siendo nuevamente $f(\boldsymbol{x}, \boldsymbol{X})$ una combinación de variables RLT asociadas a monomios de grado menor que δ', las cuales están acotadas. Por lo tanto, tenemos $X_J \leq f(\boldsymbol{x}, \boldsymbol{X})$ y así la variable también está acotada superiormente. □

El anterior resultado permite dejar las variables RLT no acotadas, lo que puede resultar beneficioso para el solver que resuelve la relajación entera del problema en presencia de cotas grandes. En el Capítulo 4 presentaremos una expresión exacta para esas cotas. A continuación introducimos un resultado fundamental para la elección del criterio de ramificación.

Lema 1.19. *Sea $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ una solución factible de $RLT(PP(\Omega))$. Si $\bar{x}_p = l_p$ para algún $p \in N$, entonces $\bar{X}_{J\cup\{p\}} = l_p\bar{X}_J$ para cada $J \subset N^\delta$ con $1 \leq |J| \leq \delta - 1$. De igual manera, si $\bar{x}_p = u_p$ para algún $p \in N$, entonces $\bar{X}_{J\cup\{p\}} = u_p\bar{X}_J$ para cada $J \subset N^\delta$ con $1 \leq |J| \leq \delta - 1$.*

Demostración. Nuevamente vamos a probarlo por inducción sobre $|J|$. Partamos del caso base, $J = \{q\}$, con $q \in N$. En virtud del Lema 1.16, de las restricciones bound factor de $RLT(PP(\Omega))$ se tiene que:

$$[(x_p - l_p)(x_q - l_q)]_L = X_{\{q,p\}} - l_q x_p - l_p x_q + l_p l_q \geq 0,$$
$$[(x_p - l_p)(u_q - x_q)]_L = -X_{\{q,p\}} + u_q x_p + l_p x_q - l_p u_q \geq 0.$$

Con lo cual, se deduce que $l_q(x_p - l_p) + l_p x_q \leq X_{\{q,p\}} \leq l_p x_q + u_q(x_p - l_p)$, y evaluando en $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$, si asumimos que $\bar{x}_p = l_p$, se verifica que $\bar{X}_{\{q,p\}} = l_p\bar{x}_q$.

Supongamos ahora que se verifica $\bar{X}_{J\cup\{p\}} = l_p \bar{X}_J$ para $|J| \in \{1,\dots,t-1\}$ y consideremos $J \subset N^\delta$ con $|J| = t$, con $2 \le t \le \delta - 1$. Para cada $q \in J$, en virtud nuevamente del Lema 1.16, se tiene que:

$$\left[(x_p - l_p)(x_q - l_q) \prod_{j\in J\setminus\{q\}} (x_j - l_j)\right]_L \ge 0,$$
$$\left[(x_p - l_p)(u_q - x_q) \prod_{j\in J\setminus\{q\}} (x_j - l_j)\right]_L \ge 0.$$

Si expresamos $\prod_{j\in J\setminus\{q\}}(x_j - l_j) = \prod_{j\in J\setminus\{q\}} x_j + \phi(\boldsymbol{x})$, donde $\phi(\boldsymbol{x})$ es un polinomio de grado menor o igual que $t-2$, tenemos que:

$$X_{J\cup\{p\}} - l_p X_J \ge l_q(X_{J\cup\{p\}\setminus\{q\}} - l_p X_{J\setminus\{q\}}) + [l_p x_q \phi(\boldsymbol{x}) - x_p x_q \phi(\boldsymbol{x})]_L + l_q[x_p\phi(\boldsymbol{x}) - l_p\phi(\boldsymbol{x})]_L,$$
$$X_{J\cup\{p\}} - l_p X_J \le u_q(X_{J\cup\{p\}\setminus\{q\}} - l_p X_{J\setminus\{q\}}) + [l_p x_q \phi(\boldsymbol{x}) - x_p x_q \phi(\boldsymbol{x})]_L + u_q[x_p\phi(\boldsymbol{x}) - l_p\phi(\boldsymbol{x})]_L.$$

Por la hipótesis de inducción sabemos que $\bar{X}_{J\cup\{p\}\setminus\{q\}} = l_p \bar{X}_{J\setminus\{q\}}$, por lo que evaluando las anteriores expresiones en $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ y teniendo en cuenta que $\bar{x}_p = l_p$, ambos lados derechos de las desigualdades se anulan, probando así la igualdad.

La demostración para el caso $\bar{x}_p = u_p$ es análoga. □

Los resultados anteriores justifican los elementos básicos sobre los que se construye el algoritmo basado en la técnica RLT. Por una parte, la relajación lineal proporciona cotas válidas para el problema original. Por otro lado, las restricciones bound factor permiten controlar las variables introducidas durante la linealización y garantizan que, cuando las identidades RLT se satisfacen, la solución de la relajación puede interpretarse como una solución factible del problema polinómico original.

A partir de esta idea, el algoritmo procede resolviendo sucesivamente relajaciones lineales sobre regiones cada vez más pequeñas del dominio. Cuando la solución obtenida no satisface las identidades RLT, se selecciona una variable sobre la que ramificar, dividiendo el dominio en dos subproblemas. Así, el proceso combina la obtención de cotas mediante relajaciones lineales con una estrategia de partición del dominio que fuerza progresivamente la consistencia entre las variables originales y las variables RLT.

Apoyándose entonces en dichos resultados, Sherali y Tuncbilek (1992) formulan el algoritmo RLT como un procedimiento de ramificación y acotación que consiste en resolver las relajaciones lineales del problema polinómico y, tras resolver cada nodo, seleccionar la variable x_p que maximiza una determinada función de las violaciones de las identidades RLT. En particular, la demostración se realiza empleando el siguiente criterio de ramificación:

$$p \in \arg\max_{j\in N}\{\theta_j\}, \text{ donde}$$
$$\theta_j = \max_{\substack{J\subset N^\delta \\ 1\le|J|\le\delta-1}} \left\{|\bar{X}_{J\cup\{j\}} - \bar{x}_j \bar{X}_J|\right\} \text{ para cada } j \in N, \tag{1.2}$$

donde $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ es la solución óptima del nodo correspondiente. Con esta elección, la ramificación se realiza añadiendo la cota $x_p \le \bar{x}_p$ en uno de los nodos hijos y $x_p \ge \bar{x}_p$ en el otro, actualizando además las restricciones bound factor en las que interviene x_p.

Procedemos a describir el algoritmo RLT para resolver el problema $PP(\Omega)$.

Inicialización.

- Inicializamos la mejor solución como $\boldsymbol{x}^* = \emptyset$ y el mejor valor objetivo como $v^* = \infty$. Fijamos $k = 1$, $t = 1$, $tot = 1$, $Q_1 = \{1\}$ y $\Omega^{1,1} = \Omega$.
- Resolvemos LP($\Omega^{1,1}$) y obtenemos una solución óptima $(\bar{\boldsymbol{x}}^{1,1}, \bar{\boldsymbol{X}}^{1,1})$ de LP($\Omega^{1,1}$). Si LP($\Omega^{1,1}$) es infactible, entonces el problema original $PP(\Omega)$ también lo es y el algoritmo termina.

- Seleccionamos la variable de ramificación x_p empleando la regla de ramificación (1.2). En caso de que $\theta_p^{1,1} = 0$, entonces $\bar{\boldsymbol{x}}^{1,1}$ es factible en $PP(\Omega)$, luego es una solución óptima de $PP(\Omega)$ y el algoritmo termina. En caso contrario pasamos a la Etapa 1.

Etapa 1 (Ramificación). Supongamos que hemos resuelto el problema relajado $\mathrm{LP}(\Omega^{k,t})$ y sea x_p la variable de ramificación seleccionada, con $\theta_p^{k,t} > 0$. Puesto que $\theta_p^{k,t} > 0$, por el Lema 1.19 sabemos que $l_p^{k,t} < \bar{x}_p^{k,t} < u_p^{k,t}$, donde $l_p^{k,t}$ y $u_p^{k,t}$ son las cotas inferior y superior, respectivamente, de x_p en $\mathrm{LP}(\Omega^{k,t})$. Definimos los siguientes subconjuntos de $\Omega^{k,t}$:

$$\begin{aligned} \Omega^{k,t_1} &= \Omega^{k,t} \cap \{\boldsymbol{x} \in \mathbb{R}^n : l_p^{k,t} \leq x_p \leq \bar{x}_p^{k,t}\}, \\ \Omega^{k,t_2} &= \Omega^{k,t} \cap \{\boldsymbol{x} \in \mathbb{R}^n : \bar{x}_p^{k,t} \leq x_p \leq u_p^{k,t}\}, \end{aligned} \tag{1.3}$$

tomando índices $t_1 = tot + 1$ y $t_2 = tot + 2$. Actualizamos $Q_k = (Q_k \setminus \{t\}) \cup \{t_1, t_2\}$, $tot = tot + 2$, y pasamos a la etapa 2.

Etapa 2a. Resolvemos $\mathrm{LP}(\Omega^{k,t_1})$. Si es infactible, seguimos a la Etapa 2b. En caso contrario, sea $(\bar{\boldsymbol{x}}^{k,t_1}, \bar{\boldsymbol{X}}^{k,t_1})$ la solución óptima de $\mathrm{LP}(\Omega^{k,t_1})$ con valor óptimo $LB_{k,t_1} = v[\mathrm{LP}(\Omega^{k,t_1})]$. Se selecciona la variable de ramificación utilizando (1.2).

- Si $\theta_p^{k,t_1} = 0$ entonces $\bar{\boldsymbol{x}}^{k,t_1}$ es factible en el problema $PP(\Omega)$. Actualizamos $Q_k = Q_k \setminus \{t_1\}$. Si $v[\mathrm{LP}(\Omega^{k,t_1})] < v^*$, entonces se actualizan también $\boldsymbol{x}^* = \bar{\boldsymbol{x}}^{k,t_1}$ y $v^* = LB_{k,t_1}$.

Etapa 2b. Resolvemos $\mathrm{LP}(\Omega^{k,t_2})$. Si es infactible, seguimos a la Etapa 3. En caso contrario, sea $(\bar{\boldsymbol{x}}^{k,t_2}, \bar{\boldsymbol{X}}^{k,t_2})$ la solución óptima de $\mathrm{LP}(\Omega^{k,t_2})$ con valor óptimo $LB_{k,t_2} = v[\mathrm{LP}(\Omega^{k,t_2})]$. Se selecciona la variable de ramificación utilizando (1.2).

- Si $\theta_p^{k,t_2} = 0$ entonces $\bar{\boldsymbol{x}}^{k,t_2}$ es factible en el problema $PP(\Omega)$. Actualizamos $Q_k = Q_k \setminus \{t_2\}$. Si $v[\mathrm{LP}(\Omega^{k,t_2})] < v^*$, entonces se actualizan también $\boldsymbol{x}^* = \bar{\boldsymbol{x}}^{k,t_2}$ y $v^* = LB_{k,t_2}$.

Etapa 3 (Poda). Se actualiza $Q_{k+1} = Q_k \setminus \{t \in Q_k : LB_{k,t} \geq v^*\}$. Si $Q_{k+1} = \emptyset$ el algoritmo termina. En caso contrario, para cada $t \in Q_{k+1}$, actualizamos $\Omega^{k+1,t} = \Omega^{k,t}$, $(\boldsymbol{x}^{k+1,t}, \boldsymbol{X}^{k+1,t}) = (\boldsymbol{x}^{k,t}, \boldsymbol{X}^{k,t})$, $LB_{k+1,t} = LB_{k,t}$, y $\theta_p^{k+1,t} = \theta_p^{k,t}$ y fijamos $k = k + 1$.

Etapa 4 (Selección de nodo). Seleccionamos (k, t) de forma que $t \in \arg\min_{t \in Q_k}\{LB_{k,t}\}$ y volvemos a la Etapa 1.

Sherali y Tuncbilek (1992) prueban que el algoritmo anteriormente descrito converge al óptimo global de $PP(\Omega)$. No obstante, seguiremos la exposición recogida en González-Rodríguez (2022) porque incluye comentarios adicionales que resultan útiles para demostrar que ciertas modificaciones del algoritmo RLT mantienen la convergencia al óptimo. Con este fin, presentamos el siguiente lema.

Lema 1.20. *Para cualquier rama infinita del algoritmo RLT se verifica que* $v[PP(\Omega)] \geq LB_{k,t(k)}$ *para todo* $k \in K \subset \mathbb{N}$*, siendo* K *un conjunto infinito.*

Demostración. Por cómo es el algoritmo, nunca se selecciona un nodo cuyo valor óptimo sea superior al óptimo del problema. Esto es cierto ya que para cada iteración k, se verifica lo siguiente:

(1) Cualquier punto $(\boldsymbol{x}, E(\boldsymbol{x}))$ factible en $\mathrm{LP}(\Omega^{k,t})$, con $\boldsymbol{x}$ factible en $PP(\Omega)$, es factible en $\mathrm{LP}(\Omega^{k,t_1})$ o en $\mathrm{LP}(\Omega^{k,t_2})$. Esto se debe a cómo se realiza la ramificación en (1.3) y la expresión de las restricciones bound factor.

(2) El RLT sólo termina una rama en un nodo $\mathrm{LP}(\Omega^{k,t(k)})$ si el nodo es infactible o si $|\bar{X}_{J\cup\{j\}} - \bar{x}_j \bar{X}_J| = 0$ para cada $J \subset N^\delta$, $1 \leq |J| \leq \delta - 1$, y $j \in N$, siendo $(\bar{\boldsymbol{x}}^{k,t(k)}, \bar{\boldsymbol{X}}^{k,t(k)})$ la solución óptima de $\mathrm{LP}(\Omega^{k,t(k)})$.

(3) Se selecciona un nodo de Q_k con menor óptimo.

(4) En la Etapa 3, se actualiza $Q_{k+1} = Q_k \setminus \{t \in Q_k : LB_{k,t} \geq v^*\}$; es decir, que se eliminan de Q_k los nodos cuyo óptimo es mayor o igual que la mejor solución encontrada en la iteración k. □

De este modo, estamos en disposición de enunciar el teorema que establece la convergencia del algoritmo RLT al óptimo global del problema.

Teorema 1.21. *Si empleamos el algoritmo RLT para resolver $PP(\Omega)$, existen dos escenarios posibles:*

1. *El algoritmo RLT termina en un número finito de iteraciones, y la solución hallada es un óptimo global de $PP(\Omega)$.*

2. *Se genera un número infinito de iteraciones de forma que, a lo largo de cualquier rama infinita del árbol de ramificación y acotación, cualquier punto de acumulación de la sucesión de variables $\boldsymbol{x}$ generadas al resolver las relajaciones lineales es un óptimo global de $PP(\Omega)$.*

Demostración. Empezamos señalando que en la Etapa 1 del algoritmo, todos los puntos $(\boldsymbol{x}, E(\boldsymbol{x}))$ que son factibles en $\text{LP}(\Omega^{k,t})$ y que verifican que $\boldsymbol{x}$ es factible en $PP(\Omega)$, son factibles o bien en $\text{LP}(\Omega^{k,t_1})$ o bien en $\text{LP}(\Omega^{k,t_2})$. Esto es consecuencia de cómo se ramifica en (1.3) y de la expresión de las restricciones bound factor. Además, si el algoritmo termina en un número finito de iteraciones k, entonces $Q_k = \emptyset$. Con lo cual, el valor de v^* en la iteración k es el óptimo de $PP(\Omega)$, porque los nodos con menor valor óptimo ya se han resuelto antes de la iteración k y v^* es el mejor de los valores encontrados para una solución factible de $PP(\Omega)$.

Supongamos entonces que el algoritmo no termina en un número finito de iteraciones y consideremos una rama arbitraria infinita del árbol de ramificación y acotación. Sea $\{\Omega^{k,t(k)}\}_{k\in K}$ la sucesión de particiones anidadas de esa rama, donde $t(k) \in Q_k$ para cada $k \in K \subset \mathbb{N}$, donde K es un conjunto infinito. Sea $(\boldsymbol{x}^{k,t(k)}, \boldsymbol{X}^{k,t(k)})$ la solución del nodo $\text{LP}(\Omega^{k,t(k)})$. La sucesión $\{\boldsymbol{x}^{k,t(k)}\}$ está acotada ya que pertenece a un conjunto acotado Ω. Con lo cual, por el Teorema de Bolzano-Weierstrass sabemos que al menos contiene una subsucesión convergente. Sea $\{\boldsymbol{x}^{k,t(k)}\}_{k\in K_1} \to \bar{\boldsymbol{x}}$ una de esas subsucesiones convergentes, donde $K_1 \subset K$ es un conjunto infinito. La demostración pasa entonces a probar que $\bar{\boldsymbol{x}}$ es una solución óptima de $PP(\Omega)$, dado que cualquier punto de acumulación es el límite de tal subsucesión.

Puesto que $\{\Omega^{k,t(k)}\}_{k\in K_1}$ es una subsucesión infinita, entonces existe al menos una variable en la que el algoritmo se ramifique un número infinito de veces. Sea x_p una de esas variables, y sea $K_2 \subset K_1$ el conjunto de todas las iteraciones en donde la variable de ramificación es x_p. Además, hay un conjunto infinito $K_3 \subset K_2$ y un conjunto de índices $J' \subset N^\delta$, con $1 \leq |J'| \leq \delta - 1$, tal que:

$$\left| X^{k,t(k)}_{J'\cup\{p\}} - x^{k,t(k)}_p X^{k,t(k)}_{J'} \right| \geq \left| X^{k,t(k)}_{J\cup\{j\}} - x^{k,t(k)}_j X^{k,t(k)}_{J} \right| \tag{1.4}$$

para cada $k \in K_3$, $J \subset N^\delta$, $1 \leq |J| \leq \delta - 1$, y $j \in N$.

Ahora, puesto que la sucesión $\{(\boldsymbol{x}^{k,t(k)}, \boldsymbol{X}^{k,t(k)}, \boldsymbol{l}^{k,t(k)}, \boldsymbol{u}^{k,t(k)})\}_{k\in K_3}$ está acotada gracias al Corolario 1.18, existe una subsucesión convergente $\{(\boldsymbol{x}^{k,t(k)}, \boldsymbol{X}^{k,t(k)}, \boldsymbol{l}^{k,t(k)}, \boldsymbol{u}^{k,t(k)})\}_{k\in K_4} \to (\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}})$, con $K_4 \subset K_3$. Con lo cual, se verifica que $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ es factible en $\text{LP}(\bar{\Omega})$, donde

$$\bar{\Omega} = \{\boldsymbol{x} \in \mathbb{R}^n : 0 \leq \bar{l}_j \leq x_j \leq \bar{u}_j \text{ para cada } j \in N\}.$$

Además, teniendo en cuenta cómo se realiza la ramificación en (1.3), se verifica que, para cada $k, k' \in K_4$ con $k' > k$, $x^{k,t(k)}_p \notin (l^{k',t(k')}_p, u^{k',t(k')}_p)$. Con lo cual, puesto que $\bar{x}_p \in [\bar{l}_p, \bar{u}_p]$, o bien $\bar{x}_p = \bar{l}_p$ o $\bar{x}_p = \bar{u}_p$. Con lo cual, por el Lema 1.19, $\bar{X}_{J'\cup\{p\}} = \bar{x}_p \bar{X}_{J'}$.

Por lo tanto, empleando la desigualdad (1.4) y haciendo tender k a infinito, se verifica que $\bar{X}_{J\cup\{j\}} = \bar{x}_j \bar{X}_J$ para cada $J \subset N^\delta$, $1 \leq |J| \leq \delta - 1$, $j \in N$. Con lo cual, $\bar{\boldsymbol{x}}$ es factible en $PP(\Omega)$ y $[\phi_0]_L(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}}) = \phi_0(\bar{\boldsymbol{x}}) \geq v[PP(\Omega)]$, donde $[\phi_0]_L$ es la función objetivo de $RLT(PP(\Omega))$ y ϕ_0 la función objetivo de $PP(\Omega)$. Por último, empleando el Lema 1.20 y haciendo tender k a infinito, se tiene que $v[PP(\Omega)] \geq [\phi_0]_L(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}}) = \phi_0(\bar{\boldsymbol{x}})$. Con lo cual, $\bar{\boldsymbol{x}}$ es la solución óptima de $PP(\Omega)$. □

Cuando se emplean algoritmos de ramificación y acotación para resolver problemas lineales con variables enteras, una de las claves es que cuando en una iteración se obtiene que la solución óptima de la relajación continua no es factible, el proceso de ramificación impone que dicha solución no sea factible en las relajaciones continuas de la misma rama, ya que en caso contrario dicha solución podría ser el óptimo de uno de los nodos hijos.

En efecto, supongamos que en una iteración dada se obtiene una solución $\bar{\boldsymbol{x}}$. Si esta no es factible entonces existe $\bar{x}_j$ tal que $\bar{x}_j \notin \mathbb{Z}$. Supongamos que se decide ramificar por la variable x_j, entonces una rama impondrá $x_j \leq \lfloor \bar{x}_j \rfloor$ y la otra $x_j \geq \lceil \bar{x}_j \rceil$, de forma que en esta rama $\bar{\boldsymbol{x}}$ no volverá a ser factible.

Aunque pareciera que en el proceso de ramificación por violaciones RLT no se diera esta situación, veamos que también sucede.

Proposición 1.22. *Sea* $(\bar{\boldsymbol{x}}^{k_0,t_0}, \bar{\boldsymbol{X}}^{k_0,t_0})$ *la solución óptima de* $\mathrm{LP}(\Omega^{k_0,t_0})$ *y sea* x_p *la variable de ramificación seleccionada, con* $\theta_p^{k_0,t_0} > 0$. *Si se ramifica por dicho punto, entonces* $(\bar{\boldsymbol{x}}^{k_0,t_0}, \bar{\boldsymbol{X}}^{k_0,t_0})$ *no es factible en ninguno de los nodos hijos.*

Demostración. En efecto, ahora para cualquier nodo hijo se verifica $x_p \leq \bar{x}_p^{k_0,t_0}$ o bien $x_p \geq \bar{x}_p^{k_0,t_0}$. Vamos a probar que es cierto en el primer caso, y en el otro el resultado es análogo. Para $k > k_0$, se pueden dar los siguientes casos: Si $u_p^{k,t} < \bar{x}_p^{k_0,t_0}$, entonces el resultado es evidente. Entonces únicamente es necesario comprobar el caso en que $u_p^{k,t} = \bar{x}_p^{k_0,t_0}$. Teniendo en cuenta que, seleccionado el criterio de ramificación, $\theta_p^{k,t}$ depende exclusivamente del punto seleccionado, $(\bar{\boldsymbol{x}}^{k,t}, \bar{\boldsymbol{X}}^{k,t})$ verifica que $\theta_p^{k,t} > 0$. No obstante, en virtud del Lema 1.19, como $u_p^{k,t} = \bar{x}_p^{k_0,t_0}$, se verifica que $\theta_p^{k,t} = 0$. De esta forma, la contradicción se da al asumir que esta es una solución factible del problema. □

Aunque este resultado parece interesante porque garantiza que en cada iteración se descarta la solución actual, más adelante veremos que no entra en juego ya que no se ramificará directamente por la solución obtenida, sino que se preferirá ponderar la solución con el punto medio del rango de la variable a fin de obtener ramas de dificultad más parecida.

El algoritmo descrito hasta ahora proporciona una base teórica sólida para resolver problemas polinómicos no convexos. No obstante, su aplicación directa puede resultar computacionalmente costosa, especialmente cuando el número de variables o el grado del problema aumentan. Por este motivo, en la práctica resulta necesario incorporar mejoras que reduzcan el tamaño de las relajaciones o aceleren el proceso de ramificación y acotación.

1.2.1. Mejoras del RLT

Dedicamos esta sección del capítulo a discutir diversas mejoras que se pueden incorporar al algoritmo basado en la técnica RLT. Para ello, seguiremos principalmente la exposición presentada en González-Rodríguez et al. (2022) e incluiremos adicionalmente uno de los resultados de González-Díaz et al. (2024), que será el eje central de la próxima sección.

J-sets

Como se mencionó anteriormente, el número de bound factors crece de manera exponencial a medida que aumenta el número de variables o el grado del problema. En Dalkiran y Sherali (2013) se demuestra que no es necesario considerar todas estas restricciones, sino que basta con atender a un subconjunto de ellas.

Proposición 1.23. *Para cada monomio* J *presente en* $PP(\Omega)$, *consideremos la siguiente familia de desigualdades:*

$$\left[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \right]_L \geq 0, \ \forall J_1 \cup J_2 = J. \tag{1.5}$$

Si se sustituyen las restricciones bound factor por las anteriormente descritas, entonces el algoritmo sigue convergiendo al óptimo global.

Demostración. Para un monomio y una solución factible dados, la convergencia del Teorema 1.21 únicamente requiere que las identidades RLT se reproduzcan cuando todas las variables que conforman el monomio alcanzan su cota inferior o superior. Atendiendo a las demostraciones de los Lemas 1.16 y 1.19, no resulta complicado ver que estas siguen siendo válidas en el caso en que sustituyamos las restricciones bound factor por las que aparecen en (1.5). □

Esto tendrá especial relevancia en los problemas donde el número de monomios distintos presentes sea notablemente menor que el total de monomios posibles de hasta orden δ, ya que permite reducir de forma considerable el número de restricciones bound factor. Además, apoyándonos en el Lema 1.16, sabemos que cuando $J' \subset J''$, las restricciones derivadas de J'' implican las de J'. De esta forma, se propone la siguiente rutina para generar los J-sets:

1. Construir una lista con todos los monomios presentes en $PP(\Omega)$.
2. Para cada elemento de la lista, comprobar si existe algún otro monomio que lo contenga. En caso afirmativo, descartar el elemento actual.
3. Para cada monomio restante, generar las correspondientes restricciones bound factor.

