USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
FACULTADE DE MATEMÁTICAS

Trabajo de Fin de Grado

Reformulaciones en problemas de programación no lineal entera mixta

Sergio Álvarez Pérez

2023/2024

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRADO DE MATEMÁTICAS

Trabajo de Fin de Grado

Reformulaciones en problemas de programación no lineal entera mixta

Sergio Álvarez Pérez

Julio, 2024

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área del conocimiento: Estadística e investigación operativa.
Título: Reformulaciones en problemas de programación no lineal y entera mixta.
Breve descripción del contenido.
Este trabajo se encuadra dentro del campo de las técnicas de reformulación en problemas de programación matemática para la resolución de problemas de optimización no lineal con variables enteras. El principal objetivo es estudiar cómo distintas reformulaciones de variables binarias y variables enteras pueden ayudar al mejorar la eficiencia en la resolución de este tipo de problemas.
Recomendaciones.
Otras observaciones.

A mis padres, por ayudarme a subir cada escalón de mi carrera educativa.

A mi hermano, por velar siempre por mí y por ser mi avanzadilla.

Al resto de mi familia por hacerme sentir siempre arropado y apoyado.

A Julio, mi tutor, por su preocupación, apoyo y por introducirme al mundo de la investigación.

A Iria, mi cotutora, por su dedicación continua y orientación en la redacción de este trabajo.

Índice

Resumen

En este trabajo se estudia una clase de reformulaciones de problemas de programación polinómica. En primer lugar se presenta la base teórica de estas reformulaciones, ideadas para problemas de optimización polinómica con variables binarias. En segundo lugar se extienden los resultados en torno a estas reformulaciones al contexto más general de problemas de optimización polinómica mixtos, con variables binarias y no binarias (enteras y continuas). Por último, se presentan los resultados numéricos obtenidos al adaptar estas reformulaciones para su inclusión en un software de optimización.

Abstract

In this project we study a class of reformulations for polynomial programming problems. First, we present the theoretical basis for these reformulations, originally devised for binary polynomial optimization problems. Then we extend the results to the more general context of polynomial optimization problems, including non-binary variables (integer and continuous). Lastly, we show the numeric results obtained after adapting and implementing these reformulations in a solver.

Introducción

En este trabajo se estudia un cierto tipo de reformulaciones aplicables a problemas de programación polinómica.

El estudio de problemas de optimización es un campo muy amplio dentro de las matemáticas, y también uno de los más aplicables a problemas del mundo real. El término *programación matemática* se acuñó durante la segunda guerra mundial, periodo en el cual se utilizó para solucionar problemas de programación y planificación, como el transporte de bienes, asignación de recursos o calendarización; con el objetivo de reducir los costes del ejército. Desde ese momento se desarrollaron grandes avances en el campo. La programación matemática aparece en multitud de ocasiones en nuestro día a día. Por ejemplo, en la forma de dar el cambio en una compra para que se devuelva el mínimo número de monedas, en el orden en el que un repartidor debe entregar los paquetes para minimizar el tiempo que tarda o en la ruta que debe tomar un mensaje enviado por internet para minimizar el tiempo que tarda sin sobrecargar la capacidad de la red.

Dentro de la programación matemática, la programación polinómica trata con los problemas en los cuales las expresiones involucradas son polinomios. Los problemas clásicos de optimización suelen encuadrarse dentro de la programación lineal y entera, en los cuales todas las expresiones son lineales respecto a las variables del problema. El hecho de poder usar polinomios permite modelizar un mayor número de problemas, con la consecuencia de aumentar dificultad y complejidad de su resolución.

Con este trabajo se busca ampliar el conocimiento disponible respecto a las formas de resolver problemas de optimización polinómica, estudiando mecanismos de reformulación recientes que permiten transformar un cierto tipo de problemas en otro tipo, con propiedades deseables y con la intención de que se puedan resolver en un tiempo menor.

En el capítulo 1 se establecerá la base teórica de las reformulaciones a tratar, en su contexto original de problemas de optimización polinómica con variables binarias. En el capítulo 2 se plantearán nuevos resultados, extendiendo los del capítulo anterior al contexto más general de problemas de optimización polinómica generales, con la posible coexistencia de variables binarias con variables enteras generales y continuas. En el capítulo 3 se expondrá el trabajo realizado

para adaptar e implementar las reformulaciones en un software de optimización desarrollado por investigadores de la USC y del CITMAga, mostrando los resultados obtenidos. En el capítulo 4 se expondrán las conclusiones obtenidas.

Capítulo 1

Reformulación de problemas de optimización polinómica con variables binarias

En este capítulo se introducirán los conceptos que servirán de base de todo el trabajo. En primer lugar, se presentarán los problemas de programación polinómica con variables binarias. A continuación, se presentarán mecanismos para describir reformulaciones de este tipo de problemas. Por último, se presentarán dos métodos utilizados para obtener reformulaciones con propiedades deseables, que ayuden en la resolución del problema.

Los resultados de este capítulo son una adaptación de los planteados en [1].

1.1. Problemas de optimización polinómica con variables binarias (BPO)

En esta sección se establecerán definiciones relacionadas con los problemas de optimización polinómica con variables binarias que se utilizarán a lo largo de todo el trabajo. En primer lugar es necesario definir la representación de monomio y polinomio que se utilizará.

Definición 1.1. Dado un entero positivo n y un vector de variables binarias $\boldsymbol{y} = (y_i)_{i\in\{1,\dots,n\}}$, se define un *monomio binario* como un producto $\prod_{i\in M} y_i$, con $M \subset \{1,\dots,n\}$. Para los propósitos de este trabajo, la parte del monomio que es de interés es el conjunto M. Por tanto cuando se use el término monomio se hablará específicamente de su representación utilizando ese conjunto. En los monomios binarios cada variable solo puede aparecer con exponente 1, ya que $y_i^2 = y_i$, por tanto usar conjuntos es suficiente para representar cualquier monomio binario.

Definición 1.2. Dado un entero positivo n y un vector de variables binarias $\boldsymbol{y} = (y_i)_{i\in\{1,\dots,n\}}$, se define un *polinomio binario* como la suma $\sum_{M\in P} c_M \prod_{i\in M} y_i$, donde $P \subset \mathcal{P}(\{1,\dots,n\})$, los $M \in P$ representan monomios tal y como se ha definido en 1.1 y los $c_M \in \mathbb{R}$ representan a los coeficientes que acompañan a los monomios. De nuevo, en este trabajo la parte de interés del polinomio es el conjunto P, por tanto cuando se use el término polinomio se estará hablando de su representación utilizando este conjunto.

Una vez se han definido los monomios y polinomios binarios, se pueden definir los problemas de optimización polinómica con variables binarias.

Definición 1.3. Dado un entero positivo n, se define un problema de optimización polinómica con variables binarias como

$$\text{(BPO)} \quad \begin{cases} \underset{\boldsymbol{y}}{\text{minimizar}} & \displaystyle\sum_{M\in P} c_M \prod_{i\in M} y_i & \\ \text{sujeto a} & y_i \in \{0,1\} & \forall M \in P,\ \forall\, i \in M. \end{cases}$$

Se puede asumir sin pérdida de generalidad que $\{i\} \in P$ para todo $i \in \{1,\dots,n\}$, sin más que asignarle el coeficiente $c_{\{i\}} = 0$ si fuera necesario. Esta propiedad será útil más adelante.

Ejemplo 1.4. Para apoyar la explicación del capítulo, se plantea el siguiente ejemplo de problema de optimización polinómica con variables binarias. Este ejemplo se reutilizará más adelante para apoyar los nuevos conceptos presentados.

$$\begin{cases} \underset{\boldsymbol{y}}{\text{minimizar}} & y_1y_2y_3 + y_1y_3y_4 + 3y_3y_4y_5 & \\ \text{sujeto a} & y_i \in \{0,1\} & \forall\, i \in \{1,2,3,4,5\}. \end{cases}$$

El polinomio P asociado a este problema es

$$P = \{\{1,2,3\},\{1,3,4\},\{3,4,5\},\{1\},\{2\},\{3\},\{4\},\{5\}\}.$$

Como se explica en la definición 1.2, se puede asumir que los monomios y_1, y_2, y_3, y_4 e y_5 están todos presentes en la función objetivo, acompañados del coeficiente 0.

Definición 1.5. Dado un polinomio P y uno de sus monomios $M \in P$ se define el grado de M como su cardinalidad, $|M|$. Se define el grado de P como el máximo grado de todos los monomios $M \in P$.

Los problemas de tipo (BPO) son difíciles de resolver, no se conocen métodos que sean capaces de obtener una solución en tiempo polinómico. En particular, pertenecen a la clase de complejidad **NP-duro**. En [1] se hace referencia a trabajos en los que se exploran diversos

algoritmos utilizados para la resolución de este tipo de problemas. En algunos de estos trabajos se transforma el problema en uno equivalente de grado 2, conocido como *quadratic unconstrained binary optimization* (QUBO). En otros, se opta por una resolución directa, sin reformulaciones.

En este trabajo se estudiará un método de resolución basado en reformular el problema como un problema de programación lineal entera mixta. Obtener una reformulación del problema de esa forma permite utilizar mecanismos ya conocidos y estudiados (como ramificación y poda) para resolverlo. Más adelante se verá que existen muchas maneras de reformular un problema de tipo (BPO) como un problema de programación lineal entera mixta. Para introducir el concepto de reformulación, se partirá de una de esas maneras, la *linealización estándar*, que es una de las formas más simples de construir una reformulación de este tipo.

Definición 1.6. Se define la *linealización estándar* asociada a un (BPO) como:

$$
(\mathrm{SL}) \quad \begin{cases} \underset{\boldsymbol{Y}}{\text{minimizar}} & \displaystyle\sum_{M \in P} c_M Y_M & & \\ \text{sujeto a} & Y_M \leq y_i & \forall M \in P,\ \forall i \in M & (1.1) \\ & Y_M \geq 1 + \displaystyle\sum_{i \in M} (y_i - 1) & \forall M \in P & (1.2) \\ & Y_M \geq 0 & \forall M \in P & (1.3) \\ & y_i \in \{0,1\} & \forall i \in \{1,\dots,n\}. & \end{cases}
$$

Ejemplo 1.7. En este ejemplo se muestra la linealización estándar asociada al problema del ejemplo 1.4. Se pueden ver las tres variables utilizadas para reemplazar los monomios de la función objetivo y las nuevas restricciones del problema asociadas a estas variables. Al denotar las variables auxiliares, por claridad se cometerá el abuso de notación de omitir la notación de conjunto, por ejemplo denotando la variable $Y_{\{1,2,3\}}$ como Y_{123}.

$$
\begin{cases} \underset{\boldsymbol{Y}}{\text{minimizar}} & Y_{123} + Y_{134} + 3Y_{345} & \\ \text{sujeto a} & Y_{123} \leq y_1,\ \ Y_{123} \leq y_2,\ \ Y_{123} \leq y_3 & \\ & Y_{134} \leq y_1,\ \ Y_{134} \leq y_3,\ \ Y_{134} \leq y_4 & \\ & Y_{345} \leq y_3,\ \ Y_{345} \leq y_4,\ \ Y_{345} \leq y_5 & \\ & Y_{123} \geq y_1 + y_2 + y_3 - 2 & \\ & Y_{134} \geq y_1 + y_3 + y_4 - 2 & \\ & Y_{345} \geq y_3 + y_4 + y_5 - 2 & \\ & Y_{123} \geq 0\ \ Y_{134} \geq 0\ \ Y_{345} \geq 0 & \\ & y_i \in \{0,1\} & \forall i \in \{1,2,3,4,5\}. \end{cases}
$$

Una de las propiedades fundamentales de las reformulaciones es que son equivalentes al problema original, es decir, se puede obtener la solución de uno de los problemas a partir de la

solución del otro, y ambos valores óptimos coinciden. Esto es cierto en particular para la linealización estándar. Para demostrar esto se parte del siguiente lema.

Lema 1.8. *En el problema (SL), equivalen:*

1. $Y_M = \prod_{i \in M} y_i$ *para todo* $M \in P$.
2. *Se cumplan las restricciones* (1.1), (1.2) *y* (1.3).

Demostración. La demostración se apoya en el hecho de que las variables y_i son binarias. Para la primera implicación, partiendo de que $Y_M = \prod_{i \in M} y_i$ para todo $M \in P$:

- Para la restricción (1.1) se tiene que si $y_i = 1$, $Y_M \leq y_i$ trivialmente por ser Y_M producto de variables binarias. Si algún $y_i = 0$, entonces $Y_M = 0$, por tanto también se cumple la restricción.
- Para la restricción (1.2), si alguno de los $y_i = 0$, entonces $Y_M = 0$, por tanto la restricción que se tiene que verificar es $1 + \sum_{i \in M}(y_i - 1) \leq 0$. Esta se cumple al existir al menos un $y_i = 0$ y ser todas las variables y_i binarias. Por otro lado, si todos los $y_i = 1$, entonces $Y_M = 1$, luego la restricción que se tiene que verificar es $1 \geq 1 + \sum_{i \in M}(y_i - 1)$. Esto se cumple porque si $y_i = 1$ para todo $i \in M$ entonces $\sum_{i \in M}(y_i - 1) = 0$.
- Por último la restricción (1.3) se cumple por ser Y_M producto de variables binarias.

Ahora, para la segunda implicación, partiendo de que se cumplen las restricciones (1.1), (1.2) y (1.3). Para cada $M \in P$ y su correspondiente Y_M se plantean los siguientes casos:

- **Caso 1**: $y_i = 1$ para todo $i \in M$. Por un lado, por (1.1) se tiene que $Y_M \leq 1$. Por otro lado, por (1.2) se tiene que $Y_M \geq 1 + \sum_{i \in M}(y_i - 1) = 1$. Por tanto en este caso $Y_M = 1 = \prod_{i \in M} y_i$.
- **Caso 2**: existe un $i \in M$ tal que $y_i = 0$. Por un lado, por (1.1) se tiene que $Y_M \leq y_i = 0$. Por otro lado, por (1.3), se tiene que $Y_M \geq 0$. Por tanto en este caso $Y_M = 0 = \prod_{i \in M} y_i$.

Como los casos 1 y 2 abarcan todas las posibles combinaciones para los y_i, se concluye que $Y_M = \prod_{i \in M} y_i$. □

Una vez se ha demostrado el lema, se puede utilizar para demostrar la equivalencia entre problemas (BPO) y (SL).

Proposición 1.9. *Los problemas (BPO) y (SL) son equivalentes.*

Demostración. Para demostrar que ambos problemas son equivalentes, se verá que se puede obtener una solución para cada uno a partir de una solución del otro, y que en ambos casos los valores óptimos coinciden.

Dada una solución de (BPO), $\boldsymbol{y} = (y_1, \ldots, y_n)$, se puede construir una solución para (SL) tomando $Y_M = \prod_{i \in M} y_i$ para todo $M \in P$. Como se ha visto en el lema 1.8, se cumplirán las restricciones (1.1), (1.2) y (1.3), y el valor de la función objetivo en ambos problemas será el mismo. Recíprocamente, dada una solución de (SL), $\boldsymbol{Y} = (y_1, \ldots, y_n, Y_{M_1}, \ldots, Y_{M_{|P|}})$, bastará con tomar los valores de los y_i para la solución de (BPO). Gracias a las restricciones de (SL), se tendrá de nuevo por el lema 1.8 que $Y_M = \prod_{i \in M} y_i$ para todo $M \in P$ y, así, los valores de la función objetivo serán los mismos en ambos problemas. □

Como se dijo previamente, la reformulación de un problema de tipo (BPO) como problema de programación lineal entera mixta se utiliza para su resolución, valiéndose de los métodos conocidos para la resolución de este tipo de problemas. También se expuso previamente que la linealización estándar no es la única, y que existen muchas formas de reformular el problema original. Estos dos hechos en conjunto pueden conducir a la pregunta: ¿Existen unas reformulaciones *más fáciles de resolver* que otras?[1] La respuesta es afirmativa. Aunque todas las reformulaciones son equivalentes, el algoritmo que obtiene la solución al problema no tiene por qué dar ni los mismos pasos, ni la misma cantidad de pasos. Esto se debe al funcionamiento del algoritmo de ramificación y poda y a su uso de las relajaciones continuas de los problemas.

Para cada problema de programación lineal entera mixta, existe un problema asociado denominado relajación continua. Este es el problema resultante al eliminar las restricciones que obligan a las variables a ser enteras o binarias. La relajación continua es muy relevante en la resolución. En particular, para los problemas de minimización, permite dar una cota inferior del problema original, es decir, un valor que acota por debajo el óptimo del problema. Las cotas inferiores son cruciales, ya que dirigen la ramificación y dictan la poda, controlan completamente el espacio de soluciones que se explora. Sucede lo análogo en los problemas de maximización.

Para este proceso de ramificación y poda, es deseable que la cota inferior sea lo más cercana posible al valor óptimo del problema, lo cual se denomina ser *ajustada*. En general, se habla de ser ajustada en términos relativos. Se dice que una reformulación de un problema es más ajustada que otra si la relajación continua asociada a la reformulación ofrece una cota inferior más cercana al valor óptimo del problema que la otra.

[1] Entendiendo por facilidad el coste computacional al resolver el problema utilizando un cierto algoritmo especializado.

La reformulación utilizada hasta ahora, la linealización estándar, no tiene por qué ser de las más ajustadas. En [1] se puede encontrar un ejemplo no patológico en el cual la linealización estándar aporta una cota significativamente peor que otra linealización. Por la importancia que tiene ser ajustada, es relevante intentar buscar otras reformulaciones y compararlas en este aspecto. Por ello en la siguiente sección se presentará un método para generar diferentes reformulaciones y más adelante se estudiará cómo escoger aquellas que proporcionen una mejor cota.

1.2. Reformulaciones lineales de (BPO)

En esta sección se estudiarán las distintas formas de generar reformulaciones lineales de un (BPO) y las propiedades de estas reformulaciones. Para ello, primero se recordarán algunas definiciones sobre grafos que se utilizarán a lo largo del trabajo.

Definición 1.10. Un *grafo* es un conjunto de nodos V y un conjunto de aristas A, de forma que cada arista es un par nodos del grafo. Si el par es ordenado, se dice que el grafo es *dirigido*. De lo contrario se dice que es *no dirigido*.

Definición 1.11. Se dice que un grafo dirigido contiene un *ciclo* si existe una colección ordenada de dos o más vértices $(v_1, v_2, \ldots, v_n)$ de forma que hay una arista en todas las parejas contiguas de vértices y entre el vértice final v_n y el inicial v_1. Es decir, si $(v_1, v_2) \in A$, $(v_2, v_3) \in A$, $\ldots$, $(v_{n-1}, v_n) \in A$ y $(v_n, v_1) \in A$. Si un grafo dirigido no contiene ningún ciclo, se dice que es un grafo dirigido *acíclico*.

Definición 1.12. En un grafo dirigido $G = (V, A)$, se dice que un cierto nodo $v \in V$ es un nodo *terminal* si no hay aristas que salgan de él, es decir, si $(v, w) \notin A$ para todo $w \in V$.

Definición 1.13. Dado un grafo $G = (V, A)$, se define su función de descendientes directos δ_G^+ como:

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

Con estos conceptos establecidos se pueden definir los *patrones de linealización*, que serán esenciales a lo largo de todo el trabajo.

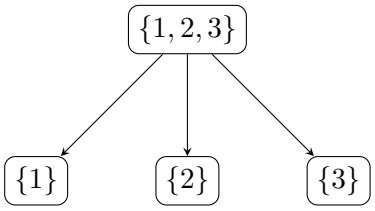
Definición 1.14. Dado un monomio M se define un *patrón de linealización* de M como un grafo dirigido acíclico $G_M = (V_M, A_M)$ que cumple las siguientes condiciones:

- El grafo tiene una raíz única: M.

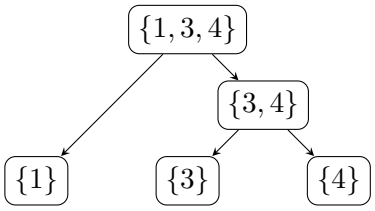
- Los nodos $v \in V_M$ son subconjuntos de M.
- Todos los conjuntos $\{i\}$ con $i \in M$ son nodos terminales del grafo, y son los únicos nodos terminales.
- Para todo nodo no terminal v se tiene que $v = \bigcup_{t \in \delta^+_{G_M}(v)} t$ y $t \subsetneq v$ para todo $v \in M$ y para todo $t \in \delta^+_{G_M}(v)$.

Los patrones de linealización permiten formalizar la manera en la que se reformulan las variables en un (BPO).