Ejemplo 1.24. En nuestro ejemplo, la reformulación del problema empleando J-sets es de la forma:

$$\begin{aligned} \text{minimizar} \quad & X_{\{1,2,2\}} - X_{\{1,1,2\}} + 2x_1 \\ \text{sujeto a} \quad & 3X_{\{1,2\}} - x_2 \geq 2 \\ & x_1 - x_2 + X_{\{1,2\}} = 4 \\ & \boldsymbol{x} \in [1,2] \times [3,4], \end{aligned}$$

a la que adicionalmente habrá que añadir las siguientes bound factor linealizadas asociadas a los monomios $\{1,1,2\}$ y $\{1,2,2\}$, omitiendo las asociadas a $\{1,1,1\}$ y $\{2,2,2\}$ al no ser monomios presentes en el problema:

$$\begin{cases} -X_{\{1,1,2\}} + 4X_{\{1,1\}} + 4X_{\{1,2\}} - 16x_1 - 4x_2 + 16 \geq 0, \\ X_{\{1,1,2\}} - 3X_{\{1,1\}} - 4X_{\{1,2\}} + 12x_1 + 4x_2 - 12 \geq 0, \\ X_{\{1,1,2\}} - 4X_{\{1,1\}} - 3X_{\{1,2\}} + 12x_1 + 2x_2 - 8 \geq 0, \\ -X_{\{1,1,2\}} + 3X_{\{1,1\}} + 3X_{\{1,2\}} - 9x_1 - 2x_2 + 6 \geq 0, \\ -X_{\{1,1,2\}} + 4X_{\{1,1\}} + 2X_{\{1,2\}} - 8x_1 - x_2 + 4 \geq 0, \\ X_{\{1,1,2\}} - 3X_{\{1,1\}} - 2X_{\{1,2\}} + 6x_1 + x_2 - 3 \geq 0, \\ \\ -X_{\{1,2,2\}} + 8X_{\{1,2\}} + 2X_{\{2,2\}} - 16x_1 - 16x_2 + 32 \geq 0, \\ X_{\{1,2,2\}} - 7X_{\{1,2\}} - 2X_{\{2,2\}} + 12x_1 + 14x_2 - 24 \geq 0, \\ -X_{\{1,2,2\}} + 6X_{\{1,2\}} + 2X_{\{2,2\}} - 9x_1 - 12x_2 + 18 \geq 0, \\ X_{\{1,2,2\}} - 8X_{\{1,2\}} - X_{\{2,2\}} + 16x_1 + 8x_2 - 16 \geq 0, \\ -X_{\{1,2,2\}} + 7X_{\{1,2\}} + X_{\{2,2\}} - 12x_1 - 7x_2 + 12 \geq 0, \\ X_{\{1,2,2\}} - 6X_{\{1,2\}} - X_{\{2,2\}} + 9x_1 + 6x_2 - 9 \geq 0. \end{cases}$$

En Dalkiran y Sherali (2013) se prueba que la relajación base es al menos tan ajustada como la que se obtiene empleando J-sets, lo cual resulta natural si se tiene en cuenta que estos se construyen a partir de un subconjunto de las restricciones bound factor de la relajación original.

Pese a que el número de restricciones sigue aumentando de manera exponencial, esta reducción da lugar a una relajación lineal de menor tamaño, lo que acelera la resolución de cada iteración del algoritmo. No obstante, al disponer de una relajación menos ajustada, puede ser necesario un mayor número de iteraciones para alcanzar el óptimo. Aun así, los estudios computacionales muestran que el uso de los *J-sets* mejora el rendimiento global del RLT.

La formulación anterior está planteada para problemas con variables continuas. Sin embargo, en muchas aplicaciones aparecen decisiones discretas, que se modelan de forma natural mediante variables enteras o binarias. Incorporar este tipo de variables modifica el comportamiento del algoritmo, ya que además de las violaciones asociadas a las identidades RLT deben considerarse también las violaciones de integralidad.

1.3. Extensión del algoritmo basado en RLT a problemas con variables enteras

Una vez descrito el caso continuo, pasamos a considerar problemas con variables enteras. Empezaremos esta sección discutiendo los posibles enfoques a la hora de añadir variables enteras al problema, para lo que nos apoyaremos en las disertaciones del final del primer capítulo de González-Rodríguez (2022) y en González-Díaz et al. (2024).

Empezamos definiendo lo que es un problema de optimización con variables enteras, distinguiendo entre variables binarias, enteras no binarias y continuas, puesto que será de utilidad en el próximo capítulo.

Definición 1.25. Sea N el conjunto de variables, con $N = N_B \cup N_Z \cup N_C$, donde N_B, N_Z y N_C son respectivamente el conjunto de variables binarias, enteras no binarias y continuas. Sea $N_{NB} := N_Z \cup N_C$ el conjunto de variables no binarias. Entonces, un problema de optimización polinómica con variables enteras es un problema de optimización de la forma:

$$\begin{array}{rll} \text{minimizar} & \phi_0(\boldsymbol{x}) & \\ \text{sujeto a} & \phi_r(\boldsymbol{x}) \geq \beta_r, & r = 1, \ldots, R_1, \\ & \phi_r(\boldsymbol{x}) = \beta_r, & r = R_1 + 1, \ldots, R, \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^{|N|}, & \end{array} \tag{$MIPP^{\Omega}$}$$

donde cada $\phi_r(\boldsymbol{x})$ es un polinomio de grado $\delta_r \in \mathbb{N}$, $\delta = \max_{r \in \{0,\ldots,R\}} \delta_r$ es el grado del problema, y $\Omega = \{\boldsymbol{x} \in \{0,1\}^{|N_B|} \times \mathbb{Z}^{|N_Z|} \times \mathbb{R}^{|N_C|} : 0 \leq l_j \leq x_j \leq u_j < \infty, \forall j \in N_{NB}\} \subset \mathbb{R}^{|N|}$.

Se define entonces la relajación lineal con variables enteras como el siguiente problema de programación lineal con variables enteras:

$$\begin{array}{rll} \text{minimizar} & [\phi_0(\boldsymbol{x})]_L & \\ \text{sujeto a} & [\phi_r(\boldsymbol{x})]_L \geq \beta_r, & r = 1, \ldots, R_1, \\ & [\phi_r(\boldsymbol{x})]_L = \beta_r, & r = R_1 + 1, \ldots, R, \\ & \left[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\right]_L \geq 0, & \forall J_1 \cup J_2 = J, \\ & & J \text{ monomio del problema} \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^{|N|}, & \end{array} \tag{$RLT(MIPP^{\Omega})$}$$

donde cada $\phi_r(\boldsymbol{x})$ es un polinomio de grado $\delta_r \in \mathbb{N}$, $\delta = \max_{r \in \{0,\ldots,R\}} \delta_r$ es el grado del problema, y $\Omega = \{\boldsymbol{x} \in \{0,1\}^{|N_B|} \times \mathbb{Z}^{|N_Z|} \times \mathbb{R}^{|N_C|} : 0 \leq l_j \leq x_j \leq u_j < \infty, \forall j \in N_{NB}\} \subset \mathbb{R}^{|N|}$. A la anterior relajación la denominaremos relajación entera. Adicionalmente, emplearemos ${}^{\text{CR}}RLT(MIPP^{\Omega})$ para referirnos a la relajación continua del problema lineal con variables enteras anterior. Dependiendo de cuál sea la relajación que pretendamos resolver, tenemos dos formas de proceder. Por una parte, podemos proceder como se hacía en la primera sección y trabajar con la relajación continua del problema. De forma análoga a como realizábamos en (1.2), es posible definir una regla de ramificación para las violaciones de integralidad de una variable:

$$\begin{gathered} p \in \arg\max_{j \in N} \{\rho_j\}, \text{ donde} \\ \rho_j = \min\{x_j - \lfloor x_j \rfloor, \lceil x_j \rceil - x_j\} \text{ para cada } j \in N, \end{gathered} \tag{1.6}$$

donde $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ es la solución óptima del nodo correspondiente. Con esta elección, la ramificación se realiza añadiendo la cota $x_p \leq \lfloor \bar{x}_p \rfloor$ en uno de los nodos hijos y $x_p \geq \lceil \bar{x}_p \rceil$ en el otro, actualizando además las restricciones bound factor en las que interviene x_p.

Teorema 1.26. *El algoritmo basado en RLT aplicado sobre las relajaciones continuas* ${}^{\mathrm{CR}}RLT(MIPP^{\Omega})$ *converge al óptimo global de* $MIPP^{\Omega}$ *si se aplican las reglas de ramificación de violaciones de identidades RLT y violaciones de integralidad, independientemente del orden en que se comprueben en cada nodo.*

Por otra parte, podemos limitarnos a resolver las relajaciones enteras en cada nodo del árbol de ramificación y acotación.

Teorema 1.27. *El algoritmo basado en la técnica RLT converge al óptimo global de* $MIPP^{\Omega}$ *si se reemplazan las relajaciones* ${}^{\mathrm{CR}}RLT(MIPP^{\Omega})$ *por las relajaciones enteras definidas en* $RLT(MIPP^{\Omega})$.

En este caso, no es necesario tener en cuenta las violaciones de integralidad en cada iteración, que desaparecen al resolver la relajación entera del problema.

1.3.1. Resolución de la relajación continua

Hasta ahora, en el algoritmo de ramificación y acotación, cada vez que se resolvía una relajación lineal en un nodo, las únicas violaciones que evitaban que esta fuera una solución factible eran las asociadas a las variables RLT. Sin embargo, al añadir variables enteras, también hay que tener en cuenta las violaciones por integralidad, que como se afirma en el Teorema 1.26, de alguna forma han de verse reflejadas en la regla de ramificación.

Una vez tenidas en cuenta, puesto que reflejan violaciones distintas, puede resultar de interés que la regla priorice uno de los tipos de violación. Atendiendo a lo que hacen otros solvers, Couenne prioriza ramificar en las variables enteras mientras que Xpress no da prioridad a ninguno, sino que calcula valores asociados a una estrategia de ramificación basada en fiabilidad (detallado en Achterberg et al. (2005)). En González-Díaz et al. (2024) se estudian las dos situaciones extremas:

- *RLT-first*, donde se priorizan las violaciones de las variables RLT, de forma que hasta que en un nodo no se verifiquen todas las identidades RLT no se consideran las violaciones de integralidad. Para que se ramifique por una violación de integralidad $\rho_p^{k,t}$, se ha de verificar que $\theta_p^{k,t} = 0$.

- *Integrality-first*, donde todas las variables enteras necesitan tomar un valor entero antes de que se tengan en cuenta las identidades RLT. Para que se ramifique por una violación $\theta_p^{k,t}$, tiene que pasar que $\rho_p^{k,t} = 0$.

1.3.2. Resolución de la relajación entera

Apoyándonos en el Teorema 1.27, es posible resolver un problema lineal con variables enteras y así seguir pensando únicamente en violaciones RLT. Al resolver una relajación más ajustada, es de esperar que se requiera un menor número de iteraciones, a costa de problemas más difíciles de resolver, por lo que cada iteración requerirá mayor tiempo.

Si bien puede parecer similar al enfoque *integrality-first*, vamos a dejar clara su diferencia. Mientras que *integrality-first* tiene ramificaciones por violaciones de integralidad, si en una rama queda completamente definido su valor (lo cual puede suceder ya que el número de ramificaciones que se puede hacer sobre una variable entera es finito), en el resto de problemas hijos de esta rama dicha variable estará completamente determinada. Por el contrario, resolviendo las relajaciones enteras del problema, las condiciones de integralidad deben imponerse de nuevo en cada nodo. De esta forma, lo que es habitual que suceda es que *integrality-first* termine con más ramas pero donde los valores de las variables enteras puedan quedar determinados.

Hasta este punto, la exposición se ha centrado en los fundamentos teóricos del algoritmo RLT y en algunas de sus extensiones. Sin embargo, para que estas ideas resulten útiles desde el punto de vista computacional, es necesario incorporarlas en una implementación capaz de generar las reformulaciones, resolver las relajaciones lineales correspondientes y gestionar el proceso de ramificación y poda.

En este contexto aparece RAPOSa, que proporciona una implementación del enfoque descrito y sirve como punto de partida para los desarrollos que se estudiarán en los capítulos posteriores. Por ello, antes de introducir nuevas reformulaciones, conviene explicar brevemente cuál es el papel de RAPOSa y cómo se relaciona con el algoritmo presentado en este capítulo.

1.4. Implementación en RAPOSa

Basándose en el algoritmo RLT descrito a lo largo de este capítulo se desarrolla RAPOSa (*Reformulation Algorithm for Polynomial Optimization–Santiago*), un software de optimización global diseñado para la solución de problemas polinómicos con variables enteras. El equipo de desarrollo de RAPOSa está formado principalmente por investigadores de la Universidad de Santiago de Compostela y de la Universidad de Vigo, en particular asociados al CITMAGa, en cuyo grupo tuve la oportunidad de incorporarme entre octubre de 2025 y junio de 2026, con el fin de colaborar en el desarrollo del mismo sobre la base de los resultados teóricos descritos en el trabajo.

El funcionamiento de RAPOSa consiste en generar las relajaciones lineales de los problemas, resolverlas empleando un optimizador auxiliar, y utilizar su solución para guiar el proceso de ramificación y poda. Con el fin de mejorar su rendimiento, en González-Rodríguez et al. (2022) y González-Díaz et al. (2024) podemos encontrar una serie de mejoras sobre el algoritmo original, en las que aquí no profundizaremos al no ser el objeto central de trabajo. Entre ellas, se incluyen el ajuste de cotas o la llamada a solvers no lineales auxiliares. Una de esas posibles mejoras consiste en la reformulación de la parte binaria del problema antes de iniciar el proceso de resolución. Esto será lo que nos ocupe el siguiente capítulo.

Capítulo 2

Reformulación de productos de variables binarias

Una vez establecido el objeto de estudio de este trabajo, ponemos el foco en las variables binarias. Por las características de dichas variables, podremos plantear reformulaciones al problema original que, pese a que son menos ajustadas, proporcionarán mejores tiempos de resolución. Para el desarrollo de este capítulo empezaremos cubriendo los resultados recogidos en Elloumi y Verchère (2023). Puesto que es en lo que se fundamenta dicho trabajo, definimos un problema de optimización polinómica con variables binarias. En una segunda sección, recogemos los resultados obtenidos por el grupo de trabajo en el desarrollo de mis prácticas, que extrapolan los resultados de Elloumi y Verchère (2023) a problemas de optimización polinómica general.

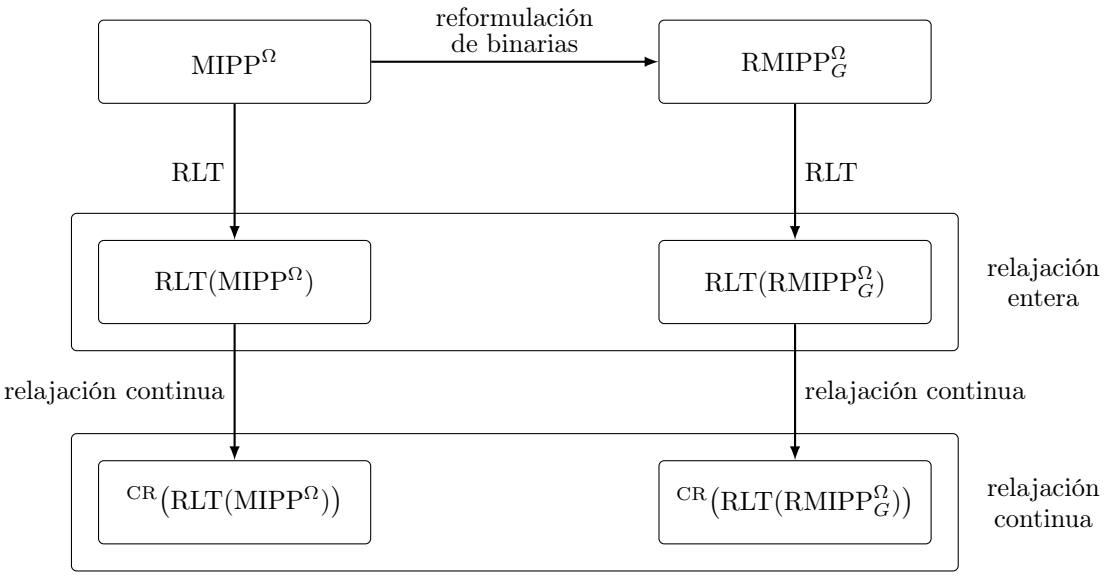


Figura 2.1: Esquema de reformulación y relajaciones consideradas.

La Figura 2.1 resume las distintas formulaciones que compararemos a lo largo de esa sección. Partiremos de un problema polinómico con variables enteras y estudiaremos el efecto de aplicar, antes de la técnica RLT, una reformulación específica de los productos que involucran variables binarias. De esta forma aparecen dos caminos posibles: aplicar directamente RLT al problema original o reformular primero la parte binaria y aplicar después RLT al problema resultante.

El interés de esta comparación reside en que ambos enfoques pueden dar lugar a problemas equivalentes desde el punto de vista de la solución entera, pero no necesariamente a relajaciones igual de ajustadas ni igual de costosas de resolver. Por ello, además de probar equivalencias entre formulaciones, será importante distinguir entre el comportamiento de sus relajaciones continuas y el de sus relajaciones enteras.

Los principales resultados en este apartado serán:

- Los problemas $MIPP^{\Omega}$ y $RMIPP_G^{\Omega}$ son equivalentes.
- La relajación $RLT(PP(\Omega))$ es al menos tan ajustada como $RLT(RMIPP_G^{\Omega})$, y lo mismo ocurre con sus relajaciones continuas.
- $RLT(PP(\Omega))$ y $RLT(RMIPP_G^{\Omega})$ son equivalentes.

Comenzamos estudiando el caso en el que todas las variables son binarias. Esto permite aislar con claridad la idea central de esta sección: sustituir productos de variables binarias por nuevas variables construidas siguiendo un determinado patrón de linealización. En este contexto, las propiedades de las variables binarias simplifican la reformulación y permiten comparar distintas formas de construir dichos patrones.

El análisis de este caso servirá como punto de partida para la sección posterior, donde se trasladarán estas ideas a problemas polinómicos generales. Allí las variables binarias aparecerán mezcladas con variables continuas o enteras, de modo que la reformulación de la parte binaria deberá combinarse con las restricciones RLT asociadas al resto del monomio.

2.1. Reformulación de problemas de optimización polinómica con variables binarias

Definición 2.1. Un problema de optimización polinómica con variables binarias es un problema de optimización de la forma:

$$\begin{aligned} \text{minimizar} \quad & \sum_{M \in P} c_M \prod_{i \in M} y_i \\ \text{sujeto a} \quad & y_i \in \{0,1\} \end{aligned} \tag{BPO}$$

Cuando presentemos ejemplos de problemas de optimización con variables binarias, emplearemos las variables y para denotar las variables binarias y las variables x para las no binarias, con el objetivo de facilitar la lectura.

Ejemplo 2.2. Un ejemplo de lo anterior podría ser el siguiente:

$$\begin{cases} \underset{\boldsymbol{y}}{\text{minimizar}} & y_1y_2y_3 - y_1y_3y_4 + 3y_3y_4y_5 \\ \text{sujeto a} & \boldsymbol{y} \in \{0,1\}^5. \end{cases}$$

Es habitual ver linealizado el producto de variables binarias de la siguiente forma: Sea Y_M la variable que representa a $\prod_{i \in M} y_i$, entonces $Y_M = \prod_{i \in M} y_i$ si se tienen las siguientes restricciones:

$$\begin{cases} Y_M \leq y_i, & \forall i \in M, \\ Y_M \geq 1 + \sum_{i \in M} (y_i - 1), \\ Y_M \geq 0. \end{cases}$$

De esta forma, un problema de optimización polinómica con variables binarias se puede expresar como un problema lineal.

$$\begin{aligned} \text{minimizar} \quad & \sum_{M\in P} c_M Y_M \\ \text{sujeto a} \quad & Y_M \leq y_i, \quad \forall i \in M, \forall M \in P, \\ & Y_M \geq 1 + \sum_{i\in M}(y_i - 1), \forall M \in P, \\ & Y_M \geq 0, \forall M \in P, \\ & y_i \in \{0,1\} \end{aligned} \tag{SL}$$

Ejemplo 2.3. En nuestro caso, lo anterior se convierte en:

$$\begin{cases} \underset{\boldsymbol{y}}{\text{minimizar}} \quad Y_{\{1,2,3\}} - Y_{\{1,3,4\}} + 3Y_{\{1,4,5\}} \\ \text{sujeto a} \quad \boldsymbol{y} \in \{0,1\}^5, \end{cases}$$

a la que hay que añadir las siguientes restricciones:

$$\begin{cases} Y_{\{1,2,3\}} \leq y_1, & Y_{\{1,2,3\}} \leq y_2, & Y_{\{1,2,3\}} \leq y_3, & Y_{\{1,2,3\}} \geq y_1 + y_2 + y_3 - 2, & Y_{\{1,2,3\}} \geq 0, \\ Y_{\{1,3,4\}} \leq y_1, & Y_{\{1,3,4\}} \leq y_3, & Y_{\{1,3,4\}} \leq y_4, & Y_{\{1,3,4\}} \geq y_1 + y_3 + y_4 - 2, & Y_{\{1,3,4\}} \geq 0, \\ Y_{\{3,4,5\}} \leq y_3, & Y_{\{3,4,5\}} \leq y_4, & Y_{\{3,4,5\}} \leq y_5, & Y_{\{3,4,5\}} \geq y_3 + y_4 + y_5 - 2, & Y_{\{3,4,5\}} \geq 0. \end{cases}$$

La anterior formulación se conoce como linealización estándar. Resultará de utilidad representarla como el grafo de la Figura 2.2 para visualizarla más claramente.

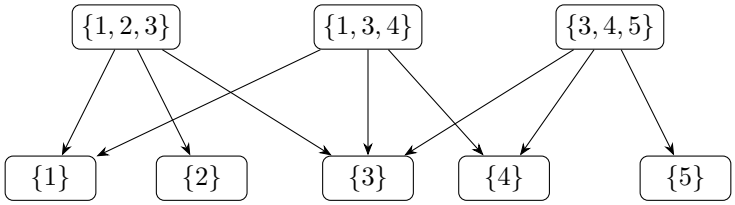


Figura 2.2: Grafo de linealización asociado a la linealización estándar del problema del Ejemplo 2.3.

Pese a su sencillez, esta resulta poco ajustada, y es posible mejorarla empleando lo que se conocen como patrones de linealización. Para presentarlos, es necesario tener claros los siguientes conceptos de teoría de grafos.

Definición 2.4. Un grafo es un conjunto de nodos V y un conjunto de aristas A, de forma que cada arista es un par de nodos del grafo. Si el par es ordenado, se dice que el grafo es dirigido. De lo contrario, se dice que el grafo es no dirigido.

Definición 2.5. Se dice que un grafo dirigido contiene un ciclo si existe un conjunto ordenado de nodos $\{v_1, \ldots, v_n\} \in V$ de forma que $(v_j, v_{j+1}) \in A$ para todo $j \in \{1, \ldots, n-1\}$ y $(v_n, v_1) \in A$.

Si un grafo dirigido no contiene ningún ciclo, se dice que es un grafo acíclico.

Definición 2.6. Dado un grafo dirigido $G = (V, A)$, se dice que un nodo $v \in V$ es un nodo raíz si no existe $w \in V$ tal que $(w, v) \in A$; y se dice que un nodo es terminal si no existe $w \in V$ tal que $(v, w) \in A$.

Definición 2.7. Dado un grafo dirigido $G = (V, A)$, se define la función de descendientes directos S_G^+ como:

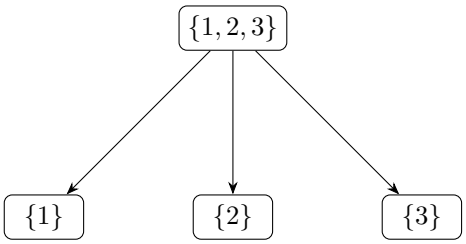
$$\begin{aligned} S_G^+ : V &\to \mathcal{P}(V) \\ v &\to \{w : (v, w) \in A\} \end{aligned}$$

Procedemos a definir los patrones de linealización.

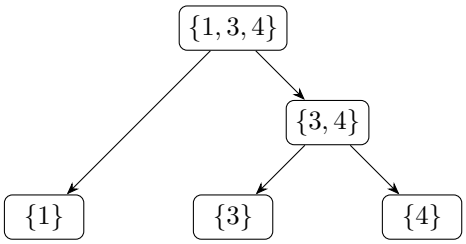
Definición 2.8. Dado un monomio M, conformado únicamente por variables binarias, se define un patrón de linealización de M como un grafo dirigido acíclico $G_M = (V_M, A_M)$ de forma que se verifican las siguientes condiciones:

- El grafo cuenta con un único nodo raíz M.
- Los nodos $v \in V_M$ son subconjuntos de M.
- Los únicos nodos terminales del grafo son los conjuntos $\{i\}$ con $i \in M$.
- Para todo nodo no terminal v se verifica $v = \bigcup_{t \in S^+_{G_M}(v)} t$ y $t \subsetneq v$ para todo $v \in V_M$ y para todo $t \in S^+_{G_M}(v)$.

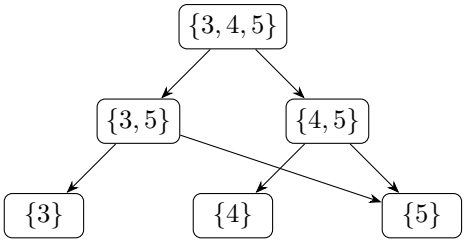
Gracias a los patrones de linealización podemos formalizar una gran variedad de formas de linealizar un monomio de binarias. En la Figura 2.3 podemos ver representados diversos patrones de linealización para un monomio de 3 binarias. De acuerdo con lo expuesto al principio de este capítulo, la linealización de la Figura 2.3a se corresponde con la linealización estándar. No obstante, parece que existe una



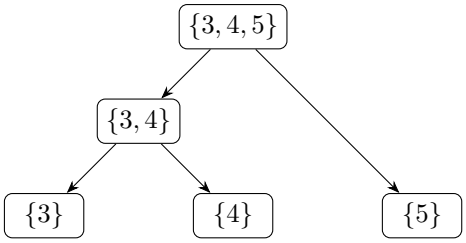
(a) Un patrón de linealización para $y_1y_2y_3$.



(b) Un patrón de linealización para $y_1y_3y_4$.



(c) Un patrón de linealización para $y_3y_4y_5$.



(d) Otro posible patrón para $y_3y_4y_5$.