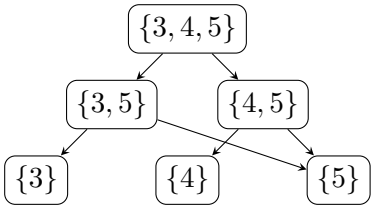
Ejemplo 1.15. En este ejemplo, para asentar la noción de patrón de linealización, se muestran varios ejemplos de posibles patrones de linealización para distintos monomios. En la figura 1.1 se tienen patrones asociados a los monomios con variables $\{1,2,3\}$, $\{1,3,4\}$ y $\{3,4,5\}$. Para este último monomio se muestran dos patrones, recalcando la idea de que existen multitud de patrones de linealización posibles para un monomio dado.



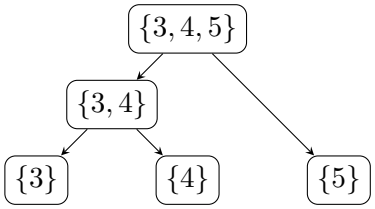
(a) Un patrón de linealización para $y_1y_2y_3$.



(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 1.1: Algunos posibles patrones para los monomios del problema del ejemplo 1.4.

Definición 1.16. Dado un monomio M y un patrón de linealización asociado al monomio $G_M = (V_M, A_M)$, se dice que este es *simple* si el grafo es un árbol, es decir, para todo nodo no terminal $v \in V_M$ se tiene que el conjunto $\delta^+_{G_M}(v)$ forma una partición de v.

Intuitivamente, que un patrón de linealización sea simple se puede entender como que ninguno de los nodos tenga dos padres. En la figura 1.1 se puede ver que los patrones 1.1a, 1.1b y 1.1d son simples, todos los descendientes directos de un nodo forman una partición del mismo. Sin embargo en el patrón 1.1c esto no sucede. En específico, los descendientes directos de $\{3,4,5\}$

son $\{3,5\}$ y $\{4,5\}$, que no forman una partición por no ser conjuntos disjuntos. Por tanto este no es un patrón simple.

En el contexto de este trabajo solamente se tendrán en cuenta patrones de linealización simples, llamándolos simplemente patrones si no hay posibilidad de confusión.

Una vez se ha introducido el concepto de patrón asociado a un monomio, se puede extender el planteamiento a polinomios. Para ello, se tomará un patrón por cada monomio y se realizará una operación entre ellos denominada concatenación.

Definición 1.17. Dados dos monomios denotados M_1, M_2 y dos patrones de linealización asociados $G_1 = (V_1, A_1)$, $G_2 = (V_2, A_2)$, se define la concatenación de los patrones como el grafo $G = \mathcal{C}(G_1, G_2) = (V, A)$, cumpliendo:

- V es la unión de los nodos de V_1 y V_2. Si un nodo aparece en ambos grafos, se considera una sola vez si y solo si el subgrafo a partir de ese nodo es igual en ambos grafos. Si no son iguales, se numeran arbitrariamente los nodos para diferenciarlos.

- A es la unión de los conjuntos de aristas A_1 y A_2, modificando las aristas necesarias si se numeran nodos según el apartado anterior.

Se puede extender la noción de concatenación entre un patrón y un grafo generado por la concatenación de patrones. El procedimiento es completamente análogo al anterior. De esta forma, se puede definir la concatenación de múltiples patrones $G_1, \ldots, G_m$ como:

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

Definición 1.18. Dado un polinomio $P = \{M_1, \ldots, M_{|P|}\}$ y un patrón para cada monomio G_i, se define el grafo de linealización de P como la concatenación de los patrones de cada monomio: $G = \mathcal{C}(G_1, \ldots, G_{|P|})$.

En la figura 1.2 se muestra un grafo de linealización asociado a los monomios del problema del ejemplo 1.4, utilizando los patrones 1.1a, 1.1b y 1.1d. Nótese que, como el subgrafo generado a partir del nodo $\{3,4\}$ es el mismo en 1.1b y en 1.1d, se reutiliza el mismo nodo.

Una vez se han establecido las definiciones de grafo de linealización y patrón de linealización se pueden utilizar para definir reformulaciones asociadas a problemas (BPO).

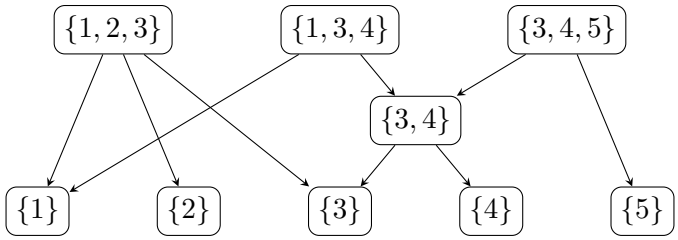


Figura 1.2: Grafo de linealización asociado al problema del ejemplo 1.4, usando los patrones 1.1a, 1.1b y 1.1d.

Definición 1.19. Dado un polinomio P y un grafo de linealización asociado $G = (V, A)$, se define el problema de programación entera mixta asociado al grafo G como:

$$(\text{MILP}_\text{G}) \begin{cases} \underset{\boldsymbol{Y}}{\text{minimizar}} & \displaystyle\sum_{M \in P} c_M Y_M & \\ \text{sujeto a} & Y_v \leq Y_s & \forall\, v \in V \ \forall\, s \in \delta_G^+(v) \\ & Y_v \geq 1 + \displaystyle\sum_{s \in \delta_G^+(v)} (Y_s - 1) & \forall\, v \in V \\ & Y_v \geq 0 & \forall\, v \in V, v \text{ no terminal} \\ & Y_v \in \{0, 1\} & \forall\, v \in V, v \text{ terminal.} \end{cases}$$

Se puede ver que este problema tiene una estructura similar al problema (SL). En particular, la linealización estándar se puede expresar como el problema asociado a un cierto grafo de linealización, siendo este grafo el que tiene únicamente los nodos asociados a los monomios y los nodos terminales. Por ejemplo, el grafo de linealización que genera la linealización estándar mostrada en el ejemplo 1.7 sería el que se muestra en la figura 1.3.

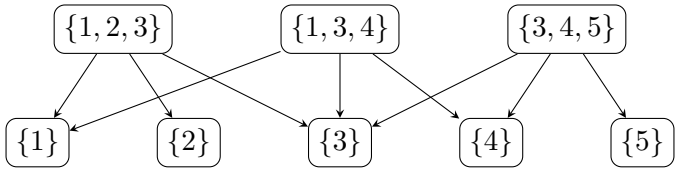


Figura 1.3: Grafo de linealización asociado al problema del ejemplo 1.7.

Ejemplo 1.20. En este ejemplo se muestra el problema de programación entera mixta asociado al grafo de la figura 1.2. Se puede ver que todos los términos son lineales y las únicas variables que son binarias son las asociadas a los nodos terminales, es decir, a los monomios con una sola

variable.

$$\begin{cases} \underset{\boldsymbol{Y}}{\text{minimizar}} & Y_{123} + Y_{134} + 3Y_{345} \\ \text{sujeto a} & Y_{123} \leq Y_1, \;\; Y_{123} \leq Y_2, \;\; Y_{123} \leq Y_3 \\ & Y_{134} \leq Y_1, \;\; Y_{134} \leq Y_{34} \\ & Y_{345} \leq Y_{34}, \;\; Y_{345} \leq Y_5 \\ & Y_{123} \geq Y_1 + Y_2 + Y_3 - 2 \\ & Y_{134} \geq Y_1 + Y_{34} - 1 \\ & Y_{345} \geq Y_{34} + Y_5 - 1 \\ & Y_{123} \geq 0, \;\; Y_{134} \geq 0, \;\; Y_{345} \geq 0, \;\; Y_{34} \geq 0 \\ & Y_1, Y_2, Y_3, Y_4, Y_5 \in \{0, 1\}. \end{cases}$$

Proposición 1.21. *Dado un polinomio P y un grafo de linealización asociado G, los problemas (BPO) y (MILP$_G$) son equivalentes.*

Demostración. La demostración es análoga a la demostración de la equivalencia entre (BPO) y (SL) en 1.9. Se puede aplicar el mismo razonamiento que en el lema 1.8 para comprobar que, bajo la hipótesis de que las variables son binarias en ambos problemas:

$$Y_M = \prod_{i \in M} y_i, \; \forall M \in P \iff \text{Se cumplen las restricciones de (MILP}_\text{G}\text{)}.$$

De esto se deduce la equivalencia al igual que se hizo en la demostración anterior. □

1.3. Reformulación más ajustada: `MaxBound`

Una vez se ha visto que se pueden definir múltiples grafos de linealización asociados a un polinomio, se pueden buscar aquellos con ciertas propiedades deseables. Uno de los objetivos que se puede buscar es obtener el grafo en el cual el óptimo de la relajación continua del problema asociado sea lo más cercano posible al óptimo del problema original. Para resolver este problema se idea `MaxBound`.

`MaxBound` es un problema de optimización que encuentra el grafo de linealización que resulta en la relajación continua más ajustada. Para describir este grafo en un problema de optimización se pueden numerar todos los patrones de linealización asociados a un monomio M dado, asignándole una variable a cada uno de ellos.

$$u_M^\alpha = \begin{cases} 1 & \text{si se escoge el patrón } \alpha \text{ para el monomio } M \\ 0 & \text{en otro caso.} \end{cases}$$

El número de patrones asociado a un monomio se calcula en [1], pero para los propósitos de este trabajo, podemos denotarlo simplemente por N_M.

Por otro lado, para representar las variables y los patrones a los que están sujetas se pueden definir las variables Y_M^α. Estas representan el monomio $\prod_{i\in M} y_i$ sujeto a las restricciones correspondientes al patrón α.

Utilizando estas variables, la linealización de un monomio M se puede representar como $\sum_{\alpha=1}^{N_M} u_M^\alpha \cdot Y_M^\alpha$, siempre que se cumpla que $\sum_{\alpha=1}^{N_M} u_M^\alpha = 1$, es decir, que solo se utiliza un patrón para cada monomio. Utilizando estas variables se pueden buscar los valores que maximicen la solución de la relajación continua del problema original entre los distintos grafos de linealización posibles:

$$
(MB)\begin{cases}
\underset{\boldsymbol{u}}{\text{maximizar}} & \begin{cases}
\underset{\boldsymbol{Y}}{\text{minimizar}} & \displaystyle\sum_{M\in P} c_M\Big(\sum_{\alpha=1}^{N_M} u_M^\alpha Y_M^\alpha\Big) \\
\text{sujeto a} & Y_v^\alpha \text{ se linealiza siguiendo el} \\
 & \text{patrón } \alpha\ \forall\, v\subseteq M\in P,\ \forall\alpha\in\{1,\dots,N_s\} \\
 & 0\le Y_v^\alpha\le 1 \quad \forall\, v\subseteq M\in P,\ \forall\alpha\in\{1,\dots,N_s\}
\end{cases} \\
\text{sujeto a} & \displaystyle\sum_{\alpha=1}^{N_M} u_M^\alpha = 1 \quad \forall M\in P \\
 & u_M^\alpha\in\{0,1\} \quad \forall M\in P,\ \forall\alpha\in\{1,\dots,N_s\}.
\end{cases}
$$

Para obtener un problema de maximización que se pueda resolver con los métodos habituales, se puede aplicar el dual para transformar el problema de minimización interno en uno de maximización, y así obtener un problema de maximización en dos grupos de variables.

`MaxBound` es una herramienta interesante, ya que ofrece una forma teórica de obtener la reformulación más ajustada entre aquellas asociadas a patrones de linealización simples. Sin embargo, tiene problemas de escalado importantes. Se debe añadir una variable binaria por cada patrón asociado a cada monomio. Este número crece exponencialmente con el grado del monomio. Para tratar de afrontar este problema, se puede intentar reducir el número de patrones que se tienen en consideración para cada monomio, tratando de garantizar que entre los que se consideran se encuentre aquel que esté asociado a la relajación continua más ajustada. Para ello, se plantea siguiente teorema.

Teorema 1.22 (Dominación entre patrones de linealización simples)**.** *Sea P un polinomio y $G=(V,A)$ y $\widetilde{G}=(\widetilde{V},\widetilde{A})$ dos grafos de linealización asociados, en los cuales todos los patrones son simples. Se asume que ambos grafos son iguales, exceptuando los patrones asociados a un*

cierto monomio $M \in P$ en cada grafo, para el cual el patrón $\widetilde{G}_M$ tiene exactamente un nodo más que G_M. Entonces, el valor óptimo de la relajación continua de (MILP$_G$) es menor o igual que el valor óptimo de la relajación continua de (MILP$_{\widetilde{G}}$).

Demostración. Se denota por (RLP$_G$) a la relajación continua de (MILP$_G$) y por (RLP$_{\widetilde{G}}$) a la relajación continua de (MILP$_{\widetilde{G}}$). Se toma $\boldsymbol{Y} = (Y_v)_{v\in\widetilde{V}}$ una solución de (RLP$_{\widetilde{G}}$). Se tiene que $\boldsymbol{Y}$ satisface las siguientes restricciones:

$$Y_v \leq Y_s \qquad \forall\, v \in \widetilde{V},\ \forall\, s \in \delta^+_{\widetilde{G}}(v) \tag{1.4}$$

$$Y_v \geq 1 + \sum_{s\in\delta^+_{\widetilde{G}}(v)} (Y_s - 1) \qquad \forall\, v \in \widetilde{V},\ v \text{ no terminal} \tag{1.5}$$

$$0 \leq Y_v \leq 1 \qquad \forall\, v \in \widetilde{V}. \tag{1.6}$$

Se toma ahora la restricción de $\boldsymbol{Y}$ a $(Y_v)_{v\in V}$, excluyendo el nodo adicional presente en $\widetilde{V}$. Se denota este nodo adicional como z. Se denota como $\hat{v}$ al predecesor de z en $\widetilde{G}_M$.

Por hipótesis G_M y $\widetilde{G}_M$ son iguales exceptuando el nodo adicional z. En particular, se tiene que $\widetilde{V}_M = V_M \cup \{z\}$. Por tanto necesariamente $|z| \geq 2$, de lo contrario z sería un nodo terminal que está en $\widetilde{G}_M$ pero no en G_M, luego G_M y $\widetilde{G}_M$ serían patrones de monomios distintos. Ahora, se busca probar que Y restringida a V es factible para (RLP$_G$). Para verlo, se pueden desglosar las restricciones (1.4), (1.5) y (1.6), específicamente los casos relativos al monomio M y que involucren a los nodos z y $\hat{v}$. Empezando por (1.4):

$$Y_v \leq Y_s \qquad \forall\, v \in V_M \setminus \{\hat{v}\},\ \forall\, s \in \delta^+_{\widetilde{G}}(v) \tag{1.7}$$

$$Y_{\hat{v}} \leq Y_s \qquad \forall\, s \in \delta^+_{\widetilde{G}}(\hat{v}) \setminus \{z\} \tag{1.8}$$

$$Y_{\hat{v}} \leq Y_z \tag{1.9}$$

$$Y_z \leq Y_s \qquad \forall\, s \in \delta^+_{\widetilde{G}}(z). \tag{1.10}$$

A continuación, para la restricción (1.5):

$$Y_v \geq 1 + \sum_{s\in\delta^+_{\widetilde{G}}(v)} (Y_s - 1) \qquad \forall\, v \in V_M \setminus \{\hat{v}\},\ v \text{ no terminal} \tag{1.11}$$

$$Y_{\hat{v}} \geq 1 + \sum_{s\in\delta^+_{\widetilde{G}}(\hat{v})\setminus\{z\}} (Y_s - 1) + (Y_z - 1) \tag{1.12}$$

$$Y_z \geq 1 + \sum_{s\in\delta^+_{\widetilde{G}}(z)} (Y_s - 1). \tag{1.13}$$

Por último, para la restricción (1.6):

$$0 \leq Y_v \leq 1 \qquad \forall\, v \in V_M \tag{1.14}$$

$$0 \leq Y_z \leq 1. \tag{1.15}$$

Es necesario demostrar las siguientes desigualdades:

$$Y_v \leq Y_s \qquad \forall\, v \in V_M,\ \forall\, s \in \delta_G^+(v) \tag{1.16}$$

$$Y_v \geq 1 + \sum_{s \in \delta_G^+(v)} (Y_s - 1) \qquad \forall\, v \in V_M,\ v \text{ no terminal} \tag{1.17}$$

$$0 \leq Y_v \leq 1 \qquad \forall\, v \in V_M. \tag{1.18}$$

Empezando por (1.16), con la restricción (1.7) se tiene la restricción para todos los nodos de $V_M \setminus \{\hat{v}\}$, ya que en estos nodos se cumple que $\delta_{\widetilde{G}}^+(v) = \delta_G^+(v)$. A continuación, combinando (1.9) y (1.10) se obtiene que $Y_{\hat{v}} \leq Y_s$ para todo $s \in \delta_{\widetilde{G}}^+(z)$. Esto, junto a (1.8) implica que $Y_{\hat{v}} \leq Y_s$ para todo $s \in \delta_{\widetilde{G}}^+(z) \cup (\delta_{\widetilde{G}}^+(\hat{v}) \setminus \{z\})$. Ahora, por la propia definición de $\hat{v}$ y z como nodo adicional de $\widetilde{G}$, se ve que $\delta_{\widetilde{G}}^+(z) \cup (\delta_{\widetilde{G}}^+(\hat{v}) \setminus \{z\}) = \delta_G^+(\hat{v})$. Por tanto, se tiene que $Y_{\hat{v}} \leq Y_s$ para todo $s \in \delta_G^+(\hat{v})$, es decir, se cumple (1.16).

Para (1.17), con la restricción (1.11) se tiene el resultado para todo $v \in V_M \setminus \{\hat{v}\}$, v no terminal. Para ver que se cumple en $\hat{v}$ se pueden combinar (1.12) y (1.13):

$$Y_{\hat{v}} \geq 1 + \sum_{s \in \delta_G^+(\hat{v}) \setminus \{z\}} (Y_s - 1) + (Y_z - 1) \geq 1 + \sum_{s \in \delta_G^+(\hat{v}) \setminus \{z\}} (Y_s - 1) + \sum_{s \in \delta_G^+(z)} (Y_s - 1) = 1 + \sum_{s \in \delta_G^+(\hat{v})} (Y_s - 1).$$

Donde en la última igualdad se tuvo en cuenta que, al igual que antes, $\delta_{\widetilde{G}}^+(z) \cup (\delta_{\widetilde{G}}^+(\hat{v}) \setminus \{z\}) = \delta_G^+(\hat{v})$ y además los dos conjuntos de la unión son disjuntos, por tanto no hay términos que se repitan en las dos sumas. Por tanto, se cumple (1.17).

Por último (1.18) se cumple trivialmente por (1.14) y (1.15).

Con esto, se demuestra que la restricción de $\boldsymbol{Y}$ a $(Y_v)_{v \in V}$ es factible en (RLP_G). Las funciones objetivo de ambos problemas solo dependen de los valores de $\boldsymbol{Y}$ asociados a los nodos terminales, que son los mismos en V y $\widetilde{V}$. Por tanto, el valor de la función objetivo en $\boldsymbol{Y}$ es el mismo para (RLP_G) y $(\mathrm{RLP}_{\widetilde{G}})$. Así, por ser ambos problemas de minimización, y poder construirse una solución de (RLP_G) para toda solución de $(\mathrm{RLP}_{\widetilde{G}})$, se tiene que el valor óptimo de (RLP_G) es menor o igual que el valor óptimo de $(\mathrm{RLP}_{\widetilde{G}})$. □

De forma general, el teorema indica que al añadir más nodos al patrón se obtendrán soluciones al menos tan ajustadas como las anteriores. El problema (SL) se puede obtener a partir de un cierto grafo de linealización, en particular el que tiene menos nodos de todos. Por el resultado

del teorema anterior, se podrá esperar que todos los grafos de linealización tengan una relajación continua al menos tan ajustada como la de (SL). En el siguiente corolario se formaliza este resultado.

Corolario 1.23 (Dominación de patrones de linealización simples sobre la linealización estándar)**.** *Sea P un polinomio, G la linealización estándar de P y $\widetilde{G}$ un grafo de linealización asociado a P en el cual todos los patrones son simples. Entonces, el valor óptimo de la relajación continua de ($MILP_G$) es menor o igual que el valor óptimo de la relajación continua de ($MILP_{\widetilde{G}}$).*

Demostración. La demostración consiste en construir una colección de grafos de linealización, iniciada en G y terminando en $\widetilde{G}$, de forma que en cada grafo haya un nodo más que en el anterior. De esta forma se podrá generar una cadena de desigualdades en los valores óptimos de las relajaciones continuas de cada problema. Se definen los grafos G^k de la siguiente forma.

- $G^1 = G$.
- $G^K = \widetilde{G}$.
- Para todo $k \in \{1, \ldots, K-1\}$, G^{k+1} es igual a G^k exceptuando uno de los monomios, en el cual el patrón asociado tiene un nodo adicional.