Figura 2.3: Algunos patrones de linealización para los monomios del Ejemplo 2.3.

diferencia entre el patrón de la Figura 2.3c y los restantes, ya que para los nodos intermedios de grado dos, la variable 3 aparece en dos monomios distintos. A fin de marcar esa diferencia, presentamos la siguiente definición.

Definición 2.9. Dado un monomio M y un patrón de linealización asociado a dicho monomio $G_M = (V_M, A_M)$, se dice que el patrón de linealización es simple si el grafo es un árbol; es decir, para todo nodo no terminal $v \in V_M$ se tiene que el conjunto $S^+_{G_M}(v)$ forma una partición de v.

De esta forma, está claro que el patrón representado en la Figura 2.3c no es simple, mientras que los restantes sí lo son. Con ello aclarado, extendamos la noción de patrón de linealización de un monomio a los polinomios.

Definición 2.10 (Concatenación de patrones de linealización y grafos)**.** Sean M_1 y M_2 dos monomios, y sean $G_1 = (V_1, A_1)$, $G_2 = (V_2, A_2)$ sus respectivos patrones de linealización. Se define la concatenación de los patrones como el grafo $G = \mathcal{C}(G_1, G_2) = (V, A)$, donde:

- $V = V_1 \cup V_2$. En caso de que un nodo aparezca en ambos grafos, se considera una sola vez si y sólo si el subgrafo a partir de ese nodo es igual en ambos grafos. Si no son iguales, se enumeran arbitrariamente los nodos para diferenciarlos.

- $A = A_1 \cup A_2$, modificando las aristas necesarias si se enumeran nodos como se describe en el apartado anterior.

La anterior definición se puede extrapolar a un conjunto arbitrario de monomios sin más que aplicar de forma iterativa la concatenación:

$$\mathcal{C}(G_1, \dots, G_m) = \mathcal{C}(G_1, \mathcal{C}(G_2, \dots, G_m)) = \cdots = \mathcal{C}(G_1, \mathcal{C}(G_2, \mathcal{C}(\dots, \mathcal{C}(G_{m-1}, G_m)\dots)))$$

Definición 2.11. Dado un polinomio $P = \{M_1, \dots, M_{|P|}\}$ y un patrón de linealización asociado a cada monomio G_j, se define el grafo de linealización de P como la concatenación de patrones de cada monomio:

$$G = \mathcal{C}(G_1, \dots, G_{|P|}).$$

En la Figura 2.4 podemos ver representado el grafo de linealización asociado a los monomios del Ejemplo 2.3 empleando los patrones de linealización 2.3a, 2.3b y 2.3d. Es evidente que, como el subgrafo generado a partir de $\{3, 4\}$ es igual en 2.3b y en 2.3d, se reutiliza el mismo nodo. De esta forma, ya

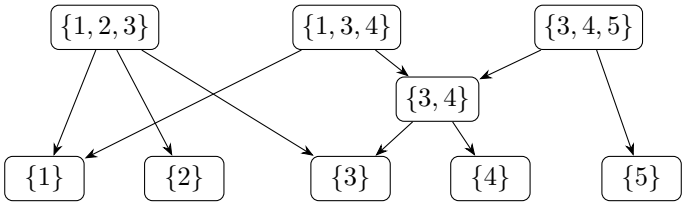


Figura 2.4: Grafo de linealización asociado al problema del Ejemplo 2.3 empleando los patrones de linealización 2.3a, 2.3b y 2.3d.

podemos plantear la reformulación del problema *BPO* usando patrones de linealización.

Definición 2.12. Dado un polinomio P y un grafo de linealización asociado $G = (V, A)$, se define el problema de programación lineal entera mixta asociado al grafo G como sigue:

$$\begin{aligned}
\text{minimizar} \quad & \sum_{M \in P} c_M Y_M & & \\
\text{sujeto a} \quad & Y_v \leq Y_s & \forall v \in V,\ \forall s \in S_G^+(v), & \\
& Y_v \geq 1 + \sum_{s \in S_G^+(v)} (Y_s - 1) & \forall v \in V, & \qquad (MILP_G) \\
& Y_v \geq 0 & \forall v \in V,\ v \text{ no terminal}, & \\
& Y_v \in \{0, 1\} & \forall v \in V,\ v \text{ terminal}. &
\end{aligned}$$

Ejemplo 2.13. En nuestro caso, el Ejemplo 2.3 se convierte en:

$$
\begin{cases}
\underset{\boldsymbol{y},\,\boldsymbol{Y}}{\text{minimizar}} & Y_{\{1,2,3\}} - Y_{\{1,3,4\}} + 3Y_{\{3,4,5\}} \\
\text{sujeto a} & Y_{\{1,2,3\}} \leq y_1, \quad Y_{\{1,2,3\}} \leq y_2, \quad Y_{\{1,2,3\}} \leq y_3, \\
& Y_{\{1,2,3\}} \geq y_1 + y_2 + y_3 - 2, \\
& Y_{\{1,3,4\}} \leq y_1, \quad Y_{\{1,3,4\}} \leq Y_{\{3,4\}}, \\
& Y_{\{1,3,4\}} \geq y_1 + Y_{\{3,4\}} - 1, \\
& Y_{\{3,4,5\}} \leq Y_{\{3,4\}}, \quad Y_{\{3,4,5\}} \leq y_5, \\
& Y_{\{3,4,5\}} \geq Y_{\{3,4\}} + y_5 - 1, \\
& Y_{\{3,4\}} \leq y_3, \quad Y_{\{3,4\}} \leq y_4, \\
& Y_{\{3,4\}} \geq y_3 + y_4 - 1, \\
& Y_{\{1,2,3\}},\, Y_{\{1,3,4\}},\, Y_{\{3,4,5\}},\, Y_{\{3,4\}} \geq 0, \\
& \boldsymbol{y} \in \{0,1\}^5.
\end{cases}
$$

El ejemplo anterior ilustra que un mismo monomio puede linealizarse de distintas formas dependiendo del patrón escogido. La elección del patrón determina qué productos intermedios se introducen como variables auxiliares y, por tanto, afecta tanto al número de variables y restricciones de la reformulación como al ajuste de su relajación continua. El resultado a continuación nos permite afirmar que ambos problemas son equivalentes.

Proposición 2.14. *Dado un polinomio P y un grafo de linealización asociado G, los problemas BPO y $MILP_G$ son equivalentes.*

Demostración. Dada una solución $\bar{\boldsymbol{y}}$ de BPO, se puede dar una solución $\bar{\boldsymbol{Y}}$ de $MILP_G$ fijando $\bar{Y}_{\{i\}} = \bar{y}_i$ para cada i. De esta forma, el resto de variables $\bar{Y}_v$ ya queda fijado gracias a las restricciones que se añaden en un patrón de linealización. Con lo cual, se tiene que para cada monomio $M \in P$, se verifica $\bar{Y}_M = \prod_{i \in M} \bar{y}_i$. Por tanto, para cada solución de BPO, existe una solución de $MILP_G$ que toma el mismo valor.

En el sentido contrario la implicación es evidente, ya que para cualquier $\bar{\boldsymbol{Y}}$ solución de $MILP_G$, podemos construir una solución $\bar{\boldsymbol{y}}$ de BPO simplemente fijando $\bar{y}_i = \bar{Y}_{\{i\}}$ para cada $i \in N$. □

Una vez establecida la equivalencia entre el problema original y su reformulación, la cuestión central pasa a ser qué patrón de linealización conviene utilizar. Distintos patrones pueden representar el mismo producto final, pero inducir relajaciones de distinta calidad y con tamaños diferentes. Las estrategias MaxBound y ND-MinVar presentadas en Elloumi y Verchère (2023) surgen precisamente para formalizar estos dos objetivos: mejorar la cota proporcionada por la relajación o reducir el número de variables auxiliares. Sin embargo, debido a que en ambos casos es necesario resolver un problema de optimización bastante complicado para la limitada ganancia que proporcionan, no profundizaremos más en ellos.

A continuación presentamos un par de resultados recogidos en Elloumi y Verchère (2023) en lo referente a cuán ajustadas son las distintas reformulaciones, pero omitimos su demostración en favor de su extensión a la versión para problemas polinómicos generales.

Teorema 2.15 (Dominancia de los patrones de linealización simples)**.** *Sea P un polinomio de BPO. Sea $G = (V, A)$ y $\tilde{G} = (\tilde{V}, \tilde{A})$ dos patrones de linealización de P, donde cada patrón de linealización es simple. Supongamos que existe un único monomio M_0 de P tal que los grafos G y $\tilde{G}$ son iguales salvo por el grafo que se produce a partir de M_0. En particular, supongamos que $\tilde{G}_{M_0}$ contiene un único*

vértice más que G_{M_0}, denotado por z, y cuyo predecesor denotamos por $\hat{v}$. Entonces, denotando por RLP_G a la relajación continua de $MILP_G$, se verifica que $v[RLP_{\tilde{G}}] \geq v[RLP_G]$; es decir, es al menos tan ajustada.

Corolario 2.16 (Dominancia de los patrones de linealización sobre la linealización estándar)**.** *Cualquier patrón de linealización es al menos tan ajustado como la linealización estándar.*

El siguiente resultado nos permite afirmar que basta explorar un subconjunto de los patrones de linealización simples para encontrar el que se obtiene mediante el método MaxBound.

Corolario 2.17 (Reducción del conjunto de patrones de linealización simples para maximizar las cotas inferiores)**.** *Sea P un polinomio de BPO. Entonces, para cualquier monomio $M \in P$, basta considerar los patrones de linealización $G_M = (V_M, A_M)$ tales que la ramificación en cada nodo no terminal tenga 2 elementos.*

Demostración. Consideremos un polinomio P y un patrón de linealización $G = \mathcal{C}_{M\in P}(G_M)$, donde $G_M = (V_M, A_M)$ es un patrón de linealización simple para cada monomio M. Supongamos que existe $M_0 \in P$ tal que un nodo no terminal v tiene más de 2 sucesores; es decir, $|S^+(v)| \geq 3$. Entonces, se puede construir una nueva linealización simple G'_{M_0}, que determinará una nueva linealización simple de P, $G' = \mathcal{C}(\mathcal{C}_{M\in P\setminus\{M_0\}}(G_M), G'_{M_0})$, introduciendo un nuevo vértice correspondiente a la unión de dos sucesores de v. Repitiendo dicho proceso de forma iterativa, se termina llegando a que $S^+(v) = 2$, y en virtud del Teorema 2.15, dicha linealización es al menos tan ajustada como la original. □

Sin embargo, tengamos presente que pese a la mejora del ajuste, en general MaxBound es menos ajustada que la versión base de RAPOSa. Un ejemplo claro de esto lo tenemos en el problema de ejemplo presente en Elloumi y Verchère (2023):

$$\begin{cases} \min\limits_{\boldsymbol{y}} & f(\boldsymbol{y}) = 5{,}35\, y_2y_4 + 9{,}31\, y_1y_2y_3y_5 - 6{,}54\, y_1y_2y_4y_5 + 9{,}97\, y_1y_3y_4y_5 - 1{,}99\, y_2y_3y_4y_5 \\ \text{sujeto a} & \boldsymbol{y} \in \{0,1\}^5 \end{cases}$$

Mientras que el valor de la función objetivo en la solución es $-1{,}19$, la linealización estándar ofrece una cota inferior de $-4{,}265$, MaxBound ofrece una cota inferior de $-1{,}723$, y la versión base de RAPOSa devuelve directamente $-1{,}19$, resolviendo así el problema en una única iteración.

Centrándonos en las distintas formulaciones implementadas en RAPOSa, se dispone de las siguientes estrategias de linealización de variables binarias.

- **Linealización estándar**: la más fácil de implementar, a costa de ser la menos ajustada de todas.
- **Patrones de linealización**: para cada monomio se construye un patrón de linealización. Sin embargo, puesto que no están implementados MaxBound y ND-MinVar dado que el tiempo de resolución del problema de optimización auxiliar se considera demasiado costoso para la mejora que proporcionan, las formas de construcción implementadas son las siguientes.
  - **Patrón no dominado canónico**: para cada monomio, se ramifica dividiéndolo a la mitad (en caso de un número impar de variables, la primera parte tiene una variable más).

    Además, están implementadas dos heurísticas cuyo fin es minimizar el número de variables adicionales (como alternativa a ND-MinVar) a la vez que se busca mejorar la cota inferior de la anterior estrategia.
  - **Heurística 1**: se ordenan los monomios de menor a mayor número de variables, y para cada uno se busca si alguno de sus divisores ha sido construido. En caso afirmativo, se ramifica en dos haciendo que una parte sea dicho divisor; en caso contrario, se aplica el patrón no dominado canónico anteriormente descrito (dividiendo el monomio a la mitad).

- **Heurística Lucía**: consiste en seguir el razonamiento análogo a la anterior heurística salvo que, en caso de que ninguno de sus divisores haya sido construido todavía, se selecciona el divisor de grado mayor o igual que dos más frecuente. Además, a la vista de que:

$$J_1 \subseteq J_2 \quad \Rightarrow \quad \text{freq}(J_1) \geq \text{freq}(J_2),$$

en caso de empate se selecciona el divisor de mayor grado. Con este enfoque se persigue favorecer la reutilización de subproductos frecuentes en lugar de realizar particiones equilibradas del monomio.

Los resultados anteriores se han presentado en el contexto de problemas formados únicamente por variables binarias. Sin embargo, los problemas que motivan este trabajo contienen también variables no binarias, por lo que los productos que aparecen en la función objetivo y en las restricciones pueden mezclar ambos tipos de variables. En lo que sigue, extenderemos la idea de patrón de linealización a este contexto, aplicándola únicamente a la parte binaria de cada monomio. Los resultados que se presentan hasta el final del capítulo fueron probados en su totalidad por el grupo de trabajo de RAPOSa con el que colaboré durante las prácticas.

2.2. Reformulación de problemas de optimización polinómica generales

En lo que sigue, extendemos las ideas anteriores a problemas con variables binarias y no binarias. Para acompañar esta sección, emplearemos el Ejemplo 2.18.

Ejemplo 2.18. El siguiente es un problema de optimización polinómica con variables enteras.

$$\begin{aligned} \text{minimizar} \quad & x_1x_2y_3 - 5x_1x_2^2 + 3x_2y_3y_4 - x_2y_4 \\ \text{sujeto a} \quad & 2y_3y_4 - y_4 \geq 1, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in [1,2] \times [3,4] \times \{0,1\}^2. \end{aligned} \tag{2.1}$$

Si aplicamos al problema la linealización vista en el capítulo anterior y únicamente construimos las restricciones bound factor para los J-sets, el problema se reformula como:

$$\begin{aligned} \text{minimizar} \quad & X_{\{1,2,3\}} - 5X_{\{1,2,2\}} + 3X_{\{2,3,4\}} - X_{\{2,4\}} \\ \text{sujeto a} \quad & 2X_{\{3,4\}} - y_4 \geq 1, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in [1,2] \times [3,4] \times \{0,1\}^2, \end{aligned} \tag{2.2}$$

donde además hay que añadir las restricciones bound factor asociadas a los monomios $\{1,2,3\}$, $\{1,2,2\}$

y $\{2,3,4\}$ (las de $\{2,4\}$ no son necesarias al ser un monomio contenido en este último):

$$\begin{cases}
-X_{\{1,2,3\}} + X_{\{1,2\}} + 4X_{\{1,3\}} - 4x_1 + 2X_{\{2,3\}} - 2x_2 - 8y_3 + 8 \geq 0,\\
X_{\{1,2,3\}} - 4X_{\{1,3\}} - 2X_{\{2,3\}} + 8y_3 \geq 0,\\
X_{\{1,2,3\}} - X_{\{1,2\}} - 3X_{\{1,3\}} + 3x_1 - 2X_{\{2,3\}} + 2x_2 + 6y_3 - 6 \geq 0,\\
-X_{\{1,2,3\}} + 3X_{\{1,3\}} + 2X_{\{2,3\}} - 6y_3 \geq 0,\\
X_{\{1,2,3\}} - X_{\{1,2\}} - 4X_{\{1,3\}} + 4x_1 - X_{\{2,3\}} + x_2 + 4y_3 - 4 \geq 0,\\
-X_{\{1,2,3\}} + 4X_{\{1,3\}} + X_{\{2,3\}} - 4y_3 \geq 0,\\
-X_{\{1,2,3\}} + X_{\{1,2\}} + 3X_{\{1,3\}} - 3x_1 + X_{\{2,3\}} - x_2 - 3y_3 + 3 \geq 0,\\
X_{\{1,2,3\}} - 3X_{\{1,3\}} - X_{\{2,3\}} + 3y_3 \geq 0,\\
\\
-X_{\{1,2,2\}} + 8X_{\{1,2\}} + 2X_{\{2,2\}} - 16x_1 - 16x_2 + 32 \geq 0,\\
X_{\{1,2,2\}} - 7X_{\{1,2\}} - 2X_{\{2,2\}} + 12x_1 + 14x_2 - 24 \geq 0,\\
-X_{\{1,2,2\}} + 6X_{\{1,2\}} + 2X_{\{2,2\}} - 9x_1 - 12x_2 + 18 \geq 0,\\
X_{\{1,2,2\}} - 8X_{\{1,2\}} - X_{\{2,2\}} + 16x_1 + 8x_2 - 16 \geq 0,\\
-X_{\{1,2,2\}} + 7X_{\{1,2\}} + X_{\{2,2\}} - 12x_1 - 7x_2 + 12 \geq 0,\\
X_{\{1,2,2\}} - 6X_{\{1,2\}} - X_{\{2,2\}} + 9x_1 + 6x_2 - 9 \geq 0,\\
\\
-X_{\{2,3,4\}} + X_{\{2,3\}} + X_{\{2,4\}} - x_2 + 4X_{\{3,4\}} - 4y_3 - 4y_4 + 4 \geq 0,\\
X_{\{2,3,4\}} - X_{\{2,4\}} - 4X_{\{3,4\}} + 4y_4 \geq 0,\\
X_{\{2,3,4\}} - X_{\{2,3\}} - 4X_{\{3,4\}} + 4y_3 \geq 0,\\
-X_{\{2,3,4\}} + 4X_{\{3,4\}} \geq 0,\\
X_{\{2,3,4\}} - X_{\{2,3\}} - X_{\{2,4\}} + x_2 - 3X_{\{3,4\}} + 3y_3 + 3y_4 - 3 \geq 0,\\
-X_{\{2,3,4\}} + X_{\{2,4\}} + 3X_{\{3,4\}} - 3y_4 \geq 0,\\
-X_{\{2,3,4\}} + X_{\{2,3\}} + 3X_{\{3,4\}} - 3y_3 \geq 0,\\
X_{\{2,3,4\}} - 3X_{\{3,4\}} \geq 0.
\end{cases}$$

Definición 2.19 (Patrón de linealización de un problema de optimización polinómico general)**.** Dado un problema de optimización polinómica $MIPP^{\Omega}$, su patrón de linealización se define como la concatenación de patrones de linealización asociados a la parte binaria de los monomios de la función objetivo y las restricciones.

Teniendo en cuenta que a las variables binarias que aparecen solas no se les aplica un patrón de linealización, los monomios de interés serán los de la forma:

$$M(\boldsymbol{x}) = \Big(\prod_{j\in M_B} x_j\Big)\Big(\prod_{j\in M_{NB}} x_j^{\beta_j}\Big),$$

con $M_B \subset N_B,\ |M_B| \geq 2,\ M_{NB} \subset N_{NB},\ \beta_j \in \mathbb{N}\cup\{0\}$. De esta forma, aplicando la linealización de binarias vista en Elloumi y Verchère (2023), definimos la reducción binaria del monomio como:

$$[M(\boldsymbol{x})]_{\text{BR}} = X_{M_B}\prod_{j\in M_{NB}} x_j^{\beta_j}.$$

Así, dado un problema $MIPP^{\Omega}$ y un patrón de linealización $G=(V,A)$, la reformulación de $MIPP^{\Omega}$ asociada a G se define como

$$\begin{array}{lll}
\text{minimizar} & [\phi_0(\boldsymbol{x})]_{\text{BR}} & \\
\text{sujeto a} & [\phi_r(\boldsymbol{x})]_{\text{BR}} \geq \beta_r, & r=1,\dots,R_1,\\
& [\phi_r(\boldsymbol{x})]_{\text{BR}} = \beta_r, & r=R_1+1,\dots,R,\\
& X_v \leq X_s, & \forall v\in V,\ \forall s\in S_G^+(v)\\
& X_v \geq 1 + \displaystyle\sum_{s\in S_G^+(v)} (X_s-1), & \forall v\in V,\ v \text{ no terminal},\\
& 0\leq X_v, & \forall v\in V,\\
& \boldsymbol{x}\in\Omega. &
\end{array} \qquad (RMIPP_G^{\Omega})$$

Ejemplo 2.20. Retomando el problema del Ejemplo 2.18, nótese que como la parte binaria tiene grado 2, cualquier patrón de linealización simple será exactamente igual que la linealización estándar, así que representamos dicha linealización.

$$\begin{aligned} \text{minimizar} \quad & x_1x_2y_3 - 5x_1x_2^2 + 3x_2Y_{\{3,4\}} - x_2y_4 \\ \text{sujeto a} \quad & 2Y_{\{3,4\}} - y_4 \geq 1, \\ & Y_{\{3,4\}} \leq y_3, \\ & Y_{\{3,4\}} \leq y_4, \\ & Y_{\{3,4\}} \geq y_3 + y_4 - 1, \\ & Y_{\{3,4\}} \geq 0, \\ & (\boldsymbol{x}, \boldsymbol{y}) \in [1,2] \times [3,4] \times \{0,1\}^2. \end{aligned} \tag{2.3}$$

A continuación, presentamos la extensión de la Proposición 2.32 para problemas polinómicos.

Lema 2.21. *Los problemas $MIPP^{\Omega}$ y $RMIPP_G^{\Omega}$ son equivalentes.*

Demostración. Hay que probar la equivalencia en ambos sentidos.

- Partiendo de $\tilde{\boldsymbol{x}}$ una solución factible de $MIPP^{\Omega}$, podemos construir una solución factible $\widetilde{\mathbf{X}}$ de $RMIPP_G^{\Omega}$ fijando $\widetilde{X}_{\{j\}} := \tilde{x}_j,\ \forall j \in \{1, \ldots, |N|\}$ y $\widetilde{X}_v := \prod_{j \in v} \tilde{x}_j$ para todo $v \in V$ no terminal.

  Por definición, las restricciones de la forma $[\phi_r(\boldsymbol{x})]_{\text{BR}} \geq \beta_r$, $r = 1, \ldots, R_1$ y $[\phi_r(\boldsymbol{x})]_{\text{BR}} = \beta_r$, $r = R_1 + 1, \ldots, R$ se satisfacen, y la función objetivo alcanza el mismo valor que $MIPP^{\Omega}$. Por tanto, únicamente necesitamos comprobar las restricciones de linealización. Dado que $\tilde{\boldsymbol{x}}$ es factible en $MIPP^{\Omega}$, para cada $j \in N_B$, $\tilde{x}_j \in \{0,1\}$. Entonces, para cada $v \in V$, se verifica que $\widetilde{X}_v \in \{0,1\}$. Además, por la construcción de los patrones de linealización, cada sucesor $s \in S_G^+(v)$ se corresponde con un subconjunto de índices de v. Distinguimos los siguientes casos.

  - Si $\widetilde{X}_v = 0$, entonces existe $s \in S_G^+(v)$ tal que $\widetilde{X}_s = 0$, luego se verifican todas las condiciones.
  - Si $\widetilde{X}_v = 1$, entonces para cada $s \in S_G^+(v)$, $\widetilde{X}_s = 1$, así que se verifican todas las condiciones.

- Partiendo de $\widetilde{\mathbf{X}}$ solución factible de $RMIPP_G^{\Omega}$, podemos construir una solución factible $\tilde{\boldsymbol{x}}$ de $MIPP^{\Omega}$. Para ello, definimos $\tilde{x}_j := \widetilde{X}_{\{j\}}$, $\forall j \in \{1, \ldots, |N|\}$. Veamos que dicho punto es una solución factible de $MIPP^{\Omega}$ con el mismo valor de la función objetivo. La factibilidad en $RMIPP_G^{\Omega}$ implica que, para cada variable binaria terminal j, $\widetilde{X}_{\{j\}} \in \{0,1\}$. Sea v un nodo no terminal y consideremos los siguientes casos:

  - Para cada $j \in v$, $\widetilde{X}_{\{j\}} = 1$. Entonces, para cada $s \in S_G^+(v)$, $\widetilde{X}_s = 1$. Por las restricciones $\widetilde{X}_v \leq \widetilde{X}_s$ y $\widetilde{X}_v \geq 1 + \sum_{s \in S_G^+(v)} (\widetilde{X}_s - 1)$, podemos deducir que $\widetilde{X}_v = 1 = \prod_{j \in v} \widetilde{X}_{\{j\}}$.
  - Existe un $j \in v$ tal que $\widetilde{X}_{\{j\}} = 0$. Entonces, existe $s \in S_G^+(v)$ con $\widetilde{X}_s = 0$. Por las restricciones $\widetilde{X}_v \leq \widetilde{X}_s$ y $\widetilde{X}_v \geq 0$ se sigue que $\widetilde{X}_v = 0 = \prod_{j \in v} \widetilde{X}_{\{j\}}$.

  De esta forma, para cada nodo no terminal $v \in V$ se verifica que $\widetilde{X}_v = \prod_{j \in v} \tilde{x}_j$, luego $\tilde{\boldsymbol{x}}$ verifica todas las restricciones de $MIPP^{\Omega}$ y la función objetivo alcanza el mismo valor. □

Para resolver con nuestro algoritmo dicho problema, todavía es necesario sustituir por variables RLT y añadir las restricciones bound factor. De esta forma, los problemas sobre los que trabajaremos en esta sección serán de la forma

$$
\begin{array}{rll}
\text{minimizar} & [[\phi_0(\boldsymbol{x})]_{\text{BR}}]_L & \\
\text{sujeto a} & [[\phi_r(\boldsymbol{x})]_{\text{BR}}]_L \geq \beta_r, & r = 1, \ldots, R_1, \\
& [[\phi_r(\boldsymbol{x})]_{\text{BR}}]_L = \beta_r, & r = R_1 + 1, \ldots, R, \\
& X_v \leq X_s, & \forall v \in V,\ \forall s \in S_G^+(v) \\
& X_v \geq 1 + \displaystyle\sum_{s \in S_G^+(v)} (X_s - 1), & \forall v \in V,\ v \text{ no terminal}, \\
& 0 \leq X_v, & \forall v \in V, \\
& \left[\displaystyle\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \right]_L \geq 0, & \forall J_1 \cup J_2 = J, \\
& & J \text{ monomio del problema} \\
& & \quad \text{reformulado} \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^{|N|}. &
\end{array}
\qquad (RLT(RMIPP_G^\Omega))
$$

Nótese que los monomios para los que se construyen las restricciones bound factor son diferentes. Esto lo podemos ver claramente linealizando el problema del Ejemplo 2.18, y tendrá importancia posteriormente cuando discutamos lo ajustadas que son las formulaciones.

Ejemplo 2.22. Puesto que ahora $Y_{\{3,4\}}$ designa una nueva variable, para distinguirla de la variable que sustituye $y_3 y_4$ en la linealización original, cuando aparezca formando parte de un producto, en el subíndice aparecerá entre paréntesis. De esta forma, nuestro problema queda linealizado como sigue:

$$
\begin{array}{rl}
\text{minimizar} & X_{\{1,2,3\}} - 5X_{\{1,2,2\}} + 3X_{\{2,(3,4)\}} - X_{\{2,4\}} \\
\text{sujeto a} & 2Y_{\{3,4\}} - y_4 \geq 1, \\
& Y_{\{3,4\}} \leq y_3, \\
& Y_{\{3,4\}} \leq y_4, \\
& Y_{\{3,4\}} \geq y_3 + y_4 - 1, \\
& Y_{\{3,4\}} \geq 0, \\
& (\boldsymbol{x}, \boldsymbol{y}) \in [1,2] \times [3,4] \times \{0,1\}^2,
\end{array}
\qquad (2.4)
$$

donde además hay que añadir las restricciones bound factor asociadas a los monomios $\{1,2,3\}$, $\{1,2,2\}$, $\{2,(3,4)\}$ y $\{2,4\}$ (este monomio antes no era necesario, pero al linealizar $y_3 y_4$, ahora ya no está claro

que sus restricciones bound factor estén implicadas por las del monomio $\{2,(3,4)\}$):

$$\begin{cases}
-X_{\{1,2,3\}}+X_{\{1,2\}}+4X_{\{1,3\}}-4x_1+2X_{\{2,3\}}-2x_2-8y_3+8\geq 0,\\
X_{\{1,2,3\}}-4X_{\{1,3\}}-2X_{\{2,3\}}+8y_3\geq 0,\\
X_{\{1,2,3\}}-X_{\{1,2\}}-3X_{\{1,3\}}+3x_1-2X_{\{2,3\}}+2x_2+6y_3-6\geq 0,\\
-X_{\{1,2,3\}}+3X_{\{1,3\}}+2X_{\{2,3\}}-6y_3\geq 0,\\
X_{\{1,2,3\}}-X_{\{1,2\}}-4X_{\{1,3\}}+4x_1-X_{\{2,3\}}+x_2+4y_3-4\geq 0,\\
-X_{\{1,2,3\}}+4X_{\{1,3\}}+X_{\{2,3\}}-4y_3\geq 0,\\
-X_{\{1,2,3\}}+X_{\{1,2\}}+3X_{\{1,3\}}-3x_1+X_{\{2,3\}}-x_2-3y_3+3\geq 0,\\
X_{\{1,2,3\}}-3X_{\{1,3\}}-X_{\{2,3\}}+3y_3\geq 0,\\
\\
-X_{\{1,2,2\}}+8X_{\{1,2\}}+2X_{\{2,2\}}-16x_1-16x_2+32\geq 0,\\
X_{\{1,2,2\}}-7X_{\{1,2\}}-2X_{\{2,2\}}+12x_1+14x_2-24\geq 0,\\
-X_{\{1,2,2\}}+6X_{\{1,2\}}+2X_{\{2,2\}}-9x_1-12x_2+18\geq 0,\\
X_{\{1,2,2\}}-8X_{\{1,2\}}-X_{\{2,2\}}+16x_1+8x_2-16\geq 0,\\
-X_{\{1,2,2\}}+7X_{\{1,2\}}+X_{\{2,2\}}-12x_1-7x_2+12\geq 0,\\
X_{\{1,2,2\}}-6X_{\{1,2\}}-X_{\{2,2\}}+9x_1+6x_2-9\geq 0,\\
\\
X_{\{2,(3,4)\}}-3Y_{\{3,4\}}\geq 0,\\
-X_{\{2,(3,4)\}}+x_2+3Y_{\{3,4\}}-3\geq 0,\\
-X_{\{2,(3,4)\}}+4Y_{\{3,4\}}\geq 0,\\
X_{\{2,(3,4)\}}-x_2-4Y_{\{3,4\}}+4\geq 0,\\
X_{\{2,4\}}-3y_4\geq 0,\\
-X_{\{2,4\}}+x_2+3y_4-3\geq 0,\\
-X_{\{2,4\}}+4y_4\geq 0,\\
X_{\{2,4\}}-x_2-4y_4+4\geq 0.
\end{cases}$$

Nótese que reducir toda la parte binaria de un monomio a una variable de grado único reduce el grado del problema, lo cual se considera positivo. No obstante, el ejemplo anterior también ilustra cómo esto puede hacer que se pierda la relación de divisibilidad, lo que afecta a la construcción de las restricciones bound factor de los J-sets, obligando a añadir nuevos monomios. En el Capítulo 4 presentaremos una reformulación que no aumenta la cantidad de monomios sobre los cuales construir las restricciones bound factor al considerar únicamente la parte no binaria del monomio.

Una vez definida la reformulación para problemas generales, el siguiente paso consiste en comparar el ajuste de las relajaciones que se obtienen. Comenzaremos por las relajaciones continuas, ya que son las que aparecen al resolver los problemas lineales sin imponer integralidad sobre las variables binarias o auxiliares. Dicha relajación no sólo tiene impacto sobre las configuraciones donde RAPOSa gestiona las violaciones de integralidad, sino que también tiene impacto sobre el tiempo lineal cuando el solver auxiliar trabaja con las relajaciones enteras del problema.

2.2.1. Comparación del ajuste de las relajaciones continuas

Empezamos generalizando los resultados de Elloumi y Verchère (2023) para la reformulación de binarias en problemas polinómicos generales. La totalidad de los resultados de esta sección fueron probados por el grupo de trabajo al que me incorporé para hacer las prácticas.

Teorema 2.23 (Adaptación del Teorema 2.15)**.** *Sea P un polinomio de $MIPP^{\Omega}$. Sea $G=(V,A)$ y $\tilde{G}=(\tilde{V},\tilde{A})$ dos patrones de linealización de P, donde los patrones de linealización para cada monomio son simples. Supongamos que existe un único monomio M_0 de P tal que los grafos G y $\tilde{G}$ son iguales salvo por el grafo que se produce a partir de M_0. En particular, supongamos que $\tilde{G}_{M_0}$ contiene un único vértice más que G_{M_0}. Entonces, denotando por RLP_G a la relajación continua de $RMIPP^{\Omega}_G$, se verifica que $v[RLP_{\tilde{G}}]\geq v[RLP_G]$; es decir, es al menos igual de ajustada.*

Demostración. Partamos definiendo la notación que vamos a emplear. Sean $\mathcal{B}$ y $\tilde{\mathcal{B}}$ las regiones factibles de RLP_G y $RLP_{\tilde{G}}$ respectivamente. Sea $\mathbf{X}\in\tilde{\mathcal{B}}$ una solución factible de $RLP_{\tilde{G}}$. Esto quiere decir que

$(X_v)_{v\in\tilde{V}}$ verifica las siguientes restricciones:

$$(\widetilde{\mathcal{B}})\quad \begin{cases} X_v \le X_w, & \forall v\in\widetilde{V},\ \forall w\in S_{\widetilde{G}}(v),\\ X_v \ge 1+\sum\limits_{w\in S_{\widetilde{G}}(v)}(X_w-1), & \forall v\in\widetilde{V},\ |v|\ge 2,\\ 0\le X_v\le 1, & \forall v\in\widetilde{V}\end{cases}$$

Entonces tenemos que probar que las restricciones de $\mathbf{X}$ asociadas a $v\in V$, es decir, la proyección de $\mathbf{X}$ sobre las variables asociadas a V pertenece a $\mathcal{B}$. Los conjuntos $\mathcal{B}$ y $\tilde{\mathcal{B}}$ únicamente difieren en cómo se linealiza el monomio M_0, donde $\widetilde{V}_{M_0}=V_{M_0}\cup\{z\}$. Esto implica que $|z|\ge 2$, ya que en caso contrario G_{M_0} y $\widetilde{G}_{M_0}$ se linealizarían en dos monomios distintos. Nos centramos en las restricciones asociadas a M_0 destacando en las que aparezca z:

$$(\widetilde{\mathcal{B}}_{M_0})\quad \begin{cases} X_v \le X_w, & \forall v\in V_{M_0}\backslash\{\hat{v}\},\ \forall w\in S_{\widetilde{G}}(v), & (a)\\ X_{\hat{v}} \le X_w, & \forall w\in S_{\widetilde{G}}(\hat{v})\backslash\{z\}, & (b)\\ X_{\hat{v}} \le X_z, & & (c)\\ X_v \ge 1+\sum\limits_{w\in S_{\widetilde{G}}(v)}(X_w-1), & \forall v\in V_{M_0}\backslash\{\hat{v}\},\ |v|\ge 2, & (d)\\ X_{\hat{v}} \ge 1+\sum\limits_{w\in S_{\widetilde{G}}(\hat{v})\backslash\{z\}}(X_w-1)+(X_z-1), & & (e)\\ X_z \le X_w, & \forall w\in S_{\widetilde{G}}(z), & (f)\\ X_z \ge 1+\sum\limits_{w\in S_{\widetilde{G}}(z)}(X_w-1), & & (g)\\ 0\le X_v\le 1, & \forall v\in V_{M_0}\backslash\{z\}, & (h)\\ 0\le X_z\le 1. & & (i)\end{cases} \tag{2.5}$$

Necesitamos probar que $(X_v)_{v\in V}$ también verifica las restricciones originales para el monomio M_0:

$$(\mathcal{B}_{M_0})\quad \begin{cases} X_v \le X_w, & \forall v\in V_{M_0},\ \forall w\in S_G^+(v), & (a)\\ X_v \ge 1+\sum\limits_{w\in S_G^+(v)}(X_w-1), & \forall v\in V_{M_0},\ |v|\ge 2, & (b)\\ 0\le X_v\le 1, & \forall v\in V_{M_0}. & (c)\end{cases} \tag{2.6}$$

Para verificar el resultado tenemos que combinar desigualdades de $\widetilde{\mathcal{B}}_{M_0}$ que nos permitan deducir las de $\mathcal{B}_{M_0}$. Tendremos que centrarnos en las restricciones donde aparezca z o su predecesor en $\widetilde{G}$ ya que son las únicas diferencias entre $\mathcal{B}_{M_0}$ y $\widetilde{\mathcal{B}}_{M_0}$.

Paso 1: Comprobación de 2.6(a). De 2.5(c) y 2.5(f) se sigue que

$$X_{\hat{v}} \le X_w,\quad \forall w\in S_{\widetilde{G}}(z).$$

Unido a 2.5(b), se tiene que

$$X_{\hat{v}} \le X_w,\quad \forall w\in \big(S_{\widetilde{G}}(\hat{v})\backslash\{z\}\big)\cup S_{\widetilde{G}}(z).$$

En virtud de las definiciones de z y $\hat{v}$, podemos afirmar que

$$\big(S_{\widetilde{G}}(\hat{v})\backslash\{z\}\big)\cup S_{\widetilde{G}}(z)=S_G(\hat{v}),$$

y para todo $v\in V_{M_0}\backslash\{\hat{v}\}$,

$$S_{\widetilde{G}}(v)=S_G^+(v).$$

De esta forma, se recupera 2.6(a).

Paso 2: Comprobación de 2.6(b). De 2.5(e) y 2.5(g) se sigue que

$$X_{\hat{v}} \geq 1 + \sum_{w \in S_{\widetilde{G}}(\hat{v}) \setminus \{z\}} (X_w - 1) + \sum_{w \in S_{\widetilde{G}}(z)} (X_w - 1).$$

Haciendo uso nuevamente de

$$\left(S_{\widetilde{G}}(\hat{v}) \setminus \{z\}\right) \cup S_{\widetilde{G}}(z) = S_G(\hat{v}),$$

obtenemos 2.6(b) para $\hat{v}$. Para el resto de nodos $v \in V_{M_0} \setminus \{\hat{v}\}$, 2.5(d) proporciona directamente 2.6(b) al verificarse $S_{\widetilde{G}}(v) = S_G^+(v)$.

Paso 3: Comprobación de 2.6(c). Es inmediato a partir de 2.5(h).

De este modo, podemos concluir que la restricción de $\mathbf{X}$ a V pertenece a $\mathcal{B}_{M_0}$. Puesto que el resto de restricciones que definen $\mathcal{B}$ y $\widetilde{\mathcal{B}}$ coinciden, se deduce que, para toda solución $\mathbf{X} \in \widetilde{\mathcal{B}}$, su restricción a los nodos de G pertenece a $\mathcal{B}$. □

Corolario 2.24 (Adaptación del Corolario 2.16)**.** *Sea P un polinomio de $MIPP^{\Omega}$. Sea $G = (V, A)$ la linealización estándar de P y $\tilde{G} = (\tilde{V}, \tilde{A})$ un patrón de linealización de P, donde el patrón de linealización para cada monomio es simple. Entonces, denotando por RLP_G a la relajación continua de $RMIPP^{\Omega}_G$, se verifica que $v[RLP_{\tilde{G}}] \geq v[RLP_G]$; es decir, es al menos igual de ajustada.*

Demostración. Consideremos un polinomio P, el grafo de linealización estándar $G = \bigcup_{M \in P} G_M$, y un grafo de linealización $\widetilde{G} = \bigcup_{M \in P} \widetilde{G}_M$, donde, para cada monomio $M \in P$, $\widetilde{G}_M = (\widetilde{V}_M, \widetilde{A}_M)$ es un patrón de linealización simple. Podemos entonces construir una sucesión de patrones de linealización $\{G_k\}_{1 \leq k \leq K}$ tal que:

- $G_1 = G$,
- $G_K = \widetilde{G}$,
- para cada $k \in \{1, \ldots, K-1\}$, G_k y G_{k+1} verifiquen las hipótesis del Teorema 2.23.

Para construir dicha sucesión, se procede de la forma que sigue: Partimos de $G_1 = G$, y dado $G_k = \bigcup_{M \in P} G_{k,M}$, se selecciona un monomio M_0 tal que $G_{k,M_0} \neq \widetilde{G}_{M_0}$. Así, construimos $G_{k+1} = \bigcup_{M \in P} G_{k+1,M}$ de la forma que sigue:

$$G_{k+1,M} = \begin{cases} G_{k,M}, & M \neq M_0, \\ G_{k,M_0} \text{ añadiendo un nodo en } \widetilde{G}_{M_0} \setminus G_{k,M_0}, & M = M_0. \end{cases}$$

Dicha construcción es siempre posible puesto que para cualquier monomio $M \in P$ la linealización estándar G_M siempre tendrá el menor número de nodos posibles. De esta forma, podemos aplicar el Teorema 2.23 de forma iterativa, lo que significa que:

$$v[RLP_G] = v[RLP_{G_1}] \leq v[RLP_{G_2}] \leq \cdots \leq v[RLP_{G_{K-1}}] \leq v[RLP_{G_K}] = v[RLP_{\widetilde{G}}],$$

probando así el resultado. □

Nótese que los anteriores resultados se refieren a $MIPP^{\Omega}$ y $RMIPP^{\Omega}_G$, mientras que el algoritmo basado en la técnica RLT trabaja con ${}^{\text{CR}}RLT(MIPP^{\Omega})$ y ${}^{\text{CR}}RLT(RMIPP^{\Omega}_G)$ respectivamente. De manera análoga a la anterior sección, en la literatura previa se ha estudiado el impacto de las reformulaciones que reducen el grado para problemas de optimización polinómica continua. Este es el caso de las Proposiciones 1-4 en Dalkiran y Ghalami (2018) y los Teoremas 1-3 en González-Rodríguez et al. (2025). A continuación, presentamos un resultado desarrollado por el equipo de RAPOSa que sirve de bloque central para relacionar el ajuste de las relajaciones de $MIPP^{\Omega}$ y sus reformulaciones.

Supongamos que tenemos el monomio $J = x_1x_2^2x_3x_4^2$ y que queremos estudiar el impacto de reformularlo como $X_{123}x_2x_4^2$, junto con la restricción $X_{123} = x_1x_2x_3$ y las cotas $u_1u_2u_3 \geq X_{123} \geq l_1l_2l_3$. Aunque dicha reformulación daría lugar a un problema de optimización polinómica equivalente, las relajaciones RLT podrían diferir en su ajuste, ya que, en particular, el monomio original tenía grado 6 y los dos nuevos monomios tienen grados 4 y 3, lo que modificaría significativamente las restricciones bound factor. En general, dado $J \subset N^\delta$ y C y $\bar{C}$ tales que $J = C \cup \bar{C}$, se puede reformular $\prod_{j\in J} x_j$ como $X_C \prod_{j\in\bar{C}} x_j$, junto con la restricción $X_C = \prod_{j\in C} x_j$ y las cotas $\prod_{j\in C} u_j \geq X_C \geq \prod_{j\in C} l_j$.

Proposición 2.25 (Ajuste de las reformulaciones de reducción de grado)**.** *Sea $J \in N^\delta$ y sean C y $\bar{C}$ tales que $J = C \cup \bar{C}$, con $|C| \geq 2$ y $|\bar{C}| \geq 1$. Entonces, las restricciones bound factor asociadas a J implican las restricciones bound factor asociadas a la reformulación donde:*

i) Se añade la restricción $X_C = \prod_{j\in C} x_j$.

ii) Se añaden las cotas $\prod_{j\in C} u_j \geq X_C \geq \prod_{j\in C} l_j$.

iii) El monomio J, dado por $\prod_{j\in J} x_j$, se sustituye por $X_C \prod_{j\in\bar{C}} x_j$.

Demostración. Puesto que $C \subset J$, en virtud del Lema 1.16, las restricciones bound factor de J implican las de C, así que i) es inmediato. Centrémonos en las restricciones bound factor asociadas a iii).

- Caso base: $|C| = 2$. Supongamos que C se corresponde con un monomio x_ix_j que es reemplazado por X_{ij}. Empezamos probando que las restricciones bound factor asociadas a dicho monomio implican las restricciones $u_iu_j - X_{ij} \geq 0$ y $X_{ij} - l_il_j \geq 0$.[1] Consideremos las restricciones bound factor asociadas a X_{ij}:

$$\begin{aligned} M_1 &= [(x_i - l_i)(x_j - l_j)]_L = X_{ij} - l_jx_i - l_ix_j + l_il_j \geq 0, \\ M_2 &= [(x_i - l_i)(u_j - x_j)]_L = -X_{ij} + u_jx_i + l_ix_j - l_iu_j \geq 0, \\ M_3 &= [(u_i - x_i)(x_j - l_j)]_L = -X_{ij} + l_jx_i + u_ix_j - u_il_j \geq 0, \quad \text{y} \\ M_4 &= [(u_i - x_i)(u_j - x_j)]_L = X_{ij} - u_jx_i - u_ix_j + u_iu_j \geq 0. \end{aligned}$$

  Nótese que sumar pares de restricciones bound factor nos permite recuperar los factores lineales, escalados por la longitud del intervalo.

$$\begin{aligned} M_1 + M_2 &= (u_j - l_j)(x_i - l_i) \Rightarrow x_i - l_i = \frac{M_1 + M_2}{u_j - l_j}, \\ M_1 + M_3 &= (u_i - l_i)(x_j - l_j) \Rightarrow x_j - l_j = \frac{M_1 + M_3}{u_i - l_i}, \\ M_2 + M_4 &= (u_j - l_j)(u_i - x_i) \Rightarrow u_i - x_i = \frac{M_2 + M_4}{u_j - l_j}, \text{y} \\ M_3 + M_4 &= (u_i - l_i)(u_j - x_j) \Rightarrow u_j - x_j = \frac{M_3 + M_4}{u_i - l_i}. \end{aligned}$$

- $\boldsymbol{X_{ij} - l_il_j \geq 0}$. Aislamos $X_{ij} - l_il_j$ partiendo de M_1:

$$X_{ij} - l_il_j = M_1 + l_j(x_i - l_i) + l_i(x_j - l_j).$$

  Sustituyendo los factores lineales por sus expresiones en términos de M_k:

$$X_{ij} - l_il_j = \left(1 + \frac{l_j}{u_j - l_j} + \frac{l_i}{u_i - l_i}\right) M_1 + \left(\frac{l_j}{u_j - l_j}\right) M_2 + \left(\frac{l_i}{u_i - l_i}\right) M_3.$$

[1] Las cotas de ii) se siguen del Teorema 1 en González-Díaz et al. (2025), pero presentamos aquí un argumento más directo, que sirve como vía para obtener las restricciones bound factor de iii).

- $\boldsymbol{u_i u_j - X_{ij} \geq 0}$. De forma análoga, partiendo de M_2:

$$u_i u_j - X_{ij} = M_2 + u_j(u_i - x_i) + l_i(u_j - x_j).$$

Sustituyendo los factores lineales:

$$u_i u_j - X_{ij} = \left(1 + \frac{u_j}{u_j - l_j}\right) M_2 + \left(\frac{l_i}{u_i - l_i}\right) M_3 + \left(\frac{u_j}{u_j - l_j} + \frac{l_i}{u_i - l_i}\right) M_4.$$

Ahora estamos en condiciones de demostrar que las restricciones bound factor asociadas al monomio $X_C \prod_{j \in \bar{C}} x_j$ están implicadas por las asociadas a J. Consideremos, por ejemplo, una restricción bound factor de la forma

$$\left[(u_i u_j - X_{ij}) \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\right]_L \geq 0,$$

para J_1 y J_2 fijos tales que $J_1 \cup J_2 = \bar{C}$, que tiene grado $\delta - 1$. Entonces, esta restricción puede replicarse haciendo uso de las siguientes bound factor de J:

$$BF_1 = \left[(x_i - l_i)(x_j - l_j) \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\right]_L, \quad \text{con } J_1 \cup J_2 = \bar{C},$$

$$BF_2 = \left[(x_i - l_i)(u_j - x_j) \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\right]_L, \quad \text{con } J_1 \cup J_2 = \bar{C},$$

$$BF_3 = \left[(u_i - x_i)(x_j - l_j) \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\right]_L, \quad \text{con } J_1 \cup J_2 = \bar{C}, \quad \text{y}$$

$$BF_4 = \left[(u_i - x_i)(u_j - x_j) \prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\right]_L, \quad \text{con } J_1 \cup J_2 = \bar{C},$$

todas ellas de grado δ. Basta con combinar estas cuatro restricciones de acuerdo con los coeficientes obtenidos anteriormente para M_1, M_2, M_3 y M_4. La misma lógica se aplica a cualquier otra restricción bound factor asociada al monomio $X_C \prod_{j \in \bar{C}} x_j$.

- **Caso general.** En general, cuando C puede tener un número arbitrario de elementos, se pueden obtener las implicaciones deseadas aplicando recursivamente el caso base mientras se agrupan las variables de dos en dos, generando una sucesión de reformulaciones de ajuste decreciente.

□

Nótese que el anterior resultado no es suficiente para probar la implicación con la linealización, ya que nosotros no disponemos de una restricción de la forma $X_v = \prod_{j \in v} x_j$. Para ello, presentamos el siguiente resultado, que afirma que las restricciones bound factor en $RLT(MIPP^{\Omega})$ implican todas las restricciones de grafo asociadas al patrón de linealización de $RLT(RMIPP^{\Omega}_G)$.

Proposición 2.26 (Patrones de linealización y restricciones bound factor)**.** *Sea $G = (V, A)$ un patrón de linealización de $MIPP^{\Omega}$ y supongamos que para cada $j \in N$, $l_j < u_j$. Entonces, las restricciones bound factor de $RLT(MIPP^{\Omega})$ implican todas las restricciones asociadas al patrón de linealización en $RMIPP^{\Omega}_G$.*

Demostración. En primer lugar, tengamos en cuenta que para cada $v \in V$, la variable X_v también está presente en $RLT(MIPP^{\Omega})$. Consideremos las restricciones bound factor.

$$[F_\delta(J_1, J_2)]_L = \left[\prod_{j\in J_1}(x_j - l_j)\prod_{j\in J_2}(u_j - x_j)\right]_L \geq 0,\ J_1 \cup J_2 \subset N^\delta,\ |J_1 \cup J_2| = \delta.$$

En virtud del Lema 1.16, puesto que para $j \in N$, $l_j < u_j$, las restricciones bound factor de grado δ implican las de cualquier grado menor que δ. De esta forma, podemos restringirnos al conjunto de restricciones bound factor asociadas a monomios únicamente compuestos por binarias. En particular, basta restringirse a los monomios que participen en el patrón de linealización de un monomio de variables binarias del problema.