Se puede aplicar el teorema 1.22 a estos grafos para así obtener la cadena de desigualdades,

$$v(\mathrm{RLP_G}) = v(\mathrm{RLP}_{G^1}) \leq v(\mathrm{RLP}_{G^2}) \leq \cdots \leq v(\mathrm{RLP}_{G^K}) = v(\mathrm{RLP}_{\widetilde{G}}),$$

donde v denota el valor óptimo del problema. □

De este corolario se deduce que no sería necesario tener en cuenta los grafos que den como resultado la linealización estándar. Con el siguiente corolario se expande esta idea, se enumeran los patrones que se deben considerar para obtener la reformulación más ajustada, siendo innecesario tener en cuenta otros.

Corolario 1.24 (Reducción del conjunto de patrones a considerar en `MaxBound`)**.** *Sea P un polinomio. Para todo $M \in P$ se pueden considerar únicamente los patrones simples de forma que el grado de salida de todo nodo no terminal sea 2 (es decir, árboles binarios), sin afectar al resultado obtenido por* ***`MaxBound`****.*

Demostración. Sea un polinomio P y un grafo de linealización asociado G en el cual todos los patrones son simples. Sea $M \in P$ un monomio tal que el patrón asociado tiene un nodo no terminal con grado de salida mayor que dos. Se pueden añadir nodos al patrón de forma que los nuevos nodos tengan grado de salida 2 y se reduzca el grado de salida del nodo original. Por

el teorema 1.22, cada vez que se añade un nodo se obtiene una relajación continua igual o más ajustada que la original. Por tanto se ve que es suficiente considerar únicamente patrones simples con grado de salida 2. □

Los patrones simples que se pueden omitir gracias al corolario 1.24 se denominan patrones *dominados*. Por el contrario, aquellos que deben tenerse en cuenta se denominan *no dominados*. De los patrones mostrados en la figura 1.1, se puede ver que los patrones 1.1b y 1.1d son no dominados, sin embargo el patrón 1.1a es dominado, ya que el nodo $\{1, 2, 3\}$ tiene grado de salida 3. El patrón 1.1c no se considera porque no es simple. En [1] se calcula el número total de patrones simples y el número de patrones no dominados. Para monomios de grado 3, la reducción es del 25 % y para grado 5 se consigue un 55 %.

1.4. Reformulación con el mínimo número de variables: ND-MinVar

El método MaxBound encuentra la reformulación más ajustada entre todas las formadas por patrones de linealización no dominados, sin tener en cuenta el número de variables que se añaden al problema original. Aunque como consecuencia del teorema 1.22 se pueda reducir el número de patrones considerados, sigue siendo difícil aplicar el algoritmo a polinomios de grados altos. Se puede plantear un método que propone un compromiso alternativo. En lugar de tratar de obtener la reformulación más ajustada, se puede intentar obtener aquella que añada el mínimo número de variables al problema original, dentro de las formadas por patrones no dominados. Este método se denomina ND-MinVar, del inglés *non-dominated, minimum variables*.

En ND-MinVar se plantea un problema de optimización parecido al utilizado en MaxBound, con el cual se busca minimizar el número de variables añadidas al problema original en la reformulación. En este problema se utilizan las variables u_M^α definidas para MaxBound y se definen las variables w_v^β para todo $v \subseteq M$ de la forma:

$$w_v^\beta = \begin{cases} 1 & \text{el patrón } \beta \text{ de } v \text{ forma parte del patrón escogido} \\ & \text{para alguno de los monomios de } P \\ 0 & \text{en otro caso.} \end{cases}$$

Esta definición implica que $w_v^\beta = 1$ si y solo si la variable Y_v^β linealizada siguiendo las restricciones asociadas al patrón β aparece en la reformulación final del problema. Se busca minimizar este número de variables, por tanto se plantea un problema minimizando la suma de

los w_v^β:

$$(MV)\begin{cases} \underset{\boldsymbol{u},\boldsymbol{w}}{\text{minimizar}} \displaystyle\sum_{v\subseteq M, M\in P}\sum_{\beta} w_v^\beta & \\ \text{sujeto a } u_M^\alpha \leq w_v^\beta & \forall M\in P,\ \forall \alpha \\ & \forall\, v \text{ tal que el patrón } \beta \text{ de } v \text{ aparece} \\ & \text{en el patrón } \alpha \text{ de } M \\ \displaystyle\sum_{\alpha=1}^{N_M} u_M^\alpha = 1 & \forall M\in P \\ u_M^\alpha \in \{0,1\} & \forall M\in P,\ \forall \alpha \\ w_v^\beta \in \{0,1\} & \forall\, v\subseteq M,\ \forall \beta. \end{cases} \tag{1.19}$$

Con la restricción (1.19) se asegura que si se escoge el patrón α para M, es decir, $u_M^\alpha = 1$, entonces para todos los nodos v cuyo patrón β aparezca en el α de M se cumple $w_v^\beta = 1$. De esta forma, se garantiza que si se escoge un patrón para un monomio, se añaden al problema todas las variables asociadas a los nodos que aparecen en dicho patrón.

No es necesario hacer que las variables w_v^β sean binarias. Siempre tomarán el mínimo valor posible, y al estar acotadas por las variables u_M^α, que son binarias, se podrá garantizar que las w_v^β tomarán únicamente los valores 0 o 1.

La solución del problema (MV) proporciona unos valores de u que determinan los patrones a escoger para cada monomio, y consecuentemente la reformulación escogida para el problema.

En este capítulo se han planteado diversos conceptos. Se han definido los problemas de optimización polinómica con variables binarias y se han establecido distintas reformulaciones de equivalentes de estos problemas, basándose principalmente en los patrones y grafos de linealización. Además, se han visto dos posibles formas de obtener reformulaciones de problemas con propiedades deseables, como maximizar la cota inferior de la relajación continua del problema (`MaxBound`) o minimizar el número de variables añadidas (`ND-MinVar`).

Capítulo 2

Reformulación de problemas de optimización polinómica generales

En el capítulo anterior se plantearon los problemas de optimización polinómica con variables binarias y se propusieron distintas técnicas que se podían utilizar para facilitar su resolución. Esta clase de problemas se puede utilizar para modelizar ciertas situaciones específicas, como por ejemplo clasificadores binarios. Sin embargo, para la mayor parte de problemas reales es necesaria la inclusión de variables no binarias (enteras generales o continuas). Además, parte del objetivo de este trabajo es la inclusión de estas técnicas en `RAPOSa`, que es un optimizador de problemas de programación polinómica mixta.[1] Para esta inclusión, es necesario extender los resultados más allá de los problemas con variables binarias. Por estos motivos, sería beneficioso poder expandir los resultados vistos hasta ahora a un contexto más general.

En este capítulo se plantearán nuevas definiciones, resultados y teoremas que extenderán a los que se han expuesto en el capítulo 1. En primer lugar, se ampliará la noción de problema de optimización polinómica, incluyendo variables no binarias. A continuación, se generalizarán los conceptos de patrón de linealización y grafo de linealización a este tipo de problemas. Posteriormente, se demostrará que en esta generalización los problemas reformulados también son equivalentes a los problemas originales. Por último, se ejemplificará la reformulación de un problema de optimización polinómica general, aplicando la linealización estándar.

[1] En el capítulo 3 se habla en mayor detalle sobre el funcionamiento de `RAPOSa`.

2.1. Problemas de optimización polinómica generales

En esta sección se definen los problemas de optimización polinómica generales, como una ampliación de los problemas de optimización polinómica con variables binarias vistos hasta ahora. Para esta definición es necesario plantear algunas nociones previas, como por ejemplo la de *multiconjunto*. Estas definiciones se basan en parte en las que se pueden encontrar en [2].

Definición 2.1. Dado un conjunto S y una aplicación $p : S \to \mathbb{N} \cup \{0\}$, se define un *multiconjunto* como el par (S, p), donde p es la aplicación que asigna a cada elemento su multiplicidad. Habitualmente se comete un abuso de notación escribiendo un multiconjunto como un conjunto en el que aparecen los elementos repetidos, indicando su multiplicidad. Por ejemplo, el multiconjunto $(\{0, 1\}, p)$ con p tal que $p(0) = 1$ y $p(1) = 2$ se denota como $\{0, 1, 1\}$.

Los multiconjuntos son necesarios porque al utilizar variables no binarias se puede dar que una misma variable aparezca varias veces en un monomio (es decir, que tenga un exponente mayor que 1). Esto no sucedía antes, ya que para las variables binarias $y_i^2 = y_i$. Utilizando multiconjuntos se puede definir una nueva representación de un monomio.

Definición 2.2. Dado un conjunto NB de índices de variables no binarias, un conjunto B de índices de variables binarias y los vectores $\boldsymbol{x} = (x_j)_{j \in NB}$, $\boldsymbol{y} = (y_i)_{i \in B}$; se define un *monomio con variables no binarias y binarias* como el producto $\prod_{j \in M_{NB}} x_j \prod_{i \in M_B} y_i$, donde M_{NB} es un multiconjunto basado en el conjunto NB y M_B es un subconjunto de B. Como pasaba en el capítulo anterior, en este trabajo la parte relevante del monomio es el par de conjuntos $M = (M_{NB}, M_B)$. Por tanto, cuando se use el término monomio se estará hablando de su representación con dicho par.

Ejemplo 2.3. Utilizando la notación de las definiciones 2.1 y 2.2, con las variables $x_1, x_2 \in \mathbb{R}$ e $y_3, y_4 \in \{0, 1\}$ el monomio $x_1 x_2^2 y_3 y_4$ se representaría como el par

$$(\{1, 2, 2\}, \{3, 4\}).$$

De la nueva representación de monomios se puede extender una nueva representación de polinomios.

Definición 2.4. Dado un conjunto NB de índices de variables no binarias, un conjunto B de índices de variables binarias y los vectores $\boldsymbol{x} = (x_j)_{j \in NB}$, $\boldsymbol{y} = (y_i)_{i \in B}$; se define un *polinomio con variables no binarias y binarias* como la suma $\sum_{M \in P} c_M \prod_{j \in M_{NB}} x_j \prod_{i \in M_B} y_i$. En la suma cada M es un par representando un monomio tal y como se define en 2.2 y cada $c_M \in \mathbb{R}$ representa el término que acompaña al monomio M. De nuevo en este trabajo la parte de interés del polinomio es el conjunto P, por tanto cuando se hable de polinomio se hablará específicamente de su representación utilizando el conjunto P.

Una vez se tienen las nuevas representaciones de monomios y polinomios, se pueden definir los problemas de optimización polinómica generales. Estos suponen una expansión de los problemas vistos hasta ahora, incluyendo variables no binarias en los monomios y una serie de restricciones adicionales.