- Restricciones $X_v \geq 0$: Sea $v \in V$ un nodo no terminal del patrón de linealización. Entonces, con $J_1 = v$, $J_2 = \emptyset$, la restricción asociada a dicha restricción bound factor es $[\prod_{i\in v}(x_i - 0)]_L = X_v \geq 0$.
- Restricciones $X_v \leq X_s$: Sea $v \in V$ y $s \in S_G^+(v)$.
  - Caso base. Supongamos $|v \setminus s| = 1$, donde $v = s \cup \{j\}$ para algún j, entonces se tiene que $[(1 - x_j)\prod_{i\in s} x_i]_L = X_s - X_v \geq 0$, de donde se deduce $X_v \leq X_s$.
  - Caso general. Si $|v \setminus s| = k > 1$, podemos construir una cadena de conjuntos añadiendo sucesivamente los elementos de $v \setminus s = \{j_1, \ldots, j_k\}$:

    $$s \subset (s \cup \{j_1\}) \subset (s \cup \{j_1, j_2\}) \subset \cdots \subset (s \cup \{j_1, \ldots, j_k\}) = v.$$

    Aplicando el caso base de forma iterativa para cada par consecutivo de la sucesión, obtenemos

    $$X_s \geq X_{s\cup\{j_1\}} \geq X_{s\cup\{j_1,j_2\}} \geq \cdots \geq X_{s\cup\{j_1,\ldots,j_k\}} \geq X_v.$$
- Restricciones $X_v \geq 1 + \sum_{s\in S_G^+(v)}(X_s - 1)$. Sea $S_G^+(v) = \{s_1, \ldots, s_n\}$. Entonces, podemos reescribir la restricción como

  $$X_v \geq 1 + \sum_{k=1}^{n}(X_{s_k} - 1), \quad \text{y, equivalentemente,} \quad (n-1) - \sum_{k=1}^{n} X_{s_k} + X_v \geq 0.$$

  Ahora bien, para cada $J \subseteq v$, considérese la restricción bound factor

  $$F(J,\, v \setminus J) = \prod_{i\in J} x_i \prod_{j\in v\setminus J}(1 - x_j) \geq 0.$$

  Denotando por $F^L(J,\, v \setminus J)$ el lado izquierdo de $F(J,\, v \setminus J)$. Al expandir y linealizar $F^L(J,\, v \setminus J)$, se obtiene

  $$[F^L(J,\, v \setminus J)]_L = \sum_{S:\ J\subseteq S\subseteq v} (-1)^{|S|-|J|} X_S.$$

  La demostración se basa en encontrar coeficientes no negativos $\lambda_J \geq 0$ tales que

  $$\sum_{J\subseteq v} \lambda_J\, [F^L(J,\, v \setminus J)]_L = (n-1) - \sum_{k=1}^{n} X_{s_k} + X_v. \tag{2.7}$$

  Al expandir el lado izquierdo obtenemos que, para cada $S \subseteq v$, el coeficiente de X_S es

  $$c(S) = \sum_{J\subseteq S} (-1)^{|S|-|J|}\lambda_J. \tag{2.8}$$

  Por tanto, los coeficientes deben satisfacer:

1. $c(\emptyset) = n - 1$;
2. $c(s_k) = -1$, para cada $k \in \{1, \ldots, n\}$;
3. $c(S) = 0$, para cada $S \subsetneq v,\ S \neq s_k$;
4. $c(v) = 1$.

Ahora, considérese un orden de los subconjuntos $S \subseteq v$ por inclusión. Para cada $S \subseteq v$, el coeficiente de 2.8 involucra únicamente variables λ_J con $J \subseteq S$, y el coeficiente de λ_S es igual a 1. Por lo tanto, el sistema lineal asociado es triangular inferior con entradas diagonales no nulas. A continuación, representamos los coeficientes del sistema para $v = \{1, 2, 3\}$:

	$\lambda_\emptyset$	$\lambda_{\{1\}}$	$\lambda_{\{2\}}$	$\lambda_{\{3\}}$	$\lambda_{\{1,2\}}$	$\lambda_{\{1,3\}}$	$\lambda_{\{2,3\}}$	$\lambda_{\{1,2,3\}}$
$\emptyset$	1	0	0	0	0	0	0	0
$\{1\}$	−1	1	0	0	0	0	0	0
$\{2\}$	−1	0	1	0	0	0	0	0
$\{3\}$	−1	0	0	1	0	0	0	0
$\{1,2\}$	1	−1	−1	0	1	0	0	0
$\{1,3\}$	1	−1	0	−1	0	1	0	0
$\{2,3\}$	1	0	−1	−1	0	0	1	0
$\{1,2,3\}$	−1	1	1	1	−1	−1	−1	1

Este sistema es, por tanto, compatible y admite una única solución para los coeficientes $\{\lambda_J\}_{J \subseteq v}$. Queda por demostrar que dicha solución satisface que, para cada $J \subseteq v$, $\lambda_J \geq 0$. Una vez establecido esto, se obtiene la representación deseada como combinación lineal no negativa de restricciones bound factor, terminando así la demostración.

Para cada $J \subseteq v$, definamos $\boldsymbol{x}^J \in \{0,1\}^v$ como

$$(\boldsymbol{x}^J)_i = \begin{cases} 1 & i \in J, \\ 0 & i \notin J. \end{cases}$$

Por definición de las restricciones bound factor, dado $\bar{J} \subset v$, al evaluar $F^L(\bar{J}, v \setminus \bar{J})$ en $\boldsymbol{x}^J$, vale cero salvo que $\bar{J} = J$, en cuyo caso se obtiene uno. Así, al evaluar el lado izquierdo de 2.7 en $\boldsymbol{x}^J$, este vale λ_J.

Finalmente, argumentamos que al evaluar el lado derecho de 2.7 en cualquier punto $\boldsymbol{x} \in \{0,1\}^v$ se obtiene un valor no negativo. En efecto, si $X_v = \prod_{i \in v} x_i = 1$, entonces $X_{s_k} = \prod_{i \in s_k} x_i = 1$ para todo k, y por tanto la expresión se convierte en $(n-1) - n + 1 = 0$. En caso contrario, al menos un monomio hijo se anula, lo que implica

$$\sum_{k=1}^{n} X_{s_k} \leq n - 1,$$

y por tanto

$$(n-1) - \sum_{k=1}^{n} X_{s_k} + X_v \geq 0.$$

Por tanto, se verifica que para cada $J \subseteq v$, $\lambda_J \geq 0$. □

Corolario 2.27. *Puesto que el anterior resultado únicamente hace uso de monomios que aparecen en el problema, este sigue verificándose cuando sólo se construyen las restricciones bound factor asociadas a los J-sets.*

Teorema 2.28 (Dominancia de la versión base de la técnica RLT frente a los patrones de linealización)**.** *Sea $G = (V, A)$ un patrón de linealización de $MIPP^\Omega$ y supongamos que para cada $j \in N$, $l_j < u_j$. Entonces, ${}^{\text{CR}}RLT(MIPP^\Omega)$ es al menos tan ajustada como ${}^{\text{CR}}RLT(RMIPP^\Omega_G)$.*

Demostración. Es consecuencia inmediata de las Proposiciones 2.25 y 2.26. □

Corolario 2.29 (Dominancia de la versión base de la técnica RLT frente a los patrones de linealización en relajaciones enteras)**.** *Sea $G = (V, A)$ un patrón de linealización de $MIPP^{\Omega}$ y supongamos que para cada $j \in N$, $l_j < u_j$. Entonces, la relajación $RLT(MIPP^{\Omega})$ es al menos tan ajustada como $RLT(RMIPP^{\Omega}_{G})$.*

El análisis anterior muestra que, al trabajar con relajaciones continuas, reformular previamente los productos de variables binarias puede empeorar el ajuste respecto a la formulación RLT base. Pese a que la relación también se da para las relajaciones enteras, cuando las variables binarias conservan su integralidad, las restricciones bound factor pueden imponer relaciones adicionales que no están presentes en la relajación continua, haciendo que ambas relajaciones sean igual de ajustadas. Los resultados recogidos en lo que resta de capítulo fueron íntegramente probados por mí.

2.2.2. Comparación del ajuste de las relajaciones enteras

Al final de la anterior sección probamos una determinada relación entre los ajustes de las relajaciones continuas, que se podía extrapolar a las relajaciones enteras. Sin embargo, veamos que es posible probar algo todavía más fuerte: que son igual de ajustadas.

Observación 2.30. *Los resultados que se presentan a continuación para variables binarias únicamente se apoyan en el hecho de que los únicos valores que una variable binaria puede tomar en la relajación entera son 0 y 1; esto es, $x_j = l_j$ o $x_j = u_j$. De esta forma, estos resultados son igualmente válidos para cualquier variable que sólo pueda tomar valores en los extremos de su rango en la relajación entera. En particular, se aplica a las variables que sustituyen productos de binarias aunque se presenten como continuas.*

Apoyándonos en el hecho de que las variables binarias sólo pueden tomar los valores de sus cotas, presentamos el siguiente resultado como corolario directo del Lema 1.19.

Proposición 2.31. *Sea y_i una variable binaria o una variable que sustituye al producto de binarias, e I un producto de variables de forma que $I \subset J$, con $J \cup \{i\}$ monomio del problema. Entonces, para cualquier solución factible de $RLT(MIPP^{\Omega})$, se verifica que $X_{I\cup\{i\}} = X_I \cdot y_i$.*

El anterior resultado nos invita a pensar que quizás las restricciones bound factor asociadas al producto de binarias pueden resultar redundantes en la relajación entera. Presentamos el siguiente resultado que así lo acredita.

Proposición 2.32. *Sea J un producto de variables e i una variable binaria o variable que sustituye al producto de binarias. Supongamos que $J \cup \{i\}$ es un monomio para el que hay que construir sus restricciones bound factor. Entonces, de sus restricciones bound factor se deduce lo siguiente:*

- *Si $y_i = 0$, entonces las restricciones donde $i \in J_1$ imponen que $X_{I\cup\{i\}} \geq 0$ o $X_{I\cup\{i\}} \leq 0$, con $I \subset J$, y las restricciones donde $i \in J_2$ son las restricciones bound factor de J.*
- *Si $y_i = 1$, entonces las restricciones donde $i \in J_1$ son las restricciones bound factor de J, mientras que las restricciones donde $i \in J_2$ imponen que $X_{I\cup\{i\}} \geq X_I$ o $X_{I\cup\{i\}} \leq X_I$, con $I \subset J$.*

Demostración. Lo demostraremos por inducción sobre $|J|$. Para comprobar que se impone $X_{I\cup\{i\}} \geq 0$ o $X_{I\cup\{i\}} \leq 0$ (alternativamente $X_{I\cup\{i\}} \geq X_I$ o $X_{I\cup\{i\}} \leq X_I$) únicamente lo comprobaremos para $I = J$, ya que las demás se deducen del Lema 1.16.

- Sea x_1 una variable, $l_1 \leq x_1 \leq u_1$ y y_2 una variable binaria, $0 \leq y_2 \leq 1$. Denotemos por X_{12} a la variable RLT asociada a $x_1 y_2$. Consideremos las restricciones bound factor

$$F_{\delta}(J_1, J_2) = \prod_{j\in J_1} (x_j - l_j) \prod_{j\in J_2} (u_j - x_j) \geq 0, \qquad \forall J_1 \cup J_2 = J.$$

  De esta forma, al linealizar se tiene:

1. $J_1 = \{1,2\}, J_2 = \emptyset$: $(x_1 - l_1)y_2 \geq 0 \Rightarrow X_{12} \geq l_1 y_2$.
2. $J_1 = \{1\}, J_2 = \{2\}$: $(x_1 - l_1)(1 - y_2) \geq 0 \Rightarrow x_1 - l_1 - X_{12} + l_1 y_2 \geq 0 \Rightarrow X_{12} \leq x_1 - l_1(1 - y_2)$.
3. $J_1 = \{2\}, J_2 = \{1\}$: $(u_1 - x_1)y_2 \geq 0 \Rightarrow X_{12} \leq u_1 y_2$.
4. $J_1 = \emptyset, J_2 = \{1,2\}$: $(u_1 - x_1)(1 - y_2) \geq 0 \Rightarrow u_1 - x_1 - u_1 y_2 + X_{12} \geq 0 \Rightarrow X_{12} \geq x_1 - u_1(1 - y_2)$.

Veamos qué pasa dependiendo del valor que toma y_2. Cuando $y_2 = 0$:

$$X_{12} \geq 0, \quad X_{12} \leq x_1 - l_1, \quad X_{12} \leq 0, \quad X_{12} \geq x_1 - u_1.$$

De la primera y tercera restricciones se deduce $X_{12} = 0$, y sustituyendo en las otras dos se tiene $x_1 - l_1 \geq 0$ y $u_1 - x_1 \geq 0$, que son las restricciones bound factor asociadas a x_1. En el caso en que $y_2 = 1$:

$$X_{12} \geq l_1, \quad X_{12} \leq x_1, \quad X_{12} \leq u_1, \quad X_{12} \geq x_1.$$

De la segunda y cuarta restricciones se deduce $X_{12} = x_1$, y reemplazando esta identidad en las restantes, obtenemos las restricciones bound factor de x_1.

- Supongamos que para un monomio J de grado $\delta \in \{1, \ldots, k-1\}$, las restricciones bound factor verifican la proposición y veamos que esto también es cierto para $|J| = k$. Teniendo en cuenta que las restricciones bound factor de un monomio J implican las de $I \subset J$, podemos asumir que para cualquier $I \subsetneq J$ se tienen las restricciones bound factor asociadas a $X_{I \cup \{i\}}$, luego podremos suponer que $X_{I \cup \{i\}} = X_I y_i$.

  Consideremos las restricciones bound factor de J:

$$\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \geq 0 \quad \forall J_1, J_2 : J_1 \cup J_2 = J.$$

  Nótese que el anterior producto se puede desarrollar de la siguiente forma:

$$\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) = \prod_{j \in J} (-1)^{|J_2|} x_j + f(J),$$

  donde el primer término es la combinación que tiene todos los productos de variables y el segundo es un polinomio donde todos los términos son monomios de grado menor o igual que $k-1$. Denotando su linealización por $[f(J)]_L$, por la hipótesis de inducción podemos afirmar que se verifica:

$$[f(J) \cdot y_i]_L = [f(J)]_L \cdot y_i.$$

  Para construir las restricciones bound factor de $J \cup \{i\}$, partimos de una bound factor de J y multiplicamos por y_i si $\{i\} \in J_1$, y por $(1 - y_i)$ si $\{i\} \in J_2$:

  - $\{i\} \in J_1$:

$$\Big[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) y_i \Big]_L = (-1)^{|J_2|} X_{J \cup \{i\}} + [f(J)]_L \cdot y_i \geq 0.$$

  - $\{i\} \in J_2$:

$$\begin{aligned} &\Big[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)(1 - y_i) \Big]_L \\ =&\Big[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \Big]_L - \Big[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) y_i \Big]_L \\ =&(-1)^{|J_2|} X_J + [f(J)]_L - (-1)^{|J_2|} X_{J \cup \{i\}} - [f(J)]_L \cdot y_i \geq 0. \end{aligned}$$

Desglosamos por casos en función del valor de y_i. Cuando $y_i = 0$:

- $\{i\} \in J_1$: $(-1)^{|J_2|} X_{J\cup\{i\}} \geq 0$.
- $\{i\} \in J_2$: $(-1)^{|J_2|} X_J + [f(J)]_L - (-1)^{|J_2|} X_{J\cup\{i\}} \geq 0$. Como de $\{i\} \in J_1$ se deduce que $X_{J\cup\{i\}} = 0$, se sigue que:
$$(-1)^{|J_2|} X_J + [f(J)]_L \geq 0,$$
que son las restricciones bound factor de J.

En el caso en que $y_i = 1$, se tiene que:

- $\{i\} \in J_2$: $(-1)^{|J_2|} X_J + [f(J)]_L - (-1)^{|J_2|} X_{J\cup\{i\}} - [f(J)]_L \geq 0 \Rightarrow (-1)^{|J_2|} X_{J\cup\{i\}} \leq (-1)^{|J_2|} X_J$.
- $\{i\} \in J_1$: $(-1)^{|J_2|} X_{J\cup\{i\}} + [f(J)]_L \geq 0$, que son las restricciones bound factor asociadas a J reemplazando J por $J \cup \{i\}$. Puesto que de $\{i\} \in J_2$ se deduce que $X_{J\cup\{i\}} = X_J$, se sigue que:
$$(-1)^{|J_2|} X_J + [f(J)]_L \geq 0,$$
recuperando las restricciones bound factor de J. □

Proposición 2.33. *Las relajaciones $RLT(MIPP^{\Omega})$ y $RLT(RMIPP^{\Omega}_G)$ son equivalentes.*

Demostración. Para demostrarlo, nos será de utilidad el resultado que acabamos de probar. Consideremos un monomio J sobre el que hay que construir las restricciones bound factor, separamos en casos.

- Si un monomio no contiene variables binarias, sus restricciones bound factor asociadas son iguales en cualquier caso.
- Si un monomio está conformado únicamente por variables binarias, las restricciones bound factor asociadas imponen $X_K = \prod_{i\in K} y_i$, $\forall K \subset I$, mientras que en las linealizaciones de binarias se imponen dichas igualdades para los monomios presentes en el patrón de linealización.
- Si un monomio está conformado por variables binarias y no binarias; es decir, es de la forma $J \cup I$, siendo J las variables no binarias e I las binarias:
  - Si se emplea un patrón de linealización de Elloumi-Verchere, denotemos por y_i a la variable asociada a I. Tenemos un producto de no binarias por una única binaria. Por la proposición anterior, las restricciones bound factor asociadas a este monomio son las restricciones bound factor de J más las restricciones que imponen $X_{K\cup\{I\}} = X_K y_i, K \subset J$.
  - Si estamos en la versión base, podemos aplicar la proposición anterior repetidamente hasta deshacernos de toda la parte binaria. Entonces las restricciones resultantes son las restricciones bound factor de J más restricciones que imponen $X_K = X_J \prod_{i\in K\setminus J} y_i$, con $J \subset K \subset J \cup I$.

De esta forma, se puede establecer una biyección entre el conjunto de soluciones de ambos problemas. Para ello, podemos construir una proyección del conjunto de soluciones de la versión base al patrón de linealización, y la inyectividad se tiene porque las variables que no están presentes en $RLT(RMIPP^{\Omega}_G)$ están unívocamente determinadas en la relajación entera por las anteriores restricciones. □

El anterior resultado nos podría llevar a pensar que cuando permitimos que el solver auxiliar resuelva las relajaciones enteras, el algoritmo basado en la técnica RLT debería converger en el mismo número de iteraciones. Sin embargo, la falta de unicidad en las soluciones óptimas de cada nodo puede hacer que el solver encuentre soluciones distintas, con violaciones RLT también distintas, de modo que no se seleccione la misma variable para ramificar. Esto hace que en general el número de iteraciones del algoritmo sea distinto.

No obstante, y puesto que esto es algo que escapa a nuestro control, sugiere que en este contexto la mejor formulación será aquella que presente un menor tiempo de resolución de la relajación lineal.

Capítulo 3

Resultados computacionales

El objetivo de este capítulo es contrastar computacionalmente las ideas desarrolladas en los capítulos anteriores. En particular, queremos estudiar cómo afecta la reformulación de productos de variables binarias al rendimiento de RAPOSa y comparar distintas formas de gestionar las variables enteras durante el proceso de ramificación. Para ello no basta con analizar únicamente el tiempo total de resolución, sino que será necesario observar también el número de nodos generados, el tiempo empleado en resolver las relajaciones lineales y la calidad de las cotas obtenidas en el nodo raíz.

La exposición se organizará en tres partes. Empezaremos el capítulo poniendo de manifiesto las carencias de las baterías empleadas en González-Díaz et al. (2024), debido a que se componen principalmente de problemas de grado 2 y esporádicamente alguno de grado mayor, lo que puede inducir a conclusiones erróneas. De esta forma, lo anterior nos condujo a diseñar una batería de problemas propia, donde pudiéramos seleccionar diversos parámetros como el grado del problema o el número de variables. Una vez hecho, la segunda parte se dedicará a analizar los resultados computacionales de las baterías resueltas, primero sobre todas las posibles reformulaciones, y después poniendo especial énfasis en la comparación entre estrategias de ramificación. Por último, presentaremos los resultados obtenidos, primero de forma agregada y después mediante el análisis de algunas instancias concretas comparando una representación de sus árboles de ramificación y acotación.

3.1. Generación de problemas

Antes de construir una nueva batería de problemas, conviene justificar por qué no resulta suficiente emplear directamente las librerías habituales. En nuestro caso, la dificultad aparece porque muchas de las instancias disponibles son cuadráticas, y en ese contexto la reformulación de productos de variables binarias no introduce diferencias sustanciales respecto a la formulación base.

Proposición 3.1. *Para problemas cuadráticos, cualquiera de las linealizaciones de binarias coincide con la formulación base de RAPOSa.*

Demostración. Diferenciemos por casos según el número de binarias que componen el monomio. Si el monomio está conformado por dos variables no binarias, las restricciones bound factor asociadas son iguales. Si el monomio contiene una variable binaria y otra no binaria, la aplicación de un patrón de linealización no afecta a la parte binaria, así que las restricciones bound factor asociadas son iguales. De esta forma, únicamente tenemos que fijarnos en lo que sucede al linealizar un monomio con dos variables binarias. Consideremos x_1x_2 el monomio a linealizar, en la versión base, las restricciones bound factor son de la forma:

$$X_{12} \geq 0, \quad X_{12} \leq x_1, \quad X_{12} \leq x_2, \quad X_{12} \geq x_1 + x_2 - 1,$$

que efectivamente coinciden con las restricciones de la linealización estándar y de cualquier patrón de linealización. □

Observación 3.2. *Pese a que los problemas tienen las mismas restricciones, en el caso de la versión base se mide* $|X_{12} - x_1x_2|$ *y se tiene en cuenta a la hora de calcular las violaciones RLT. Aunque esto no tenga efecto si se priorizan las ramificaciones de integralidad, sí que puede resultar importante en un enfoque RLT-first.*

Puesto que la mayoría de los problemas de la batería `QPLIB` y `MINLPLib` son de grado 2, con contadas excepciones de grado superior, consideramos pertinente desarrollar una nueva batería con la cual medir con más detalle el impacto de las reformulaciones de binarias.

Para la generación, los parámetros más destacables son los que se presentan a continuación:

- `v` $\in \mathbb{N} \cup \{0\}$ es el número de variables continuas.
- `b` $\in \mathbb{N} \cup \{0\}$ es el número de variables binarias.
- `gv` $\in \mathbb{N} \cup \{0\}$ es el grado máximo de la parte continua de un monomio.
- `gb` $\in \mathbb{N} \cup \{0\}$ es el grado máximo de la parte binaria de un monomio.
- `k` ≥ 0 es un factor de escala que controla el número de restricciones.
- `d` $\in (0, 1]$ es la proporción de monomios presentes en el problema.
- `p` $\in (0, 1]$ es la proporción de monomios del problema que aparecen en cada restricción.

No resulta complicado probar que el número de monomios de hasta grado g_v que se pueden formar con v variables es:

$$N_v = \sum_{r=0}^{g_v} \binom{v + r - 1}{r}.$$

Sin embargo, al entrar en juego variables binarias, tenemos que tener en cuenta que si y_i es binaria, entonces $y_i^n = y_i \ \ \forall n \in \mathbb{N},\ n \geq 1$. De esta forma, teniendo en cuenta que la anterior simplificación se hace en RAPOSa, la cantidad de monomios de hasta grado g_b que se pueden conformar con b variables binarias es:

$$N_b = \sum_{r=0}^{g_b} \binom{b}{r}$$

Por tanto, seleccionados estos cuatro parámetros, el número de monomios de los que disponemos es

$$M_{\text{total}} = N_v \cdot N_b$$

Puesto que es raro que un problema tenga todos los monomios posibles, el parámetro `d` controla la cantidad de ellos seleccionados:

$$M_{\text{global}} = d \cdot M_{\text{total}}$$

El proceso de generación de un monomio es el siguiente:

- Salvo que se indique lo contrario, se selecciona el grado t_v de la parte continua de forma uniforme entre 0 y `gv` y después se genera una v-tupla de enteros no negativos cuya suma es dicho grado (lo que se conoce como una composición débil). Se realiza lo análogo para la parte binaria y se juntan. Se multiplica por un coeficiente entero no nulo en el rango de `coef_obj`.

El proceso de selección de los monomios es como sigue:

- Se genera un monomio con una parte continua de grado `gv` y con una parte binaria de grado `gb` para garantizar que el problema sea de grado `gv` + `gb`.
- Se generan monomios hasta llegar a M_{global}.

- Se comprueba que todas las variables aparecen en el problema. En caso contrario, se muestrea sobre el conjunto de monomios todavía por cubrir de forma análoga a la anterior eliminando las variables cubiertas en cada iteración.

Si bien la última comprobación puede hacer que la proporción de monomios presentes en el problema se incremente, no es algo que suceda de manera habitual salvo que se seleccione una d extremadamente baja. Por ello no planteamos ninguna alternativa como generar monomios que cubran todas las variables al principio del proceso de generación o reemplazar monomios ya construidos por otros.

Una vez seleccionados los monomios que van a estar presentes en el problema, se añaden todos a la función objetivo con coeficientes enteros no nulos en el rango de `coef_obj`.

Ahora se construyen $\mathtt{k}\cdot(\mathtt{v}+\mathtt{b})$ restricciones. Los parámetros $\mathtt{q\_geq} \in [0,1]$ y $\mathtt{q\_eq} \in [0,1]$ controlan la proporción de ellas que son del tipo "mayor o igual" y de igualdad respectivamente. En cada restricción hay presentes $\mathtt{p} \cdot M_{\text{global}}$ monomios, con coeficientes enteros no nulos en el rango de `coef_con`, y con lados derechos en el rango de `rhs_range`.

Puesto que no tenemos garantías de que un problema así generado vaya a ser factible, añadimos variables de holgura para garantizar que lo sea. De esta forma, el problema termina siendo de la forma:

$$
\begin{aligned}
\text{minimizar} \quad & \phi_0(\boldsymbol{x}) + \mathtt{penalty}\left(\sum_r Y_{\text{mas},r} + Y_{\text{menos},r}\right) \\
\text{sujeto a} \quad & \phi_r(\boldsymbol{x}) + \mathtt{coef\_Y\_mas[r]} * Y_{\text{mas},r} \geq \beta_r, && r = 1,\dots,R_1, \\
& \phi_r(\boldsymbol{x}) - \mathtt{coef\_Y\_menos[r]} * Y_{\text{menos},r} \leq \beta_r, && r = R_1+1,\dots,R_2, \\
& \phi_r(\boldsymbol{x}) + \mathtt{coef\_Y\_mas[r]} * Y_{\text{mas},r} - \mathtt{coef\_Y\_menos[r]} * Y_{\text{menos},r} = \beta_r, && r = R_2+1,\dots,R, \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^n,
\end{aligned}
$$

El parámetro `use_additive_slack` se emplea para determinar los coeficientes `coef_Y_menos[r]` y `coef_Y_mas[r]`, y la operación $*$.

- Si vale 0, entonces $*$ es el producto, y $\mathtt{coef\_Y\_menos[r]} = \mathtt{coef\_Y\_mas[r]} = 1$ para todo r.
- Si vale 1, entonces $*$ es la suma, y $\mathtt{coef\_Y\_menos[r]} = \mathtt{coef\_Y\_mas[r]} = 0$ para todo r.

El motivo para añadir estos coeficientes adicionales, a priori innecesarios, es evitar que la función objetivo tome valores muy grandes, ya que estos pueden dar lugar a problemas numéricos.

El programa tiene más de un modo de generación, ya que aparte de generar los problemas, tiene la posibilidad de resolverlos en RAPOSa conectándose al Cesga.

- Por una parte, podemos enviarlo y descargarnos la reformulación del nodo raíz para constatar que es un problema factible.
- Por otra parte, enviándolo con dos reformulaciones de binarias diferentes, si descargamos los nodos raíz y los resolvemos en local, podemos quedarnos con problemas con distintas cotas inferiores, lo que puede resultar interesante a la hora de medir el impacto de la reformulación de binarias.

Al resolverlo con `Gurobi`, si fijamos `create_coefup` en 1, nos permitirá también actualizar los valores de `coef_Y_menos[r]` y `coef_Y_mas[r]` con los valores de Y_{mas} e Y_{menos}, para que así cuando se resuelva con RAPOSa se esté "más cerca de la factibilidad".

Las cotas de las variables de holgura, inicialmente fijadas a `initial_slack_variables_bound`, se actualizan a `slack_variables_bound`, y `coef_Y_mas[r]` pasa a ser

$$\mathtt{initial\_slack\_variables\_bound} \cdot \lfloor Y_{\text{mas}}/\mathtt{initial\_slack\_variables\_bound} \rfloor.$$

Una iteración previa de esta batería consistía en considerar todas las variables juntas y que una proporción de ellas fuera binaria. Esto resultaba conflictivo, ya que cuando la proporción de variables

binarias era considerablemente alta o baja, esto podía tener impacto sobre los problemas generados, ya que afectaba a la probabilidad de seleccionar variables de uno u otro tipo. Sin embargo, como este error fue lo que nos motivó a emplear este segundo diseño, abordaremos algún resultado de alguna de las baterías generadas de la anterior forma. Así, cuando hablemos de

`v{v}_gv{gv}_b{b}_gb{gb}_k{k}_d{d_tag:05d}_p{p_tag:03d}_{seed}`,

nos estaremos refiriendo a un problema con la generación actual, mientras que cuando empleemos

`v{v}_k{k}_g{g}_d{d_tag:05d}_p{p_tag:03d}_bin{bin}_{seed}`

se referirá a la anterior. En dicho caso, el significado de las variables será el siguiente:

- $\texttt{v} \in \mathbb{N} \cup \{0\}$ es el número de variables.
- $\texttt{k} \geq 0$ es el factor de escala que controla el número de restricciones.
- $\texttt{g} \in \mathbb{N} \cup \{0\}$ es el grado del problema.
- $\texttt{d} \in (0, 1]$ es la proporción de monomios presentes en el problema.
- $\texttt{p} \in (0, 1]$ es la proporción de monomios del problema que aparecen en cada restricción.
- $\texttt{bin} \in [0, 1]$ es la proporción de variables que son binarias.

En cualquier caso, las únicas variables enteras con las que trabajaremos serán binarias.

3.2. Entorno de pruebas

Las ejecuciones de los problemas se llevaron a cabo empleando la herramienta RAPOSa Cloud, un software complementario a RAPOSa que facilita la ejecución de pruebas y la obtención de resultados. En particular, RAPOSa Cloud emplea los servidores del Cesga para ejecutar los problemas.

Dichos servidores cuentan con una elevada capacidad de cómputo, incorporando procesadores *Intel Xeon Platinum 8352Y* de 64 núcleos y 256 GB de memoria RAM. Para cada problema se fijó un tiempo máximo de ejecución de 1 hora, y aquellos que no se resolvieron dentro del límite se consideraron no resueltos.

3.3. Métricas de evaluación

Para evaluar el funcionamiento de los mecanismos de reformulación disponemos de diversas métricas. En este trabajo detallaremos el uso de las siguientes.

- **Geometric mean time (GMT)**: media geométrica del tiempo utilizado por el solver para resolver cada uno de los problemas. Aquí se excluyen los problemas no resueltos por ninguno de los mecanismos y aquellos resueltos por todos en menos de 5 segundos, ya que no aportan información significativa en la comparación. El empleo de la media geométrica en lugar de la aritmética busca mitigar el efecto de los posibles valores atípicos.
- **Geometric mean gap (GMG)**: media geométrica de los gaps obtenidos por el optimizador. Cuando el solver no es capaz de resolver el problema para las tolerancias impuestas, proporciona una cota inferior y una cota superior para su valor óptimo, cuya diferencia se denomina gap. Cuanto menor sea, mejor es la aproximación. En esta métrica se excluyen los problemas resueltos por todas las estrategias y aquellos en los que alguna de las configuraciones no ha sido capaz de proporcionar un gap, ya que en estos casos no se pueden comparar adecuadamente los valores obtenidos.

- **Número total de problemas resueltos**: se considera que un problema ha sido resuelto si ha obtenido una solución dentro de las tolerancias en un tiempo inferior a 1 hora.
- **Geometric mean nodes (GMN)**: media geométrica del número de relajaciones resueltas para resolver el problema.
- **Geometric mean linear time (GMLT)**: media geométrica del promedio del tiempo de resolución de la relajación que se envía al solver auxiliar.
- **Mean relative difference lower bound (GMRDLB)**: media aritmética de la diferencia relativa respecto a la peor cota inferior. Este valor es un número entre 0 y 1 que representa lo ajustada que es la formulación en el nodo raíz:

$$RDLB = \begin{cases} \frac{LB - \text{mín } LB}{\text{máx } LB - \text{mín } LB}, & \text{si } \text{mín } LB < \text{máx } LB; \\ 1, & \text{si } \text{mín } LB = \text{máx } LB. \end{cases}$$

- **Número de problemas resueltos**: el número que acompaña al nombre de la configuración indica la cantidad de problemas resueltos en el tiempo límite.

Una vez fijadas las métricas, pasamos a comparar las distintas configuraciones consideradas. En cada batería se combinan diferentes reformulaciones de la parte binaria con distintas estrategias para tratar las variables enteras. Por un lado, compararemos la versión base de RAPOSa con la linealización estándar y con varios patrones de linealización. Por otro, distinguiremos entre priorizar las violaciones de integralidad, priorizar las violaciones RLT o resolver directamente las relajaciones enteras.

El objetivo de esta comparación es doble. En primer lugar, queremos comprobar si las reformulaciones de productos de variables binarias mejoran el rendimiento respecto a la formulación base. En segundo lugar, queremos determinar en qué situaciones resulta conveniente resolver relajaciones enteras, teniendo en cuenta que esta estrategia puede reducir el número de nodos, pero también aumenta el coste de resolver cada relajación.

3.4. Resultados de las pruebas

En primer lugar, vamos a probar a realizar todas las combinaciones de reformulaciones de binarias (versión de RAPOSa base, linealización estándar, patrón de linealización canónico no dominado, heurística 1 y heurística Lucía) con las 3 posibles estrategias de resolución de problemas con variables enteras (encargándonos nosotros de las ramificaciones enteras, donde podemos adoptar un enfoque *integrality-first* o *RLT-first*, o enviando al solver auxiliar el problema con variables enteras). Empezamos midiendo el impacto sobre una batería de problemas que incluye alrededor de 50 problemas de cada una de las siguientes configuraciones:

- `v0_gv0_b25_gb7_k2_d00200_p030`
- `v5_gv1_b20_gb7_k2_d00200_p030`
- `v20_k2_g5_d02000_p010_bin75`
- `v20_k2_g5_d02000_p010_bin50`
- `v18_k2_g5_d02000_p010_bin50`
- `v16_k2_g5_d04000_p010_bin25`
- `v6_gv4_b8_gb4_k2_d00500_p030`
- `v6_gv4_b8_gb5_k2_d00500_p030`

	GMT (259)	**GMG (223)**	**GMN (35)**	**GMLT (35)**	**MRDLB (259)**
Base *int-first* (158)	603.01	0.013	61.88	0.694	0.2433
SL *int-first* (254)	96.40	0.001	69.04	0.401	0.0000
CNDP *int-first* (254)	93.29	0.001	69.04	0.403	0.0133
H1 *int-first* (254)	93.97	0.001	68.67	0.416	0.0191
HL *int-first* (254)	98.93	0.001	68.76	0.422	0.0190
Base *RLT-first* (116)	904.45	0.022	50.64	0.858	0.2433
SL *RLT-first* (42)	2046.38	0.153	122.80	0.538	0.0000
CNDP *RLT-first* (43)	2019.73	0.143	97.67	0.549	0.0133
H1 *RLT-first* (43)	1972.30	0.136	98.78	0.529	0.0191
HL *RLT-first* (45)	1958.58	0.133	98.75	0.529	0.0190
Base int. rel. (140)	726.63	0.023	15.34	9.953	1.000
SL int. rel. (258)	41.88	0.001	15.05	2.653	1.000
CNDP int. rel. (258)	47.31	0.001	15.04	2.694	1.000
H1 int. rel. (258)	40.70	0.001	15.03	2.626	1.000
HL int. rel. (258)	40.36	0.001	15.03	2.623	1.000

Tabla 3.1: Resultados computacionales de la primera batería.

En la Tabla 3.1 tenemos recogidas las métricas de evaluación para esta primera batería de problemas. En primer lugar, fijémonos en que las métricas de nodos y tiempo lineal se calculan únicamente sobre 35 problemas, lo que supone menos del 10 % del tamaño de la batería. Como podemos ver que la configuración que presenta peor rendimiento de todas es *RLT-first*, vamos a extraer conclusiones sobre esta estrategia, y posteriormente replicaremos esta tabla omitiéndola. De ella, podemos extraer las siguientes conclusiones:

- ***RLT-first* es la peor estrategia de ramificación**: En un primer vistazo vemos que *RLT-first* resuelve una menor cantidad de problemas que las otras dos configuraciones, y, en los problemas que resuelve, suele requerir un mayor número de nodos que los demás.

- **Impacto negativo de la linealización de binarias**: Pese a que el tiempo lineal medio es inferior, el número de nodos y el tiempo de resolución se incrementan al aplicar una linealización de binarias, al contrario de lo que sucede en las otras dos estrategias. En primer lugar, los resultados sobre la equivalencia de las reformulaciones únicamente aplican sobre las relajaciones enteras de los problemas. De esta forma, si no tenemos fijados los valores de las variables binarias se espera que requiera un mayor número de iteraciones resolver el problema. La cuestión es que la versión base es capaz de reducir el número de variables binarias libres, ya que al no linealizar el producto de binarias, todavía se calculan sus violaciones RLT, permitiendo ramificar por ellas antes de encontrar una solución sin violaciones RLT, que es lo que requiere cualquiera de las versiones que linealiza las binarias.

  En ese aspecto, resulta pertinente destacar que, al contrario de lo que sucede con las violaciones RLT, las de integralidad son finitas, ya que a lo sumo en una rama se puede ramificar por el número de binarias que haya.

- **Mejora de los patrones de linealización frente a la linealización estándar**: En línea con el argumento empleado en el punto anterior, disponer de unas relajaciones más ajustadas hace que el número de nodos sea inferior sin tener un impacto significativo sobre el tiempo de resolución de la relajación.

- **La versión base con *RLT-first* mejora a la versión base con *integrality-first***: En los problemas resueltos vemos que el número de nodos es inferior en el primero. El motivo probablemente sea porque al no hacer linealización de binarias, la discrepancia entre un producto de binarias y la variable RLT que lo sustituye se tiene en cuenta como una violación RLT, y así

es posible ramificar por variables binarias (que al linealizar binarias sólo se podría considerar ramificación por violación de integralidad) cuando todavía hay violaciones RLT.

- **Mismas cotas inferiores en el nodo raíz en *integrality-first* que en *RLT-first* para una configuración dada**: Puesto que en ambos casos se resuelve la misma relajación en el nodo raíz, es normal que se obtengan los mismos valores en ambos casos.

Una vez visto esto, volvemos a construir la tabla pero omitiendo las configuraciones *RLT-first*, a la espera de aumentar el número de problemas resueltos. Algo que resulta llamativo en la Tabla 3.2 es

	GMT (376)	**GMG (188)**	**GMN (187)**	**GMLT (187)**	**MRDLB (376)**
Base *int-first* (232)	560.14	0.057	100.22	0.866	0.3146
SL *int-first* (370)	109.36	0.001	113.12	0.270	0.0000
CNDP *int-first* (370)	96.48	0.001	110.76	0.267	0.0138
H1 *int-first* (370)	97.44	0.001	112.39	0.267	0.0204
HL *int-first* (370)	102.24	0.001	112.02	0.272	0.0204
Base int. rel. (202)	575.09	0.167	10.48	10.235	0.8124
SL int. rel. (375)	31.23	0.001	10.52	1.792	1.000
CNDP int. rel. (375)	40.05	0.001	10.52	1.797	1.000
H1 int. rel. (375)	31.55	0.001	10.53	1.698	1.000
HL int. rel. (375)	31.25	0.001	10.53	1.682	1.000

Tabla 3.2: Resultados computacionales de la primera batería omitiendo *RLT-first*.

cómo el valor de MRDLB para la versión base de RAPOSa resolviendo las relajaciones enteras deja de valer 1. Esto se debe a que ante problemas más complicados, RAPOSa es incapaz de resolver la relajación entera en el nodo raíz dentro del tiempo límite. En ese supuesto, se resuelve la relajación continua del problema para garantizar que se tiene una cota inferior, y esta es la que se reporta como primera cota inferior, lo que da lugar a un valor inferior a 1. Veamos qué más extraemos de aquí.

- **El uso de linealizaciones de binarias tiene un impacto positivo**: Vemos que en ambos enfoques de gestión de enteras se reduce el tiempo de ejecución y el tiempo lineal medio, mientras que el incremento en el número de nodos en *integrality-first* no es tan grande como para hacerlo empeorar.
- **El número de iteraciones resolviendo las relajaciones enteras es muy similar**: Esto se debe a que dichas relajaciones son equivalentes, y que la posible discrepancia nazca de ramificar por otra variable al haber encontrado un óptimo distinto, lo cual se escapa a nuestro control.
- **Las heurísticas son ligeramente más ajustadas que CNDP**: No obstante, queda claro que no son capaces de acercarse al ajuste que proporciona la versión base, a la vez que estas están bastante lejos del ajuste de las relajaciones enteras.
- **Hacer que el solver auxiliar gestione las relajaciones enteras *parece* mejor que hacer que RAPOSa gestione las violaciones de integralidad**: A la vista de los tiempos de resolución obtenidos, esta conclusión parece evidente y es también a la que se llega en González-Rodríguez et al. (2022). No obstante, a continuación vamos a replicar el anterior análisis separando por cada configuración, y negaremos que esto sea así en cualquier caso.

3.4.1. `v0_gv0_b25_gb7_k2_d00200_p030`

Al ser un problema sin variables continuas, está claro que el problema resultante de aplicar una linealización de binarias se convierte en un problema lineal con variables enteras donde no hay que construir identidades RLT. De esta forma, cuando se resuelve la relajación entera se ha de hacer en una única iteración. Adicionalmente, ante esa ausencia de identidades RLT, las únicas violaciones son de

integralidad. Por tanto, cualquier linealización de binarias debería presentar el mismo comportamiento en *integrality-first* que en *RLT-first*.

A fin de reducir espacio, omitimos la inclusión de *RLT-first* por lo que acabamos de decir. Como se aprecia en la Tabla 3.3 la versión base resolviendo las relajaciones enteras no consigue resolver ningún problema, omitimos las columnas relativas al número de nodos y tiempo lineal.

	GMT (50)	**GMG (50)**	**MRDLB (50)**
Base *int-first* (10)	3450.18	0.027	0.8942
SL *int-first* (50)	1524.78	0.001	0.0000
CNDP *int-first* (50)	768.98	0.001	0.0057
H1 *int-first* (50)	777.98	0.001	0.0212
HL *int-first* (50)	778.48	0.001	0.0220
Base int. rel. (0)	3600	0.105	0.8942
SL int. rel. (50)	48.14	0.001	1.000
CNDP int. rel. (50)	104.53	0.001	1.000
H1 int. rel. (50)	62.22	0.001	1.000
HL int. rel. (50)	63.25	0.001	1.000

Tabla 3.3: Resultados computacionales de `v0_gv0_b25_gb7_k2_d00200_p030`.

De la Tabla 3.3 podemos extraer las siguientes conclusiones:

- **Permitir que un solver auxiliar gestione las variables enteras supera claramente nuestra capacidad de resolución del problema**.
- **Impacto positivo de los patrones de linealización**.
- **La versión base de RAPOSa es bastante ajustada respecto a la relajación entera**: Con un valor cercano al 0.9, para problemas de optimización binarios parece que ofrece una relajación bastante ajustada, en comparación con lo que se veía con toda la batería en general.
- **Coincidencia de cotas inferiores en ambas versiones base**: Esto es una coincidencia particular de esta batería, ya que, como no se resuelve ninguna relajación entera con la versión base, la primera cota inferior que puede proporcionar es la de relajación continua, con lo cual son iguales.

Exploramos más en detalle el impacto de los patrones de linealización mostrando el número de nodos y tiempo lineal medio de estos. Lo tenemos recogido en la Tabla 3.4, omitiendo la columna de gaps y cotas inferiores iniciales porque no aportan información adicional.

	GMT (50)	**GMN (50)**	**GMLT (50)**
SL *int-first* (50)	1610.94	48137.12	0.0339
CNDP *int-first* (50)	803.01	16327.48	0.0501
H1 *int-first* (50)	819.62	21399.12	0.0389
HL *int-first* (50)	818.34	21354.84	0.0388
SL int. rel. (50)	52.56	1.000	46.7118
CNDP int. rel. (50)	107.82	1.000	101.7444
H1 int. rel. (50)	63.60	1.000	59.5907
HL int. rel. (50)	64.97	1.000	60.5012

Tabla 3.4: Resultados computacionales de `v0_gv0_b25_gb7_k2_d00200_p030`.

De la Tabla 3.4 podemos extraer las siguientes conclusiones.

- **Los patrones de linealización mejoran el rendimiento de la linealización estándar**: Puesto que las relajaciones lineales tienen un tiempo lineal medio similar, la diferencia en rendimiento surge de la reducción en el número de iteraciones necesarias para resolver el problema.

- **Comportamiento sorprendente de CNDP**: Por un lado, en *integrality-first* presenta un número de nodos notablemente inferior frente a las heurísticas, pero ante un tiempo lineal ligeramente superior, se compensan para terminar con tiempos de resolución muy similares.

  Por otra parte, llama la atención que en la resolución de las relajaciones enteras sea la que peor rendimiento presenta. Da la sensación de que aplicar este patrón de linealización canónico sin criterio hace que aparezcan muchas variables en el problema, lo que produce una relajación más difícil de resolver.

- **La proporción del número de nodos entre la resolución de relajaciones enteras e *integrality-first* es del orden de 10^5**: Será de especial interés estar al tanto de esta medida a lo largo de estos problemas.

- **La proporción del tiempo lineal entre la resolución de relajaciones enteras e *integrality-first* es del orden de 10^{-3}**: Se observa el mismo comportamiento que en el punto anterior.

3.4.2. `v5_gv1_b20_gb7_k2_d00200_p030`

Para agilizar la exposición, omitiremos entrar en detalle en cuestiones ya exploradas, como por qué resolver la relajación entera de la versión base devuelve una cota inferior inicial menor que 1.

	GMT (50)	**GMG (50)**	**MRDLB (50)**
Base *int-first* (0)	3600.00	0.682	0.2710
SL *int-first* (50)	985.43	0.001	0.0000
CNDP *int-first* (50)	830.64	0.001	0.0045
H1 *int-first* (50)	816.31	0.001	0.0063
HL *int-first* (50)	828.49	0.001	0.0062
Base *RLT-first* (0)	3600.00	0.695	0.2710
SL *RLT-first* (0)	3600.00	1.019	0.0000
CNDP *RLT-first* (0)	3600.00	1.014	0.0063
H1 *RLT-first* (0)	3600.00	1.012	0.0062
HL *RLT-first* (0)	3600.00	1.012	0.2710
Base int. rel. (0)	3600.00	0.727	1.0000
SL int. rel. (50)	55.39	0.001	1.0000
CNDP int. rel. (50)	87.12	0.001	1.0000
H1 int. rel. (50)	60.19	0.001	1.0000
HL int. rel. (50)	59.03	0.001	1.0000

Tabla 3.5: Resultados computacionales de `v5_gv1_b20_gb7_k2_d00200_p030`.

De la Tabla 3.5 podemos extraer las siguientes conclusiones.

- **El rendimiento de *RLT-first* es claramente peor que el de *integrality-first***: Esto puede resultar llamativo porque realmente hay a lo sumo una única variable continua por monomio, luego las únicas violaciones RLT que se dan son entre esta y la parte binaria. No obstante, sabemos que eso no produce una violación RLT en la relajación entera, y, por tanto, esta se ha de resolver en una iteración, como comprobaremos en la Tabla 3.6.

  Nótese que el comportamiento de *RLT-first* es esperable. Como sólo hay 5 variables continuas, son las que aparecerán con mayor frecuencia en los monomios del problema, ya que estas pueden acompañar a cualquier producto de binarias, mientras que la variable que sustituye a un producto de binarias puede aparecer, a lo sumo, junto a 5 variables continuas. En caso de que se empleara un criterio de ramificación como la suma, descrito en González-Rodríguez et al. (2022), siempre se ramificaría por variables continuas sin una mejora real de la solución, porque lo que hay que hacer es ramificar las violaciones de integralidad. Por suerte, el criterio de ramificación tiene en cuenta los rangos de las variables, y eventualmente se produce la ramificación por alguna binaria

o variable que sustituye al producto de binarias. Aun así, no es suficiente para conseguir que se resuelva ningún problema.

En el Capítulo 4 presentaremos una reformulación que nos permitirá esquivar esta problemática.

- **Impacto positivo de las linealizaciones de binarias**.
- **Mejor rendimiento del solver auxiliar gestionando las variables enteras que *integrality-first***.

	GMT (50)	**GMN (50)**	**GMLT (50)**
SL *int-first* (50)	985.43	15142.84	0.0590
CNDP *int-first* (50)	830.64	10257.48	0.0745
H1 *int-first* (50)	816.31	11327.85	0.0658
HL *int-first* (50)	828.49	11502.14	0.0657
SL int. rel. (50)	55.39	1.00	53.1477
CNDP int. rel. (50)	87.12	1.00	83.0458
H1 int. rel. (50)	60.19	1.00	56.7394
HL int. rel. (50)	59.03	1.00	55.9678

Tabla 3.6: Resultados computacionales de `v5_gv1_b20_gb7_k2_d00200_p030` sin la versión base ni *RLT-first*.

Destacamos lo siguiente de la Tabla 3.6:

- **La proporción del número de nodos entre la resolución de relajaciones enteras e *integrality-first* es del orden de 10^5**: aunque es menor que en la anterior batería explorada.
- **La proporción del tiempo lineal entre la resolución de relajaciones enteras e *integrality-first* es del orden de 10^{-3}**: pero es prácticamente igual que en la batería anterior.

A la vista del peor rendimiento del enfoque *RLT-first*, omitimos sus resultados computacionales a partir de aquí, teniendo en cuenta que en las baterías que discutimos a continuación se ha visto que no son competitivas respecto de las otras dos.

3.4.3. `v20_k2_g5_d02000_p010_bin75`

Seguimos reduciendo la cantidad de binarias del problema y ahora llegamos a uno de los problemas creados con una versión primitiva del generador.

	GMT (44)	**GMG (10)**	**GMN (34)**	**GMLT (34)**	**MRDLB (44)**
Base *int-first* (34)	308.68	0.113	517.41	0.6707	0.1090
SL *int-first* (40)	106.33	0.006	702.94	0.1549	0.0000
CNDP *int-first* (40)	93.20	0.006	670.47	0.1338	0.0097
H1 *int-first* (40)	93.53	0.006	645.29	0.1326	0.0169
HL *int-first* (40)	96.45	0.006	651.00	0.1341	0.0170
Base int. rel. (44)	31.76	0.001	6.53	4.2407	1.0000
SL int. rel. (44)	7.10	0.001	6.35	0.8856	1.0000
CNDP int. rel. (44)	9.57	0.001	6.35	0.9463	1.0000
H1 int. rel. (44)	7.05	0.001	6.35	0.8127	1.0000
HL int. rel. (44)	6.89	0.001	6.35	0.7922	1.0000

Tabla 3.7: Resultados computacionales de `v20_k2_g5_d02000_p010_bin75` omitiendo *RLT-first*.

Lo más destacable en la Tabla 3.7 es, sin duda, lo siguiente:

- **La proporción del número de nodos entre la resolución de relajaciones enteras e *integrality-first* es del orden de 10^3**: esto ya es un par de órdenes de magnitud inferior a lo que habíamos observado hasta ahora.

- **La proporción del tiempo lineal entre la resolución de relajaciones enteras e *integrality-first* en las linealizaciones de binarias es del orden de 10^{-1}**: aunque es necesario poner este resultado en tela de juicio ya que los tiempos de resolución medios son extremadamente bajos (inferiores a 10 segundos).

3.4.4. `v20_k2_g5_d02000_p010_bin50` y `v18_k2_g5_d02000_p010_bin50`

Puesto que ambas baterías son muy similares, las analizamos de manera conjunta.

	GMT (93)	GMG (7)	GMN (86)	GMLT (86)	MRDLB (44)
Base *int-first* (88)	187.77	0.031	387.18	0.7542	0.1595
SL *int-first* (91)	71.47	0.003	450.44	0.2559	0.0000
CNDP *int-first* (91)	72.59	0.003	429.90	0.2589	0.0182
H1 *int-first* (91)	73.63	0.003	448.86	0.2587	0.0187
HL *int-first* (91)	77.99	0.003	448.35	0.2620	0.0187
Base int. rel. (90)	118.83	0.003	28.65	6.1479	1.0000
SL int. rel. (93)	34.88	0.001	28.81	1.7737	1.0000
CNDP int. rel. (93)	39.07	0.001	28.37	1.7243	1.0000
H1 int. rel. (93)	33.67	0.001	28.47	1.6725	1.0000
HL int. rel. (93)	32.95	0.001	28.47	1.6626	1.0000

Tabla 3.8: Resultados computacionales de `v20_k2_g5_d02000_p010_bin50` y `v18_k2_g5_d02000_p010_bin50` omitiendo *RLT-first*.