Definición 2.5. Dado un conjunto NB de índices de variables no binarias, un conjunto $Z \subset NB$ de índices de variables enteras y un conjunto B de índices de variables binarias, se define un *problema de optimización polinómica general* como

$$
\text{(PG)} \quad \begin{cases} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimizar}} & \displaystyle\sum_{M\in P_0} c_M \Big(\prod_{j\in M_{NB}} x_j\Big)\Big(\prod_{i\in M_B} y_i\Big) & \\ \text{sujeto a} & \displaystyle\sum_{M\in P_r} a_{r,M} \Big(\prod_{j\in M_{NB}} x_j\Big)\Big(\prod_{i\in M_B} y_i\Big) \geq b_r & \forall\, r \in R_1 \\ & \displaystyle\sum_{M\in P_r} a_{r,M} \Big(\prod_{j\in M_{NB}} x_j\Big)\Big(\prod_{i\in M_B} y_i\Big) = b_r & \forall\, r \in R_2 \\ & \boldsymbol{x} \in \Omega_{NB} & \\ & \boldsymbol{y} \in \Omega_B & \end{cases}
$$

donde:

- P_0 representa el polinomio de la función objetivo.
- Los polinomios P_r aparecen en las restricciones de desigualdad $R_1 = \{1, \ldots, r_1\}$ y en las restricciones de igualdad $R_2 = \{r_1 + 1, \ldots, r_1 + r_2\}$.
- Para cada $M \in P_0$, c_M representa el coeficiente real que acompaña al monomio.
- Para cada $M \in P_r$, $a_{r,M}$ representa el coeficiente real que acompaña al monomio M en la restricción r.
- El conjunto Ω representa la región factible del problema, siendo:
  1. $\Omega = \Omega_{NB} \times \Omega_B$.
  2. $\Omega_{NB} = \{\boldsymbol{x} = (x_j)_{j\in NB} \in \mathbb{R}^{|NB|} : 0 \leq l_j \leq x_j \leq u_j < \infty,\ \forall\, j \in NB,$
     $x_j \in \mathbb{Z},\ \forall\, j \in Z\}$.
  3. $\Omega_B = \{\boldsymbol{y} = (y_i)_{i\in B} \in \mathbb{R}^{|B|} : 0 \leq y_i \leq 1,\ y_i \in \mathbb{Z},\ \forall\, i \in B\}$.

Pese a la posible complejidad de la notación, esta definición es solo una expansión de la definición 1.3. A cada monomio se le añade el término $\prod_{j\in M_{NB}} x_j$ representando las variables no binarias. Por otro lado, se añaden restricciones de igualdad y desigualdad, con sus correspondientes polinomios. Por último se añaden cotas para las variables no binarias con Ω, y se restringen las variables enteras.

Tomando $NB = R_1 = R_2 = \emptyset$ se pueden formular problemas de tipo (BPO) de la misma forma que se hizo en el capítulo anterior.

2.2. Reformulación de problemas de optimización polinómica generales

En esta sección se adaptarán algunas de las definiciones vistas en la sección 1.2 al contexto de problemas de optimización polinómica generales. Esta adaptación consistirá en tomar la parte binaria de los problemas y utilizar las definiciones ya vistas aplicadas a esa parte.

Definición 2.6. Dado un monomio de variables no binarias y binarias $M = (M_{NB}, M_B)$ se define un patrón de linealización de M como un patrón de linealización del monomio de variables binarias M_B, definido en 1.14. Es decir, un patrón de linealización asociado a la parte binaria del monomio.

Ejemplo 2.7. Dadas las variables y_1, y_2, $y_3 \in \{0,1\}$ y x_4, $x_5 \in \mathbb{R}$ y el monomio $y_1y_2y_3x_4^2x_5^3$ se tiene que al patrón de la figura 1.1a sería un patrón de linealización válido para el monomio. Lo es ya que es un patrón de linealización asociado a la parte binaria del monomio, que es $\{1,2,3\}$.

Definición 2.8. Se define un grafo de linealización asociado a un polinomio P como un grafo de linealización asociado al polinomio formado únicamente por los monomios de variables binarias, representado por $P_B = \{M_B : (M_{NB}, M_B) \in P\}$.

Como en los problemas (PG) hay más de un polinomio, algo que no sucedía en los problemas anteriores, es necesario definir un grafo de linealización asociado al problema en sí, que abarque a todos los monomios de todos sus polinomios.

Definición 2.9. Dado un problema de programación polinómica general se define un grafo de linealización asociado al problema como la concatenación de grafos de linealización asociados al polinomio de la función objetivo y los de las restricciones.

Ejemplo 2.10. En este ejemplo se muestra un problema de programación polinómica general y un grafo de linealización asociado. Se parte del problema:

$$\begin{cases} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimizar}} & y_1y_2y_3x_6^2 - 2y_3y_4y_5x_7 + x_6x_7 \\ \text{sujeto a} & y_1x_6 \geq 3 \\ & y_2x_7 - y_1y_3x_6 \geq 5 \\ & y_1, y_2, y_3 \in \{0,1\} \\ & x_6, x_7 \geq 0. \end{cases}$$

Se puede construir un grafo de linealización asociado, que se muestra en la figura 2.1. En particular, este grafo es la concatenación de patrones de linealización asociados a los monomios $y_1y_2y_3$, $y_3y_4y_5$ e y_1y_3.

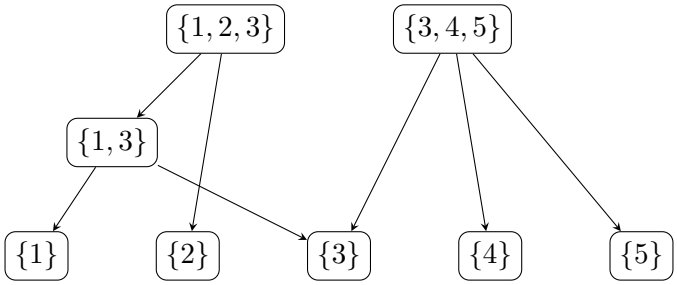


Figura 2.1: Un grafo de linealización asociado al problema anterior.

Una vez se ha trasladado la definición de grafo de linealización, se puede definir la reformulación asociada a un cierto grafo.

Definición 2.11. Dado un problema de programación y un grafo de linealización asociado al problema $G = (V, A)$, se define la reformulación asociada a G del problema como:

$$
(RPG_{\mathrm{G}}) \quad \begin{cases}
\underset{\boldsymbol{x},\boldsymbol{Y}}{\text{minimizar}} & \displaystyle\sum_{M\in P_0} c_M\Big(\prod_{j\in M_{NB}} x_j\Big)Y_{M_B} & & \\
\text{sujeto a} & \displaystyle\sum_{M\in P_r} a_{r,M}\Big(\prod_{j\in M_{NB}} x_j\Big)Y_{M_B} \geq b_r & \forall\, r\in R_1 & (2.1)\\
& \displaystyle\sum_{M\in P_r} a_{r,M}\Big(\prod_{j\in M_{NB}} x_j\Big)Y_{M_B} = b_r & \forall\, r\in R_2 & (2.2)\\
& Y_v \leq Y_s & \forall\, v\in V\ \forall\, s\in \delta_G^+(v) & \\
& Y_v \geq 1 + \displaystyle\sum_{s\in\delta_G^+(v)} (Y_s - 1) & \forall\, v\in V & (2.3)\\
& Y_v \geq 0 & \forall\, v\in V, v \text{ no terminal} & (2.4)\\
& Y_v \in \{0,1\} & \forall\, v\in V, v \text{ terminal.} & (2.5)\\
& \boldsymbol{x}\in\Omega_{NB}. &
\end{cases}
$$

Para que esta reformulación sea útil es necesario demostrar que a partir de su resolución se puede obtener una solución del problema original.

Teorema 2.12. *Dado un problema (PG) y un grafo de linealización asociado G, el problema (RPG_G) es equivalente.*

Demostración. Esta demostración es análoga a la demostración de la proposición 1.9. Se puede aplicar el mismo razonamiento que se utiliza en el lema 1.8 para demostrar que al ser las variables

y_i binarias, se tiene que el hecho de que se cumplan las restricciones (2.3), (2.4) y (2.5) es equivalente a que $Y_M = \prod_{i \in M} y_i$ para todo monomio binario presente en el problema. Para esta parte de la demostración las nuevas variables no binarias no influyen.

A continuación, se verá que dada una solución de cualquiera de los dos problemas se puede obtener una solución del otro cuyo valor de la función objetivo sea el mismo. Partiendo de una solución del problema (PG) $(\boldsymbol{x}, \boldsymbol{y})$, se construyen los términos Y_M del problema (RPG_G), tomando cada $Y_M = \prod_{i \in M} y_i$. Como se ha visto, esto garantiza el cumplimiento de las restricciones (2.3), (2.4) y (2.5). Ahora, para las restricciones (2.1) y (2.2) se aplica el razonamiento

$$\sum_{M \in P_r} a_{r,M} \Big(\prod_{j \in M_{NB}} x_j \Big) Y_{M_B} = \sum_{M \in P_r} a_{r,M} \Big(\prod_{j \in M_{NB}} x_j \Big) \Big(\prod_{i \in M_B} y_i \Big) \geq b_r,$$

donde la última desigualdad se cumple por ser $(\boldsymbol{x}, \boldsymbol{y})$ solución del problema original. Se puede hacer lo mismo para las restricciones de igualdad. Por último, teniendo en cuenta de nuevo la igualdad $Y_M = \prod_{i \in M} y_i$ se tiene que el valor de la función objetivo es el mismo en ambos problemas. Así, se ha demostrado que dada una solución de (PG) se puede obtener una solución de (RPG_G) con el mismo valor de la función objetivo.

La demostración de que se puede obtener una solución al problema (PG) a partir de una solución de (RPG_G) es análoga. Con ella se demuestra que ambos problemas son equivalentes. □

En el capítulo anterior, se enuncian varios teoremas que aportan garantías teóricas sobre las posibles reformulaciones de un problema. En particular, se estudia la cota inferior (en problemas de minimización) que se puede obtener con la relajación continua de las reformulaciones. Estos resultados no son directamente trasladables al contexto de problemas de clase (PG), pero desde el grupo de investigación se está trabajando para obtener sus demostraciones.

2.3. Un caso específico: la linealización estándar

En el capítulo anterior, se vio que la linealización estándar aplicada a problemas (BPO) era simplemente un caso particular de un problema asociado a un cierto grafo de linealización G. En esta sección, con el objetivo de ejemplificar las reformulaciones en problemas de programación polinómica generales, se planteará un problema, su grafo de linealización asociado a la linealización estándar y, finalmente, su reformulación.

Se parte del siguiente problema:

$$\begin{cases} \underset{x,y}{\text{minimizar}} & x_1x_2y_4 - 2x_2x_3 + x_3y_4y_5 - 3y_4y_5y_6 + y_7 \\ \text{sujeto a} & x_2 - x_1y_4y_6 \geq 2 \\ & x_3 + y_5y_6 = 3 \\ & x_3y_7 - x_2y_6 \geq 1 \\ & x_1, x_2, x_3 \geq 0 \\ & y_4, y_5, y_6, y_7 \in \{0, 1\}. \end{cases}$$

En este problema se pueden encontrar los monomios binarios $y_4y_5y_6$, y_4y_5, y_4y_6, y_5y_6, y_4, y_6 e y_7. Para construir la linealización estándar es necesario formar su grafo asociado. Como se vio en el capítulo anterior, este grafo es aquel que contiene únicamente los nodos asociados a los monomios y los nodos terminales. En este caso, para los monomios binarios presentes en el problema, se tendría el grafo representado en la figura 2.2.

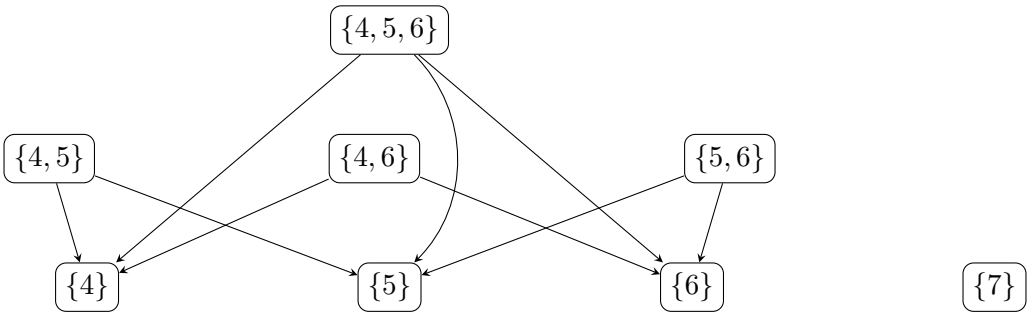


Figura 2.2: Grafo de linealización de la linealización estándar del problema de ejemplo.

Siguiendo la definición 2.11, se pueden sustituir los monomios binarios por variables en la función objetivo y en las restricciones, añadiendo las nuevas restricciones asociadas a las nuevas variables. De esta forma se obtiene la linealización estándar del problema.

$$
\begin{cases}
\underset{x,Y}{\text{minimizar}} & x_1x_2Y_4 - 2x_2x_3 + x_3Y_{45} - 3Y_{456} + Y_7 \\
\text{sujeto a} & x_2 - x_1Y_{46} \geq 2 \\
& x_3 + Y_{56} = 3 \\
& x_3Y_7 - x_2Y_6 \geq 1 \\
& Y_{456} \leq Y_4, \;\; Y_{456} \leq Y_5, \;\; Y_{456} \leq Y_6 \\
& Y_{45} \leq Y_4, \;\; Y_{45} \leq Y_5 \\
& Y_{46} \leq Y_4, \;\; Y_{46} \leq Y_6 \\
& Y_{56} \leq Y_5, \;\; Y_{56} \leq Y_6 \\
& Y_{456} \geq Y_4 + Y_5 + Y_6 - 2 \\
& Y_{45} \geq Y_4 + Y_5 - 1 \\
& Y_{46} \geq Y_4 + Y_6 - 1 \\
& Y_{56} \geq Y_5 + Y_6 - 1 \\
& x_1, x_2, x_3 \geq 0 \\
& Y_{456}, \; Y_{45}, \; Y_{46}, \; Y_{56} \geq 0 \\
& Y_4, Y_5, Y_6, Y_7 \in \{0, 1\}.
\end{cases}
$$

Utilizando una herramienta de resolución por ordenador para resolver ambos problemas, se llega al mismo resultado, tal y como se esperaba por el teorema 2.12. La función objetivo toma el valor -11 y las variables toman los valores $x_1 = 0$, $x_2 = 2$, $x_3 = 3$, $y_4 = 0$, $y_5 = 0$, $y_6 = 0$ y $y_7 = 1$.

Capítulo 3

Reformulación de problemas de optimización polinómica general en RAPOSa

En este trabajo se han planteado varios resultados teóricos, técnicas y algoritmos que pueden ser útiles para la resolución de un cierto tipo de problemas de optimización. Para poder tener una visión real del funcionamiento de estos mecanismos sería interesante aplicarlos a problemas concretos y estudiar los resultados obtenidos.

De septiembre a diciembre de 2023 realicé unas prácticas en el CITMAga en las cuales colaboré con el equipo de desarrollo de `RAPOSa`, familiarizándome con la herramienta, su ecosistema de desarrollo y su entorno de ejecución y validación. Por último, implementé los mecanismos expuestos en este capítulo como un primer paso de cara a su futura integración en la herramienta.

En este capítulo se explicará el trabajo realizado para implementar los mecanismos de reformulación en el optimizador `RAPOSa`. En primer lugar, se explicará la técnica RLT, que es la base del funcionamiento del optimizador. A continuación se explicará brevemente qué es `RAPOSa` y cómo se han implementado los algoritmos en el optimizador. Por último, se explicarán los experimentos planteados y los resultados numéricos obtenidos.

3.1. La técnica RLT

La técnica RLT (*Reformulation-Linearization-Technique*) es una técnica diseñada para la resolución de problemas de programación polinómicos con restricciones de caja. Esto es, problemas similares a aquellos de la forma (PG), sin variables enteras (y consecuentemente tampoco

binarias). Esto se traduce en un problema (PG) con $B = Z = \emptyset$:

$$\begin{cases} \underset{\boldsymbol{x}}{\text{minimizar}} & \displaystyle\sum_{M\in P_0} c_M\Big(\prod_{j\in M_{NB}} x_j\Big) & \\ \text{sujeto a} & \displaystyle\sum_{M\in P_r} a_{r,M}\Big(\prod_{j\in M_{NB}} x_j\Big) \geq b_r & \forall\, r \in R_1 \\ & \displaystyle\sum_{M\in P_r} a_{r,M}\Big(\prod_{j\in M_{NB}} x_j\Big) = b_r & \forall\, r \in R_2 \\ & \boldsymbol{x} \in \Omega_{NB}. & \end{cases}$$

La técnica se basa en la obtención de relajaciones lineales del problema original que permitan guiar la búsqueda de la solución óptima. Para explicar esta técnica son necesarias algunas definiciones previas.

Definición 3.1. Dado un monomio $M = (M_{NB}, \emptyset)$, se define la identidad RLT de M como la expresión:

$$X_M = \prod_{j\in M_{NB}} x_j,$$

donde las X_M se denominan variables RLT.

Definición 3.2. La linealización de un polinomio P es la expresión que se obtiene al sustituir en la expresión de P cada monomio por su variable RLT correspondiente. En particular, se denota de la forma:

$$\left[\sum_{M\in P} c_M \prod_{j\in M_{NB}} x_j\right]_L = \sum_{M\in P} c_M X_M.$$

Definición 3.3. Dado un conjunto NB de variables no binarias y un grado $\delta \in \mathbb{N}$, se define NB^δ como el multiconjunto (NB, p) donde $p(j) = \delta$ para todo $j \in NB$.

Definición 3.4. Dados dos multiconjuntos $J_1,\ J_2 \subset NB^\delta$ se define la *bound factor constraint* asociada a J_1 y J_2 como la restricción:

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

donde l_j y u_j son las cotas inferiores y superiores de la variable x_j respectivamente, presentes en el conjunto Ω_{NB}.

Se puede ver que desarrollando la expresión $F_\delta(J_1, J_2)$ se obtiene un polinomio de la misma forma que los utilizados hasta ahora. Consecuentemente también se podrá obtener su linealización.

Definición 3.5. Dado un problema de programación polinómica general (PG) sin variables binarias, se define su linealización para RLT como el problema:

$$\begin{cases} \underset{\boldsymbol{x}}{\text{minimizar}} & \left[\sum_{M\in P_0} c_M\Big(\prod_{j\in M_{NB}} x_j\Big)\right]_L & \\ \text{sujeto a} & \left[\sum_{M\in P_r} a_{r,M}\Big(\prod_{j\in M_{NB}} x_j\Big)\right]_L \geq b_r & \forall\, r\in R_1 \\ & \left[\sum_{M\in P_r} a_{r,M}\Big(\prod_{j\in M_{NB}} x_j\Big)\right]_L = b_r & \forall\, r\in R_2 \\ & \left[\prod_{j\in J_1}(x_j-l_j)\prod_{j\in J_2}(u_j-x_j)\right]_L \geq 0 & \forall\, J_1\cup J_2\subset NB^\delta,\ |J_1\cup J_2|=\delta \\ & \boldsymbol{x}\in\Omega_{NB}. & \end{cases}$$

Las *bound factor constraints* se añaden para garantizar que RLT converge a la solución óptima global y hacen que la relajación lineal sea más ajustada, algo que ya se vio que es beneficioso en el capítulo 1. Su número crece exponencialmente respecto a δ y al número de variables no binarias, por tanto estas restricciones pueden hacer el problema intratable muy rápidamente. Por ello, se desarrolló el concepto de *J-sets*, que permite utilizar un subconjunto más reducido de estas restricciones preservando la convergencia global.

La idea de RLT es utilizar estas relajaciones lineales para guiar el algoritmo de ramificación y poda. Con la solución del problema linealizado se puede asignar un peso para cada variable, teniendo en cuenta en qué grado el valor de la variable viola las identidades RLT en las que aparece. La variable con mayor peso, es decir, aquella que vulnera en mayor grado las identidades RLT, se selecciona como variable de ramificación. Por otro lado, el valor óptimo de las relajaciones lineales se utiliza para la parte de poda, evitando la resolución de relajaciones lineales que no aportarían información nueva.

Al principio de la sección se especificó que RLT fue planteado originalmente para problemas sin variables enteras. Sin embargo, en [2] se demuestra que el algoritmo basado en RLT se puede extender a problemas de programación polinómica con variables enteras, utilizando relajaciones lineales y enteras. Se preserva la convergencia del algoritmo a un óptimo global.