Las métricas recogidas en la Tabla 3.8 nos permiten afirmar lo siguiente:

- **Cada vez las diferencias relativas entre tiempos de resolución de *integrality-first* y resolver la relajación entera son inferiores**: no obstante, todavía sigue sin ser lo suficientemente competitiva como para resultar atractiva.

- **La proporción del número de nodos entre la resolución de relajaciones enteras e *integrality-first* es del orden de 10^2**: vemos que se reduce todavía más.

- **La proporción del tiempo lineal entre la resolución de relajaciones enteras e *integrality-first* en las linealizaciones de binarias es del orden de $0{,}5 \cdot 10^{-1}$**: los tiempos son cada vez más parecidos.

3.4.5. `v16_k2_g5_d04000_p010_bin25`

A la vista de los resultados anteriores, y sabiendo que el rendimiento cuando no hay binarias tiene que ser exactamente igual, estamos inclinados a pensar que observaremos un comportamiento intermedio. Veamos qué sucede.

Atendiendo a los resultados recogidos en la Tabla 3.9, podemos extraer las siguientes conclusiones.

- **Resolver las relajaciones enteras no siempre es mejor que *integrality-first***: Al contrario de lo que se afirmaba en González-Rodríguez et al. (2022) o lo que podría parecer el resumen global de la batería, afinar en la cantidad y dificultad de la parte binaria puede llevar a que compense encargarse de la ramificación por violaciones de integralidad. Seguramente tenga relación con los puntos que comentaremos a continuación.

	GMT (45)	GMG (2)	GMN (43)	GMLT (43)	MRDLB (45)
Base *int-first* (45)	34.36	0.001	40.42	0.5357	0.1732
SL *int-first* (45)	31.77	0.001	41.49	0.5745	0.0000
CNDP *int-first* (45)	33.25	0.001	41.53	0.5863	0.0038
H1 *int-first* (45)	33.71	0.001	41.56	0.5978	0.0028
HL *int-first* (45)	36.17	0.001	41.56	0.6135	0.0028
Base int. rel. (43)	198.70	0.013	21.31	7.9008	1.0000
SL int. rel. (43)	100.30	0.006	21.68	3.8244	1.0000
CNDP int. rel. (44)	106.67	0.006	21.71	3.9245	1.0000
H1 int. rel. (44)	100.87	0.006	21.67	3.8432	1.0000
HL int. rel. (44)	99.79	0.006	21.67	3.8309	1.0000

Tabla 3.9: Resultados computacionales de `v16_k2_g5_d04000_p010_bin25` omitiendo *RLT-first*.

- **La proporción del número de nodos entre la resolución de relajaciones enteras e *integrality-first* es del orden de 2**: parece que tiene sentido teniendo en cuenta que a medida que disminuye el número de binarias es un valor que ha de tender a 1.
- **La proporción del tiempo lineal entre la resolución de relajaciones enteras e *integrality-first* en las linealizaciones de binarias es del orden de 10^{-1}**: lo cual no es una ganancia significativa respecto a la reducción relativa del número de nodos.

  Adicionalmente, es necesario poner de manifiesto lo siguiente.

- **El impacto de la reformulación de binarias es prácticamente nulo**: al tener tan pocas variables binarias, la parte binaria no puede ser tan complicada, especialmente al tratarse de un problema de grado 5.

En conjunto, esta primera batería deja claro que la elección de la estrategia para tratar las variables enteras no puede decidirse únicamente a partir del comportamiento agregado. Aunque la resolución de las relajaciones enteras por parte del solver auxiliar suele ofrecer mejores tiempos globales, esta ventaja parece depender de dos efectos contrapuestos: por un lado, reduce drásticamente el número de nodos respecto a gestionar explícitamente las violaciones de integralidad; por otro, cada nodo resulta mucho más costoso, ya que el tiempo de resolución de la relajación entera crece respecto al de la relajación continua conforme aumenta el peso de la parte binaria. Así, cuando hay muchas binarias, la reducción en nodos puede compensar sobradamente el mayor coste por nodo. Sin embargo, en configuraciones con baja densidad de variables continuas o con una parte binaria menos dominante, estas proporciones se equilibran hasta el punto de que la gestión propia de las violaciones de integralidad puede resultar competitiva, e incluso preferible.

También se observa que, al disminuir la complejidad de la parte binaria, se atenúan las diferencias entre las distintas reformulaciones de binarias. En esos casos, el ajuste adicional que proporcionan los patrones de linealización tiene un impacto cada vez menor sobre el rendimiento final, porque la estructura binaria deja de ser el principal factor limitante. Por otra parte, el buen comportamiento relativo de la versión base con *RLT-first* en algunos escenarios sugiere que no conviene imponer una prioridad rígida de ramificación: permitir que las violaciones RLT asociadas a productos de binarias guíen parte del proceso puede ser beneficioso cuando la linealización de binarias elimina esa información. En consecuencia, las lecciones aprendidas apuntan a que la estrategia más adecuada debe depender de la composición del problema —número de binarias, presencia de continuas y grado efectivo de la parte binaria—, más que de una regla uniforme válida para toda la batería.

Las tablas anteriores permiten identificar tendencias generales, pero no siempre explican por sí solas el motivo de las diferencias observadas. En particular, dos configuraciones pueden presentar tiempos de resolución similares y, sin embargo, generar árboles de ramificación muy distintos. Del mismo modo, una mejora en la cota inferior del nodo raíz no necesariamente se traduce en un menor número de iteraciones si el criterio de ramificación conduce a particiones menos efectivas.

Por este motivo, complementamos el estudio agregado con la representación gráfica de algunos árboles de ramificación. Estas figuras permiten visualizar cómo evoluciona la cota inferior a lo largo del proceso, qué tipo de ramificaciones se producen y en qué medida las distintas estrategias modifican la estructura del árbol.

3.5. Validación de ideas mediante representación gráfica

Tras este estudio computacional, se llevó a cabo un segundo estudio sobre una batería distinta, con el objetivo de comprobar si el comportamiento inesperado observado en presencia de pocas variables binarias era replicable. Los resultados obtenidos permitieron confirmar dicha replicabilidad. A fin de no sobrecargar el trabajo con resultados redundantes, vamos a profundizar un poco más los árboles que se producen en el proceso de ramificación y en cómo estos pueden estar afectando a los resultados obtenidos. Puesto que para cada problema disponemos de un archivo de texto que incluye el problema resuelto en cada iteración, junto con el valor de la función objetivo en la solución óptima de la relajación y la variable de ramificación seleccionada, podemos representar dicho proceso gráficamente.

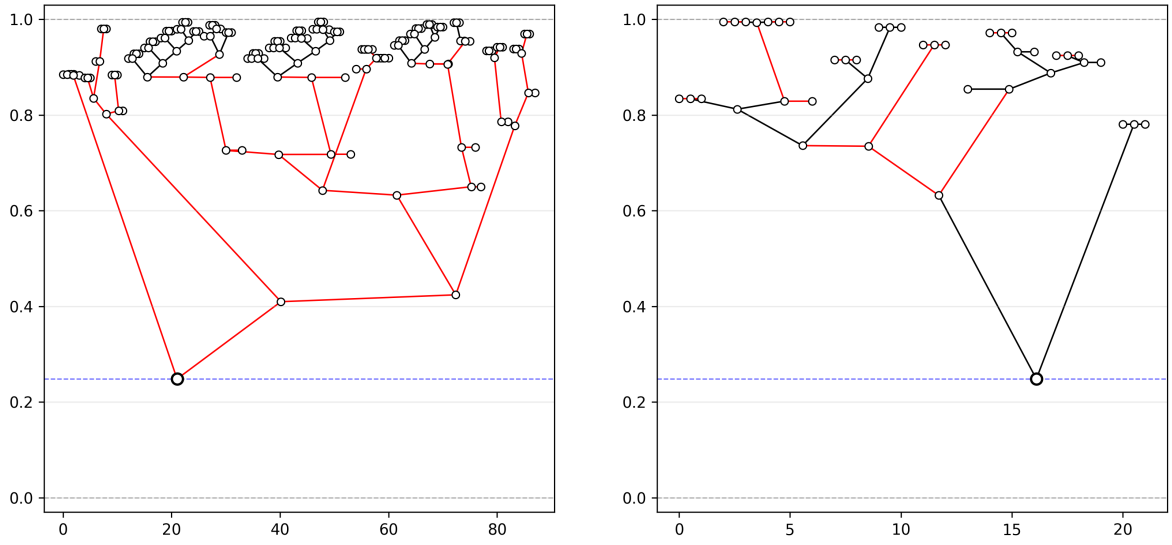
Para que sean fácilmente comparables, los valores de las funciones objetivo estarán normalizados entre 0 y 1, siendo 0 el valor de la solución óptima en el nodo raíz de la relajación continua de la linealización estándar, ya que es la formulación menos ajustada de la que disponemos, y el 1 el valor de la función objetivo en la solución del problema original. Indicaremos con una línea azul discontinua el valor de la función objetivo en el nodo raíz. Cada nodo representará un subproblema. Una arista roja representará una ramificación por una variable binaria, azul cuando se ramifique por una variable que sustituye a un producto de binarias en una linealización de binarias (como sabemos, sólo puede suceder en el enfoque *RLT-first*), mientras que en caso de ser negra será por una variable continua. Si se ramifica en un nodo, siempre serán construidos los nodos hijos. En caso de que no se llegue a resolver, se le asignará el mismo valor de la función objetivo que el del nodo padre.

3.5.1. `v18_k2_g5_d02000_p010_bin50_101` (base *int-first* vs base *RLT-first*)

Seleccionamos este problema para comparar el rendimiento entre las versiones base con las estrategias *integrality-first* y *RLT-first*, ya que la segunda mejora a la primera. En primer lugar, cabe destacar que mientras que *integrality-first* obtiene la solución óptima al resolver el último nodo visitado, *RLT-first* se detiene porque el gap relativo es inferior al umbral fijado. En ese aspecto, un gap más estricto habría perjudicado a la segunda estrategia de ramificación. En este ejemplo concreto, los tiempos lineales medios de las configuraciones comparadas son prácticamente iguales. Por tanto, la diferencia en el tiempo total de resolución no parece venir determinada por el coste medio de resolver cada relajación lineal, sino por el número de iteraciones realizadas. En otras palabras, en esta instancia la mejora observada se explica principalmente por una reducción del tamaño del árbol de ramificación.

En la Figura 3.1 vemos representados ambos árboles, y apreciamos que el enfoque que prioriza las violaciones RLT funciona mejor. Inicialmente, sospechábamos que pudiera ser consecuencia de la forma en la que se generan los problemas. Puesto que los coeficientes de la función objetivo se generan en un rango uniforme, ramificar por una violación RLT de una variable continua en las primeras iteraciones tendrá un impacto más significativo sobre la cota inferior del problema que el que puede tener una variable binaria, cuyo rango está entre 0 y 1.

Aunque en efecto se constata que la cota inferior evoluciona más rápido, vemos que en *RLT-first* se realiza una ramificación por una variable binaria ya en la segunda iteración. Aunque inicialmente podría pensarse que esta ramificación es accidental ante la ausencia de violaciones RLT en la segunda iteración, conviene tener en cuenta que al no aplicar una linealización de binarias, para las variables que sustituyen al producto de binarias también se miden las violaciones RLT, y por tanto pueden ser seleccionadas por dicha estrategia de ramificación. De esta forma, en el único caso en que se ramifica por una violación de integralidad es cuando el producto de las binarias coincide con las identidades RLT, pero estas no toman valores en $\{0,1\}$, lo cual será poco frecuente. En este aspecto, aquí *RLT-first*



(a) Representación para base *integrality-first*.

(b) Representación para base *RLT-first*.

Figura 3.1: Representación de los árboles de ramificación de `v18_k2_g5_d02000_p010_bin50_101`.

sería lo más parecido a no priorizar ninguno de los tipos de ramificación.

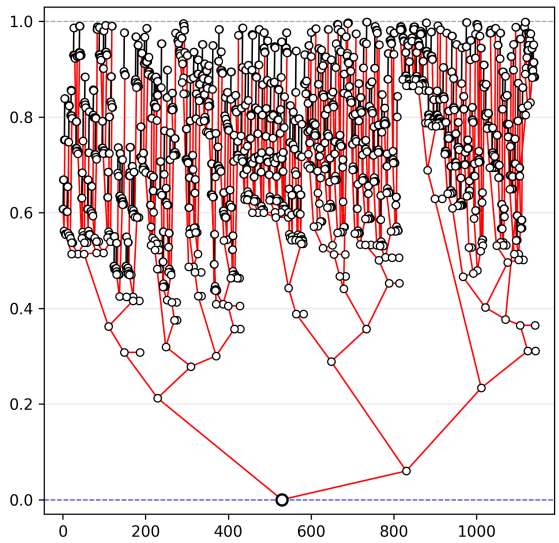
3.5.2. `v20_k2_g5_d02000_p010_bin50_102` (linealización estándar *int-first* vs linealización estándar resolviendo relajaciones enteras)

Comenzamos mostrando un caso que ilustra el comportamiento más habitual observado al comparar la estrategia *integrality-first* con la resolución de relajaciones enteras. En este tipo de instancias, cuando la cantidad de variables binarias es suficientemente alta, resolver relajaciones enteras suele producir árboles de ramificación más pequeños, ya que evita explorar nodos cuya solución incumple directamente las restricciones de integralidad.

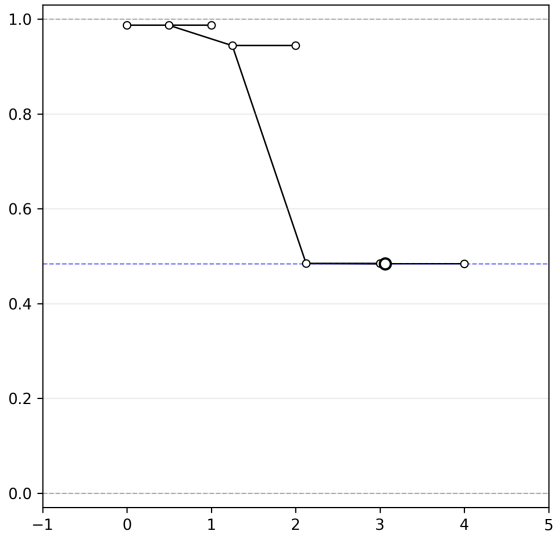
A la vista de la Figura 3.2, se observa que el árbol de ramificación obtenido al resolver relajaciones continuas y gestionar después las violaciones de integralidad es mucho más complejo que el obtenido al resolver directamente relajaciones enteras. En particular, nótese la cantidad de ramificaciones por violaciones de integralidad que se producen en *integrality-first* por debajo del valor de la función objetivo en el nodo raíz de la relajación entera. Esto indica que dichas soluciones ni siquiera son factibles para la relajación entera, por lo que buena parte del árbol se dedica a recuperar una factibilidad que la segunda estrategia impone desde el principio.

3.5.3. `v16_k2_g5_d04000_p010_bin25_128` (linealización estándar *int-first* vs linealización estándar resolviendo relajaciones enteras)

El comportamiento anterior es el más habitual cuando la parte binaria tiene un peso relevante en el problema. Sin embargo, no todas las instancias siguen este patrón. Un caso especialmente llamativo aparece en `v16_k2_g5_d04000_p010_bin25_128`, donde la estrategia *integrality-first* no presenta el deterioro esperado frente a la resolución de relajaciones enteras. Por este motivo, mostramos a continuación los árboles de ramificación y acotación correspondientes a ambas configuraciones. A diferencia del caso anterior, en la Figura 3.3 no se aprecia una ventaja tan clara al resolver relajaciones enteras. Esto sugiere que, cuando la proporción de variables binarias es menor, imponer la integralidad desde cada relajación puede no compensar necesariamente el coste adicional asociado. En estos casos, el

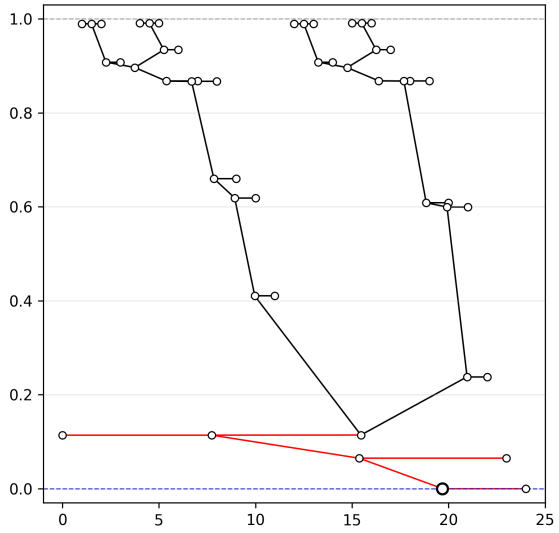


(a) Representación para *integrality-first*.

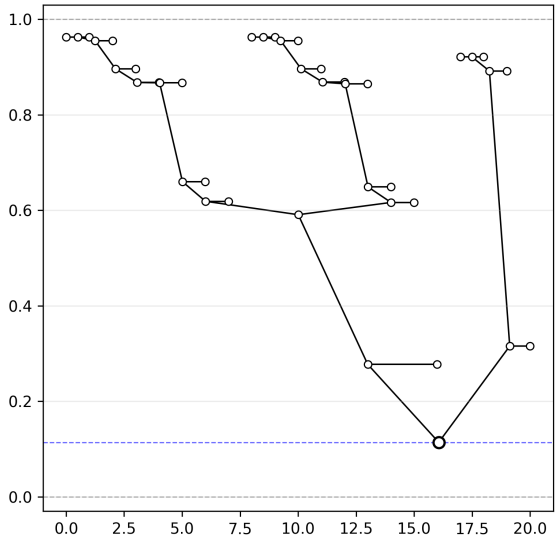


(b) Representación resolviendo relajaciones enteras.

Figura 3.2: Representación de los árboles de ramificación de la linealización estándar en `v20_k2_g5_d02000_p010_bin50_102`.



(a) Representación para *integrality-first*.



(b) Representación resolviendo relajaciones enteras.

Figura 3.3: Representación de los árboles de ramificación de la linealización estándar en `v16_k2_g5_d04000_p010_bin25_128`.

número de nodos no es el único factor determinante: también debe tenerse en cuenta el tiempo medio necesario para resolver cada relajación.

Capítulo 4

Posibles extensiones futuras

Durante mi colaboración en RAPOSa han surgido diversas líneas de trabajo que amplían algunos de los resultados desarrollados en los capítulos anteriores. Varias de ellas cuentan ya con resultados teóricos preliminares, y por ello dedicamos este último capítulo a presentarlas.

Comenzaremos revisando los resultados del Capítulo 3 para proponer una reformulación del producto de binarias linealizado con la parte no binaria del monomio. Seguidamente, ponemos la mirada sobre los criterios de ramificación discutidos en el Capítulo 2. A continuación, mencionaremos cómo se podría mejorar el comportamiento actual de RAPOSa respecto de las variables que sustituyen a monomios de binarias. Por último, presentaremos la posibilidad de considerar un conjunto distinto a los J-sets sobre los que construir las restricciones bound factor.

4.1. Linealización de la parte no binaria por la parte binaria

Los resultados computacionales del capítulo anterior muestran que la reformulación de productos de variables binarias puede modificar de forma significativa el comportamiento del algoritmo. Sin embargo, incluso después de linealizar la parte binaria de un monomio, todavía puede ser necesario construir restricciones RLT que involucren conjuntamente la parte binaria y la parte no binaria. Esto motiva la siguiente pregunta: si una parte del monomio ya ha sido sustituida por una variable binaria o por una variable que representa un producto de binarias, ¿es posible aprovechar esta estructura para reducir el número de restricciones bound factor que se construyen?

La idea de esta sección consiste precisamente en tratar la parte binaria como un elemento que activa o desactiva el producto no binario. De este modo, en lugar de construir directamente las restricciones bound factor sobre el monomio completo, se construyen sobre su parte no binaria y se añaden restricciones adicionales que vinculan dicha parte con la variable binaria correspondiente.

Volviendo de nuevo a la Proposición 2.32, conviene poner de manifiesto las condiciones que se imponen en las restricciones de tipo bound factor.

Corolario 4.1. *Las restricciones bound factor asociadas a $J \cup \{i\}$, donde y_i es una variable binaria o una variable que sustituye al producto de binarias, son equivalentes al siguiente conjunto de restricciones:*

- *Las restricciones bound factor de J.*
- $X_{I\cup\{i\}} = X_I y_i \quad \forall I \subset J.$

Este corolario muestra que las restricciones bound factor de un monomio que contiene una parte binaria pueden descomponerse en dos elementos: las restricciones asociadas a la parte no binaria y un conjunto de restricciones que relacionan dicha parte con la variable binaria.

Puesto que podemos no estar interesados en trabajar con las variables $X_{I\cup\{i\}}$ en caso de que no aparezcan explícitamente en el problema, el comportamiento de la variable que sustituye al producto de binarias en la reformulación de binarias nos permite linealizar la parte no binaria con la parte binaria. Proponemos entonces la siguiente formulación.

Proposición 4.2. *Consideremos el conjunto de monomios sobre los que se construirían las restricciones bound factor. En lugar de construirlas directamente, procedemos de la siguiente forma:*

- *Construimos los J-sets considerando únicamente la parte no binaria de los monomios.*
- *Para cualquier monomio $I \cup \{i\}$, donde y_i es una variable binaria o una variable que sustituye al producto de binarias, añadimos las siguientes restricciones:*
  - $X_{I\cup\{i\}} \geq \left(\prod_{j\in I} l_j\right) y_i$.
  - $X_{I\cup\{i\}} \leq \left(\prod_{j\in I} u_j\right) y_i$.
  - $X_{I\cup\{i\}} \leq X_I - \left(\prod_{j\in I} l_j\right)(1-y_i)$.
  - $X_{I\cup\{i\}} \geq X_I - \left(\prod_{j\in I} u_j\right)(1-y_i)$.

De esta forma, no sólo reducimos las restricciones bound factor cuando $|J|$ crece (ya que el número de restricciones bound factor asociadas a $J \cup \{i\}$ es el doble que las de J), sino que, como se discute en la rutina de generación de J-sets, quedarnos con la parte no binaria de los monomios favorecerá la aparición de relaciones de inclusión entre monomios, haciendo que se reduzca todavía más el número de restricciones.

Si bien la anterior transformación se puede aplicar sobre la versión base de RAPOSa sin más que eliminar una a una las variables binarias del monomio, a partir de ahora nos centramos en la situación donde se ha aplicado previamente una linealización de binarias. Haciendo una analogía al esquema que presentábamos al inicio del Capítulo 2, en la Figura 4.1 podemos visualizar el esquema de reformulaciones que consideraremos, y los resultados a probar también serán análogos.

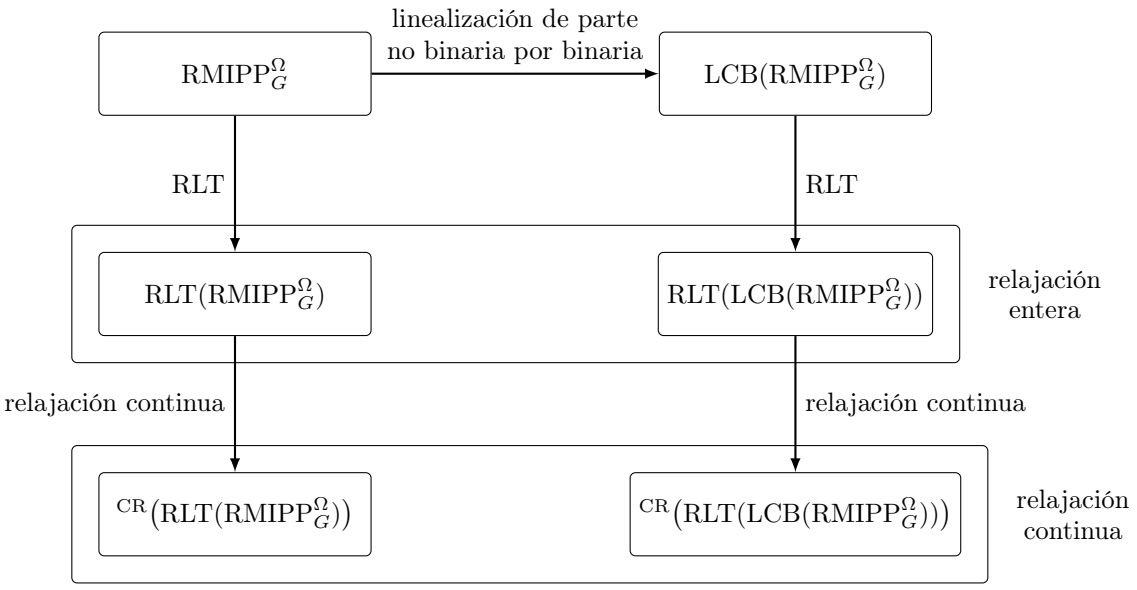


Figura 4.1: Esquema de reformulación y relajaciones consideradas.

De manera análoga al Capítulo 2, se verifican los siguientes resultados:

- Los problemas $RMIPP_G^\Omega$ y $LCB(RMIPP_G^\Omega)$ son equivalentes.
- La relajación $RLT(RMIPP_G^\Omega)$ es al menos tan ajustada como $RLT(LCB(RMIPP_G^\Omega))$, y lo mismo ocurre con sus relajaciones continuas.
- $RLT(RMIPP_G^\Omega)$ y $RLT(LCB(RMIPP_G^\Omega))$ son equivalentes.

No obstante, como el primer y el tercer resultado se pueden probar fácilmente a partir del Corolario 4.1, únicamente nos centraremos en el segundo punto.

Dado un problema reformulado como $RMIPP_G^\Omega$, la reformulación obtenida al linealizar la parte no binaria por la parte binaria se define como:

$$
\begin{array}{rll}
\text{minimizar} & [\phi_0(\boldsymbol{x})]_{\text{BR}} & \\
\text{sujeto a} & [\phi_r(\boldsymbol{x})]_{\text{BR}} \geq \beta_r, & r = 1, \ldots, R_1, \\
& [\phi_r(\boldsymbol{x})]_{\text{BR}} = \beta_r, & r = R_1 + 1, \ldots, R, \\
& X_v \leq X_s, & \forall v \in V,\ \forall s \in S_G(v) \\
& X_v \geq 1 + \displaystyle\sum_{s \in S_G(v)} (X_s - 1), & \forall v \in V,\ v \text{ no terminal}, \\
& 0 \leq X_v, & \forall v \in V, \\
& \left.\begin{array}{l}
\displaystyle\prod_{j \in J} x_j \cdot X_v \geq \prod_{j \in J} l_j X_v, \\
\displaystyle\prod_{j \in J} x_j \cdot X_v \leq \prod_{j \in J} u_j X_v, \\
\displaystyle\prod_{j \in J} x_j \cdot X_v \leq \prod_{j \in J} x_j - \prod_{j \in J} l_j (1 - X_v), \\
\displaystyle\prod_{j \in J} x_j \cdot X_v \geq \prod_{j \in J} x_j - \prod_{j \in J} u_j (1 - X_v),
\end{array}\right\} & \begin{array}{l} \forall J, v:\ |J| \geq 1, \\ J \cup v \text{ monomio del problema}, \end{array} \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^{|N|}. &
\end{array}
$$

$$(LCB(RMIPP_G^\Omega))$$

Ejemplo 4.3. Si recuperamos el ejemplo que teníamos en el Capítulo 2, su formulación tras haber linealizado los productos de variables binarias era:

$$
\begin{array}{rl}
\text{minimizar} & x_1 x_2 y_3 - 5 x_1 x_2^2 + 3 x_2 Y_{\{3,4\}} - x_2 y_4 \\
\text{sujeto a} & 2 Y_{\{3,4\}} - y_4 \geq 1, \\
& Y_{\{3,4\}} \leq y_3, \\
& Y_{\{3,4\}} \leq y_4, \\
& Y_{\{3,4\}} \geq y_3 + y_4 - 1, \\
& Y_{\{3,4\}} \geq 0, \\
& (\boldsymbol{x}, \boldsymbol{y}) \in [1,2] \times [3,4] \times \{0,1\}^2.
\end{array}
\tag{4.1}
$$

Para resolver el problema con nuestro algoritmo, todavía es necesario sustituir los monomios por variables RLT y añadir las restricciones bound factor. Así, las formulaciones que se resuelven son de la forma

$$
\begin{array}{rll}
\text{minimizar} & [[\phi_0(\boldsymbol{x})]_{\text{BR}}]_L & \\
\text{sujeto a} & [[\phi_r(\boldsymbol{x})]_{\text{BR}}]_L \geq \beta_r, & r = 1, \ldots, R_1, \\
& [[\phi_r(\boldsymbol{x})]_{\text{BR}}]_L = \beta_r, & r = R_1 + 1, \ldots, R, \\
& X_v \leq X_s, & \forall v \in V,\ \forall s \in S_G(v) \\
& X_v \geq 1 + \displaystyle\sum_{s \in S_G(v)} (X_s - 1), & \forall v \in V,\ v \text{ no terminal}, \\
& 0 \leq X_v, & \forall v \in V, \\
& \left.\begin{array}{l} X_{J\cup v} \geq \displaystyle\prod_{j\in J} l_j X_v, \\ X_{J\cup v} \leq \displaystyle\prod_{j\in J} u_j X_v, \\ X_{J\cup v} \leq X_J - \displaystyle\prod_{j\in J} l_j(1 - X_v), \\ X_{J\cup v} \geq X_J - \displaystyle\prod_{j\in J} u_j(1 - X_v), \end{array}\right\} & \begin{array}{l} \forall J, v :\ |J| \geq 1, \\ J \cup v \text{ monomio del problema}, \end{array} \\
& \left[\displaystyle\prod_{j\in J_1} (x_j - l_j) \prod_{j\in J_2} (u_j - x_j)\right]_L \geq 0, & \begin{array}{l} \forall J_1, J_2 :\ J_1 \cup J_2 = J, \\ J \text{ monomio del problema reformulado} \\ \quad \text{(restringido a la parte no binaria)} \end{array} \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^{|N|}. & \\
& & (RLT(LCB(RMIPP_G^\Omega)))
\end{array}
$$

Ejemplo 4.4. Así, el problema de ejemplo se reformula como:

$$
\begin{array}{rll}
\text{minimizar} & X_{\{1,2,3\}} - 5X_{\{1,2,2\}} + 3X_{\{2,(3,4)\}} - X_{\{2,4\}} & \\
\text{sujeto a} & 2Y_{\{3,4\}} - y_4 \geq 1, & \\
& Y_{\{3,4\}} \leq y_3, \qquad Y_{\{3,4\}} \leq y_4, & (LCB(RMIPP_G^\Omega)) \\
& Y_{\{3,4\}} \geq y_3 + y_4 - 1, \qquad Y_{\{3,4\}} \geq 0, & \\
& (\boldsymbol{x}, \boldsymbol{y}) \in [1,2] \times [3,4] \times \{0,1\}^2. &
\end{array}
$$

donde además hay que añadir la descomposición de los siguientes productos:

- $\{1,2,3\}$ en $\{1,2\}$ y $\{3\}$:

$$
\begin{cases}
X_{\{1,2,3\}} \geq 3y_3, \\
X_{\{1,2,3\}} \leq 8y_3, \\
X_{\{1,2,3\}} \leq X_{\{1,2\}} - 3(1 - y_3), \\
X_{\{1,2,3\}} \geq X_{\{1,2\}} - 8(1 - y_3).
\end{cases}
$$

- $\{2,(3,4)\}$ en $\{2\}$ y $\{(3,4)\}$:

$$
\begin{cases}
X_{\{2,(3,4)\}} \geq 3Y_{\{3,4\}}, \\
X_{\{2,(3,4)\}} \leq 4Y_{\{3,4\}}, \\
X_{\{2,(3,4)\}} \leq x_2 - 3(1 - Y_{\{3,4\}}), \\
X_{\{2,(3,4)\}} \geq x_2 - 4(1 - Y_{\{3,4\}}).
\end{cases}
$$

- $\{2,4\}$ en $\{2\}$ y $\{4\}$:

$$\begin{cases} X_{\{2,4\}} \geq 3y_4, \\ X_{\{2,4\}} \leq 4y_4, \\ X_{\{2,4\}} \leq x_2 - 3(1-y_4), \\ X_{\{2,4\}} \geq x_2 - 4(1-y_4). \end{cases}$$

Por último, se añaden también las restricciones bound factor asociadas a $\{1,2,2\}$ (las de $\{1,2\}$ no son necesarias porque quedan implicadas por las anteriores):

$$\begin{cases} -X_{\{1,2,2\}} + 8X_{\{1,2\}} + 2X_{\{2,2\}} - 16x_1 - 16x_2 + 32 \geq 0, \\ X_{\{1,2,2\}} - 7X_{\{1,2\}} - 2X_{\{2,2\}} + 12x_1 + 14x_2 - 24 \geq 0, \\ -X_{\{1,2,2\}} + 6X_{\{1,2\}} + 2X_{\{2,2\}} - 9x_1 - 12x_2 + 18 \geq 0, \\ X_{\{1,2,2\}} - 8X_{\{1,2\}} - X_{\{2,2\}} + 16x_1 + 8x_2 - 16 \geq 0, \\ -X_{\{1,2,2\}} + 7X_{\{1,2\}} + X_{\{2,2\}} - 12x_1 - 7x_2 + 12 \geq 0, \\ X_{\{1,2,2\}} - 6X_{\{1,2\}} - X_{\{2,2\}} + 9x_1 + 6x_2 - 9 \geq 0. \end{cases}$$

Veamos entonces la relación entre los ajustes de las distintas reformulaciones. Para ello, primero necesitamos el siguiente lema, que nos permite dar una cota para las variables RLT a partir de las restricciones bound factor.

Lema 4.5. *Las restricciones bound factor asociadas al monomio J imponen las siguientes cotas para las variables RLT:*

$$\prod_{j\in I} l_j \leq X_I \leq \prod_{j\in I} u_j, \quad \forall I \subset J.$$

La demostración del anterior resultado está recogida en el Teorema 1 de González-Díaz et al. (2025). Con él, ya es posible probar la relación entre el ajuste de ambas relajaciones continuas.

Proposición 4.6. *La relajación continua asociada a una linealización de binarias (linealización estándar o patrón de linealización) es al menos tan ajustada como la nueva reformulación asociada a dicha linealización.*

Demostración. Sea $J \cup v$ un monomio para el que se construyen las restricciones bound factor en $RLT(RMIPP^{\Omega}_{G})$, donde J es un monomio y v es una variable binaria o una variable que sustituye al producto de binarias. Entonces, debemos comprobar que, para cualquier $I \subset J$, de las restricciones bound factor asociadas a $J \cup v$ se deducen las siguientes restricciones:

- Las restricciones bound factor de J.
- $X_{I\cup v} \leq (\prod_{j\in I} u_j) X_v$.
- $X_{I\cup v} \geq (\prod_{j\in I} l_j) X_v$.
- $X_{I\cup v} \leq X_I - (\prod_{j\in I} l_j)(1 - X_v)$.
- $X_{I\cup v} \geq X_I - (\prod_{j\in I} u_j)(1 - X_v)$.

El primer punto se deduce directamente del Lema 1.16, así que veamos las demás. Además, por el mismo lema también podemos considerar únicamente el caso en que $I = J$, ya que las restricciones bound factor de $J \cup v$ implican las de $I \cup v$. En virtud de las cotas obtenidas en el lema previo:

- Multiplicando las cotas de J por X_v, tenemos:

$$\Big(\prod_{j\in J} l_j\Big) X_v \leq X_J \cdot X_v \leq \Big(\prod_{j\in J} u_j\Big) X_v.$$

  Linealizando, se tiene:

$$\Big(\prod_{j\in J} l_j\Big) X_v \leq X_{J\cup v} \leq \Big(\prod_{j\in J} u_j\Big) X_v.$$

- Multiplicando las cotas de J por $(1 - X_v)$, tenemos:

$$\Big(\prod_{j\in J} l_j\Big)(1 - X_v) \leq X_J(1 - X_v) \leq \Big(\prod_{j\in J} u_j\Big)(1 - X_v).$$

Linealizando y multiplicando por -1:

$$-\Big(\prod_{j\in J} u_j\Big)(1 - X_v) \leq X_{J\cup v} - X_J \leq -\Big(\prod_{j\in J} l_j\Big)(1 - X_v),$$

y sumando X_J:

$$X_J - \Big(\prod_{j\in J} u_j\Big)(1 - X_v) \leq X_{J\cup v} \leq X_J - \Big(\prod_{j\in J} l_j\Big)(1 - X_v), \qquad \square$$

Observación 4.7. *Recordemos que al aplicar las linealizaciones de binarias, se deja de medir como una violación RLT $|X_v - x_i X_{v\setminus i}|$. Después de aplicar la linealización que acabamos de presentar, sería coherente dejar de medir como violación RLT $|X_{J\cup v} - X_v \cdot X_J|$. Pese a la falta de resultados computacionales, cabe esperar que esto resolvería los problemas presentes al aplicar RLT-first en la batería v5_gv1_b20_gb7_k2_d00200_p030, puesto que en ese caso no habría violaciones RLT, y presentaría el mismo comportamiento que integrality-first.*

4.2. Extensiones de las estrategias de ramificación

Las propuestas anteriores modifican la formulación del problema. En cambio, las ideas de esta sección se centran en el comportamiento del algoritmo de ramificación y acotación. Los resultados del Capítulo 3 muestran que la elección de la estrategia de ramificación puede tener un impacto muy distinto dependiendo de la estructura del problema y de la reformulación empleada. En particular, se observaron casos en los que resolver relajaciones enteras no era claramente superior y otros en los que priorizar un tipo concreto de violación podía condicionar de forma excesiva la evolución del árbol.

Por este motivo, planteamos dos posibles modificaciones. La primera consiste en evitar una prioridad rígida entre violaciones RLT y violaciones de integralidad. La segunda busca combinar las ventajas de resolver relajaciones continuas en las primeras etapas con las de imponer integralidad en fases posteriores del árbol.

4.2.1. No priorizar tipos de violación

A la vista de la mejora de rendimiento presente en la versión base con el enfoque *RLT-first* frente a *integrality-first*, al permitirse ramificar por binarias cuando hay violaciones RLT porque también las producen, cabe preguntarse si esa mejora de rendimiento también se podría trasladar a las linealizaciones de binarias. Para su implementación, lo que se hará es no establecer ninguna prioridad sobre el tipo de violación, sino asignar pesos a las violaciones para corregir la diferencia de escala entre las violaciones de integralidad y las violaciones RLT, y ramificar por la violación más grande, independientemente del tipo que sea.

4.2.2. Control de la profundidad de la ramificación entera

Una posibilidad que no ha sido considerada en el estudio computacional del presente trabajo, pero que se aborda en González-Díaz et al. (2024), es la posibilidad de empezar resolviendo las relajaciones continuas, y a partir de un determinado momento pasar a resolver las relajaciones enteras. Pese a que las conclusiones que allí extraen son que nunca es positivo frente a resolver las relajaciones enteras directamente desde el principio, podemos tener algunas objeciones sobre el desarrollo. Esto se debe a

que la estrategia que se considera de partida es *RLT-first*, de la que tenemos claro su pésimo rendimiento cuando se aplica sobre cualquier linealización de binarias. Además, parece haber una cierta confusión entre lo que significa la estrategia *integrality-first* y hacer que el solver auxiliar resuelva las relajaciones enteras, ya que en el segundo caso estas tienen libertad en cada nodo.

De esta forma, parecería más adecuado plantearse empezar empleando un enfoque *integrality-first*, para reducir la dificultad de las relajaciones enteras. Así, una vez empiecen a resolverse las relajaciones enteras, podría reflejarse en una reducción del tiempo lineal.

Evidentemente, los dos puntos anteriormente comentados podrían testearse de manera conjunta.

4.3. Profundizando en las variables que sustituyen a productos de binarias

En esta sección se analizan algunas cuestiones adicionales relacionadas con las variables introducidas al linealizar productos de variables binarias. Aunque estas variables auxiliares se incorporan habitualmente como continuas en la formulación linealizada, su interpretación como productos de binarias permite explotar propiedades estructurales que pueden resultar útiles tanto desde el punto de vista teórico como computacional. En particular, se estudia primero el papel que puede desempeñar su integralidad en las soluciones factibles y, posteriormente, se considera una estrategia de ramificación múltiple que aprovecha la relación entre una variable de linealización y sus sucesores dentro del patrón correspondiente.

4.3.1. Integralidad en las soluciones factibles

Cuando aplicamos una linealización de binarias, las variables que sustituyen al producto de las mismas se consideran continuas. No obstante, existen varias situaciones en las que nos resulta conveniente tener en cuenta su integralidad en las soluciones factibles.

Cuando gestionamos nosotros las violaciones de integralidad y no se priorizan las ramificaciones por estas, es posible que estas variables que sustituyen al producto de variables binarias se seleccionen porque estén produciendo una violación RLT.

Las técnicas de ajuste de cotas constituyen un componente fundamental en numerosos algoritmos de optimización global aplicados a problemas no lineales. Su objetivo es reducir el espacio de búsqueda mediante el refinamiento de las cotas de las variables, lo que da lugar a reformulaciones más ajustadas y a relajaciones lineales que no resultan más difíciles de resolver que las originales. Por esta razón, su uso está ampliamente extendido en la literatura. El ajuste de cotas se discute en detalle en González-Rodríguez et al. (2022) y González-Díaz et al. (2024), pero en este trabajo no se aborda ante la falta de resultados computacionales empleándolo.

Respecto a las variables que sustituyen a binarias, si imponemos su integralidad, en cuanto se obtenga una cota inferior por encima de 0 (análogamente una cota superior por debajo de 1), ya podemos fijar la cota inferior de la variable a 1 (respectivamente, a 0), y así dejarla completamente determinada.

4.3.2. Ramificación múltiple

Acabamos de mencionar que en la ramificación por variables de la linealización de binarias v se puede fijar su valor a 0 o a 1. Nótese que en la rama en la que la variable vale 1, se le impone al resto de sucesores del patrón de linealización que valgan 1, mientras que en la rama en la que dicha variable se fija a 0 todavía existe libertad en sus sucesores. Para mitigar dicha disparidad en la dificultad de las ramas, podemos tener en cuenta que si esa variable vale 0 es porque al menos uno de los sucesores vale 0: $\exists s \in S^+(v) : X_s = 0$. Así, en lugar de ramificar en dos, podríamos considerar $1 + |S^+(v)|$ ramas:

- $v = 1,\ X_s = 1 \quad \forall s \in S^+(v)$.

- Para cada $s \in S^+(v),\ X_v = 0,\ X_s = 0,\ X_r$ libre $\forall r \in S^+(v) \setminus \{s\}$.

4.4. Reducción del número de restricciones bound factor más allá de los J-sets: Súper J-sets

La rutina de construcción de J-sets busca reducir el número de restricciones bound factor evitando construir restricciones redundantes. No obstante, el conjunto de monomios seleccionado por esta rutina no tiene por qué ser óptimo desde el punto de vista del número total de restricciones. En algunos casos, construir restricciones sobre un monomio mayor puede implicar simultáneamente las restricciones de varios monomios más pequeños y, aun así, requerir menos restricciones en total.

Dado un monomio $J = x_1^{p(1)} \cdots x_n^{p(n)}$, el número de restricciones bound factor asociadas es:

$$\prod_{i=1}^{n}(p(i)+1).$$

Sea $J_{\max} = \{J_1, \ldots, J_r\}$ el conjunto de monomios sobre el que hay que construir restricciones bound factor. El número de restricciones bound factor será entonces

$$\sum_{k=1}^{r}\prod_{i=1}^{n}(p_k(i)+1).$$

Si bien está claro que este número es inferior al que se obtendría construyendo las restricciones bound factor de $J_1 \cup J_2 = N^\delta$, es posible reducir el número de restricciones bound factor si cambiamos el conjunto de monomios para los que las construimos. Vamos a ilustrarlo con el siguiente ejemplo:

Ejemplo 4.8. Supongamos que tenemos que construir las restricciones bound factor de los monomios $x_1x_2x_3x_4$, $x_1x_2x_3x_5$, $x_1x_2x_4x_5$, $x_1x_3x_4x_5$ y $x_2x_3x_4x_5$. Entonces, el número de restricciones bound factor será $5 \cdot 2^4$. Sin embargo, si construimos las restricciones bound factor de su mínimo común múltiplo, esto es, $x_1x_2x_3x_4x_5$, el número de restricciones bound factor asociado será 2^5, que es claramente inferior.

Lo anterior motiva el siguiente desarrollo.

Sea T el conjunto de mínimos comunes múltiplos de elementos de $J_{\max}$. Sobre T se puede establecer una relación de orden parcial de divisibilidad. Dado un monomio $I \in T$, se define el número de restricciones bound factor asociadas a I como

$$w(I) = \prod_{i=1}^{n}(p_I(i)+1).$$

Lo que queremos es escoger un subconjunto S de T de forma que todos los elementos de $J_{\max}$ estén cubiertos (es decir, que cada elemento de $J_{\max}$ divida a algún elemento de S) de forma que haya que construir el menor número de restricciones bound factor posibles:

$$\min_{S \subset T} \sum_{I \in S} w(I) \quad \text{s.a. } \forall J \in J_{\max}\ \exists I \in S : J|I.$$

Lo anterior se puede formular como un problema lineal con variables enteras. Sea $x_I \in \{0,1\}$ la variable que indica si $I \in S$. Se tiene entonces:

$$\begin{aligned} &\min \sum_{I \in T} w(I)x_I \\ &\text{s.a.} \sum_{I \in T : J|I} x_I \geq 1 \quad \forall J \in J_{\max} \end{aligned}$$

Proposición 4.9. *La relajación inducida por la formulación anterior es al menos tan ajustada como la que emplea los J-sets.*

Demostración. En virtud del Lema 1.16, las restricciones construidas implican las que se obtienen al emplear los J-sets. □

Observación 4.10. *Una posible simplificación para reducir el espacio de búsqueda es dividir J_{max} en subconjuntos coprimos; es decir, si por ejemplo tenemos los monomios $x_1x_2x_3$ y x_4x_5, nunca compensará juntarlos en uno único, ya que la suma de los números de restricciones asociados será inferior al producto que resultaría al unirlos.*

Sea $J_{max} = \bigsqcup_{a=1}^{t} C_a$ bloques de monomios coprimos y sea T_a el conjunto de mínimos comunes múltiplos de C_a. Para cada $a = 1, \ldots, t$, se plantea el problema:

$$\begin{aligned} &\text{mín} \sum_{I \in T_a} w(I) x_I \\ &s.a. \sum_{I \in T_a : J|I} x_I \geq 1 \quad \forall J \in C_a \end{aligned}$$

y la solución global es la unión de las soluciones, luego el número total de restricciones es la suma de los números obtenidos en cada bloque.

Observación 4.11. *A la vista de que el problema de optimización planteado resulta muy costoso de resolver, es posible plantear la siguiente heurística. Definimos $C(I) = \{J \in J_{max} : J|I\}$ y aplicamos la siguiente heurística:*

- *$U = J_{max}$, $S = \emptyset$.*
- *Mientras $U \neq \emptyset$, tomamos $I^* = \arg\text{mín} \frac{w(I)}{|C(I) \cap U|}$.*
- *Actualizamos $S = S \cup \{I^*\}$ y $U = U \setminus C(I^*)$.*

Se quiso medir el impacto de estos súper J-sets sobre la versión actual de RAPOSa. Para ello, se generaron problemas de manera análoga a lo que se hizo en el Capítulo 3, considerando todas las variables continuas, puesto que la implementación de la construcción de los J-sets sobre la parte no binaria del problema se espera que presente un mejor rendimiento. Puesto que los tiempos de computación para el problema original son excesivamente elevados, hubo que recurrir a la heurística para resolver el problema. Después, denotando por J_S dicho conjunto, para cada $J \in J_S \setminus J_{\max}$, se añadió al problema la restricción

$$\prod_{j \in J} x_j \geq \prod_{j \in J} l_j,$$

con el único objetivo de que el monomio apareciera explícitamente en el problema. De esta forma, se obtiene un problema equivalente cuyo J-set coincide con J_S.

Pese a que se reduce ligeramente el número de restricciones (del 5 % al 10 %, aunque en varios casos quedan igual), el impacto que tiene en el número de variables es muy notable (en muchos casos, el incremento de estas es superior a la reducción del número de restricciones). Esto, aunque pudiera ser mitigado modificando ligeramente la función a optimizar en la heurística, se debe a que para cada monomio para el que se construyen las restricciones bound factor en el problema han de aparecer todos sus divisores, así que a medida que crece el grado de los monomios considerados, más variables adicionales aparecerán.

Esto supuso que el rendimiento fuera en todos los casos peor que el que presentaba el uso de los J-sets, y pese a tener relajaciones más ajustadas, en muchos casos, el número de iteraciones requerido fue superior. Esto seguramente se deba a que, además de comprobar las violaciones RLT para las variables RLT del problema original, el algoritmo también tiene que considerar todas las nuevas variables.

Pese a que todo parezca apuntar a que esta idea no supondrá una mejora de rendimiento, quizás podemos emplear el razonamiento de la Proposición 1 de Dalkiran y Sherali (2013) para justificar que no es necesario tener en cuenta las violaciones RLT de monomios que no aparecen en el problema original para que converja el algoritmo, y así obtener una reducción efectiva en el número de iteraciones.

Bibliografía

Achterberg, T, Koch, T, Martin, A (2005). Branching rules revisited. Operations Research Letters, 33, 1, 42-54.

Dalkiran, E, Ghalami, L (2018). On linear programming relaxations for solving polynomial programming problems. Computers & Operations Research, 99, 67-77.

Dalkiran, E, Sherali, HD (2012). Enhancing RLT-based relaxations for polynomial programming problems via a new class of υ-semidefinite cuts. Computational Optimization and Applications, 52, 483-506.

Dalkiran, E, Sherali, HD (2013). Theoretical filtering of RLT bound-factor constraints for solving polynomial programming problems to global optimality. Journal of Global Optimization, 57, 1147-1172.

Dalkiran, E, Sherali, HD (2016). RLT-POS: Reformulation-Linearization Technique-based optimization software for solving polynomial programming problems. Mathematical Programming Computation, 8, 337-375.

Elloumi, S, Verchère, Z (2023). Efficient linear reformulations for binary polynomial optimization problems. Computers & Operations Research, 155, 106240.

González-Díaz, J, González-Rodríguez, B, Rodríguez-Acevedo, I (2024). Extending a continuous RLT-based algorithm to mixed-integer polynomial problems.

González-Díaz, J, González-Rodríguez, B, Gómez-Casares, I (2025). Bound tightening in lifted formulations: (sub) solver-dependent impact on performance in RLT-based algorithms. arXiv preprint arXiv:2509.18731.

González-Rodríguez, B. (2022). Advances in Polynomial Optimization. Tesis, Universidade de Santiago de Compostela.

González-Rodríguez, B, Ossorio-Castillo, JJ, González-Díaz, J, González-Rueda, ÁM, Rodríguez-Penas, D, Rodríguez-Martinez, D (2022). Computational advances in polynomial optimization: RAPOSa, a freely available global solver. Journal of Global Optimization, 85, 541-568.

González-Rodríguez, B, Naoum-Sawaya, J (2025). Degree reduction techniques for polynomial optimization problems. European Journal of Operational Research, 322, 2, 401-413.

Sherali, HD, Adams, WP (2025). Reformulation–Linearization Techniques for Mixed-Discrete Optimization Problems. Handbook of Combinatorial Optimization, Springer, 1–54.

Sherali, HD, Tuncbilek, CH (1992). A global optimization algorithm for polynomial programming problems using a Reformulation–Linearization Technique. Journal of Global Optimization, 2, 101–112.