3.2. Un optimizador basado en la técnica RLT: RAPOSa

RAPOSa [3] (*Reformulation Algorithm for Polynomial Optimization–Santiago*) es un software de optimización global especialmente diseñado para la resolución de problemas de programación

lineal y entera. El equipo de desarrollo de RAPOSa está formado principalmente por investigadores de la Universidad de Santiago de Compostela, en particular asociados al CITMAga. De septiembre a diciembre de 2023, tuve la oportunidad de integrarme en este grupo y colaborar en el desarrollo del optimizador, aplicando algunos de los resultados teóricos que se describen en este trabajo.

El funcionamiento de RAPOSa se basa en la técnica RLT expuesta en la sección anterior. El optimizador genera las relajaciones lineales, las resuelve utilizando un optimizador auxiliar y utiliza la solución para guiar el proceso de ramificación y poda. Para mejorar su rendimiento, RAPOSa incluye una serie de mejoras sobre el algoritmo original basado en RLT. Estas están relacionadas con el ajuste de las cotas del problema, ajustes en la estrategia de ramificación o el uso de los optimizadores auxiliares. Una posible mejora consiste en la reformulación del problema original antes de iniciar el proceso de resolución, utilizando las técnicas de reformulación descritas en este trabajo. En la siguiente sección se explicará como se ha llevado a cabo la inclusión de estas técnicas.

3.3. Reformulación en RAPOSa

RAPOSa es un optimizador capaz de resolver problemas de optimización polinómica incluyendo variables enteras (y en particular, binarias). Por ello, incluir en el optimizador las técnicas descritas en este trabajo no ampliaría su capacidad de resolución de problemas. Lo que se pretende con la implementación de estas técnicas es una posible mejora en el rendimiento al resolver ciertos tipos de problemas, como por ejemplo problemas con polinomios de grado alto y con variables binarias.

Para incluir las reformulaciones en RAPOSa no es necesario interactuar directamente con el algoritmo. Como son equivalentes al problema original, se pueden incluir como una fase de preprocesamiento del problema. De esta forma, el algoritmo de RAPOSa resolverá el problema reformulado, pero de la solución obtenida se podrá extraer la solución del problema original.

Antes de iniciar el desarrollo, se evaluó qué técnicas sería posible implementar en el optimizador. Se buscaba encontrar un grafo de linealización apropiado y aplicarlo al problema para obtener la reformulación. Se consideró que utilizar los algoritmos de MaxBound y ND-MinVar no sería viable, por la complejidad de su implementación. Pese a ser estos los resultados principales de [1], en el artículo también se menciona que una posible alternativa a la utilización de esos algoritmos es el uso de heurísticas. Estas no garantizan tener el mínimo número de variables adicionales o la máxima cota inferior, pero su implementación es más simple y en la práctica pueden dar buenos resultados. Se pensaron tres mecanismos para encontrar un grafo de linealización:

- **Linealización estándar**: se puede generar el grafo asociado a la linealización estándar, vista en la sección 2.3, que es simple y fácil de implementar. Puede funcionar como línea base para comparar con los otros mecanismos.
- **Patrón no dominado canónico**: se puede implementar un algoritmo para generar un patrón no dominado para cada uno de los monomios. Estos se pueden concatenar para obtener un grafo de linealización.
- **Patrón no dominado obtenido mediante heurística**: se puede diseñar una heurística que tenga en cuenta todos los monomios del problema, en lugar de ir uno por uno, para intentar obtener patrones que reutilicen variables. Esto es una alternativa a la utilización de `ND-MinVar`.

Una vez se selecciona un grafo, su aplicación consiste en sustituir los monomios binarios y añadir las restricciones necesarias tal y como se describe en la definición 2.11. Después de obtener el problema reformulado, se pasa a la fase de resolución del problema. En las siguientes secciones se detallará la implementación de cada uno de los mecanismos y se mostrarán los resultados obtenidos al aplicarlos a problemas concretos.

3.3.1. Linealización estándar

La implementación de la linealización estándar fue la más simple de las tres, por la propia simplicidad del patrón de linealización asociado. Un detalle a tener en cuenta tanto en la linealización estándar como en los siguientes mecanismos es que, en la implementación, los procesos de generar el grafo de linealización y de aplicarlo al problema no son independientes. En el código lo más natural es realizar ambos pasos a la vez, a medida que se genera el grafo se sustituyen los monomios. En líneas generales, el algoritmo consiste en los siguientes pasos:

1. Iterar por todos los monomios del problema.
2. Para cada monomio M seleccionar sus variables binarias M_B.
3. Si M_B no se ha encontrado en ninguno de los monomios hasta ahora, sustituirlo por una variable nueva. Si ya se ha encontrado, sustituirlo por la variable utilizada previamente.
4. Finalmente, después de iterar por todos los monomios, añadir las restricciones asociadas a las nuevas variables.

Esta reformulación es simple, pero, como ya se ha visto antes, no proporciona los mejores resultados en cuanto al ajuste de las cotas inferiores de las linealizaciones.

3.3.2. Patrón no dominado canónico

Como se definió a partir del corolario 1.24, los patrones no dominados son los patrones de linealización simples en los cuales el grado de salida de todo nodo no terminal es 2. Se pueden pensar diversas formas de obtener un patrón no dominado para un cierto monomio. Una de las más sencillas es la siguiente:

1. Generar un nodo raíz con todas las variables binarias del monomio. Sustituir la parte binaria del monomio por una nueva variable.

2. Ordenar en orden ascendente las variables en una colección y dividir esta colección a la mitad (pudiendo tener la primera mitad un elemento más si el número de variables del monomio es impar).

3. Añadir al nodo de partida dos hijos, uno con las variables de la primera mitad y otro con las variables de la segunda mitad.

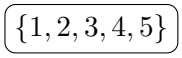
4. Repetir el proceso para cada uno de los nodos hijos, hasta llegar a nodos con un solo elemento.

Utilizando este proceso se pueden obtener un patrón no dominado para cada monomio del problema. En la figura 3.1 se muestra el proceso de generación de un patrón para un monomio con 5 variables binarias. El algoritmo para generar un grafo de linealización consiste en repetir este proceso para cada uno de los monomios del problema y concatenar los patrones obtenidos en un único grafo.

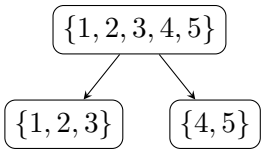
Un posible problema que presenta este algoritmo es que no tiene una visión global de todos los monomios del problema. Se generan los patrones independientemente para cada monomio, por tanto no se tienen en cuenta posibles oportunidades de reutilizar variables entre monomios. Por ejemplo, continuando con el ejemplo iniciado en la figura 3.1, en el caso en el cual el siguiente monomio encontrado contenga las variables binarias $\{1, 2, 3, 4\}$. En la figura 3.2 se muestra el proceso de generación del patrón para dicho monomio.

La variable asociada al nodo $\{1, 2\}$ se podrá reutilizar, pero para el nodo $\{3, 4\}$ será necesario añadir una nueva. Si se hubiera utilizado la subdivisión $\{1, 2, 3\}$ y $\{4\}$, se podría haber reutilizado la variable asociada a $\{1, 2, 3\}$, que ya está presente en el anterior monomio.

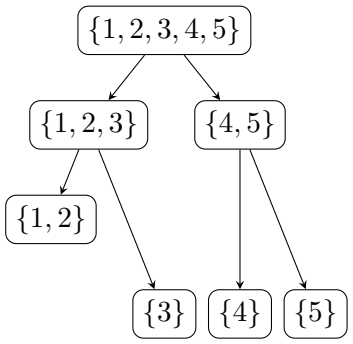
Para tratar de mitigar el problema de la reutilización de variables, se implementa la generación de patrones no dominados mediante heurística.



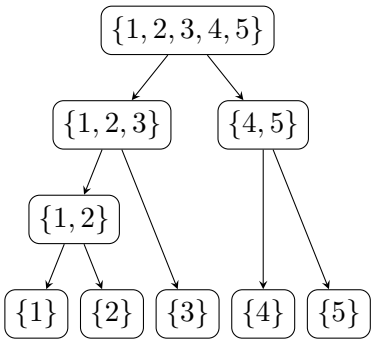
(a) Nodo raíz con todas las variables binarias.



(b) División del nodo raíz en dos nodos.

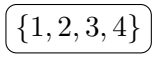


(c) División de los nodos anteriores.

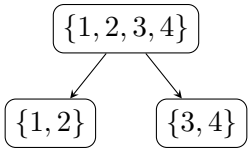


(d) Patrón de linealización final.

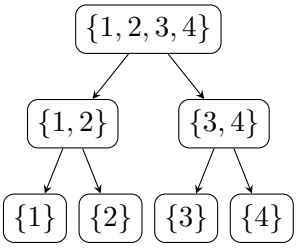
Figura 3.1: Ejemplo de patrón no dominado canónico para un monomio con 5 variables binarias.



(a) Nodo raíz con todas las variables binarias.



(b) División del nodo raíz en dos nodos.



(c) División de los nodos anteriores.

Figura 3.2: Ejemplo de patrón no dominado canónico para un monomio con 4 variables binarias.

3.3.3. Patrón no dominado obtenido mediante heurística

La heurística planteada para obtener los patrones no dominados tratando de minimizar el número de variables utilizadas se apoya en dos ideas. En primer lugar, si se sustituyen primero los monomios con un menor número de variables, se aumenta la probabilidad de encontrarse estos mismos monomios contenidos en otros monomios más grandes, pudiendo así reutilizar las variables. En segundo lugar, dado un monomio se pueden analizar todas las posibles formas de separarlo en dos para analizar cuál reutiliza más variables de las sustituidas hasta ahora. Teniendo en cuenta estas dos ideas, se plantea la siguiente heurística:

1. Ordenar todos los monomios del problema según su número de variables binarias, de menor a mayor.

2. Para cada monomio M seleccionar sus variables binarias M_B.
3. Si M_B ya se ha encontrado antes, sustituirlo por la variable utilizada previamente.
4. En otro caso, si M_B solo tiene 2 variables, sustituirlo por una nueva variable.
5. En última instancia, generar todas las posibles formas de dividir M_B en dos. Seleccionar la división que permita reutilizar el mayor número de variables.
6. Finalmente añadir las restricciones asociadas a las nuevas variables.

Ejecutar este algoritmo conlleva un tiempo añadido a la resolución del problema. Sin embargo, se espera que este tiempo adicional de preprocesado se compense con un menor tiempo de resolución.

3.4. Experimentos realizados y resultados

En esta sección se describirán los experimentos realizados para evaluar los mecanismos de reformulación implementados en RAPOSa y se analizarán los resultados obtenidos.

3.4.1. Batería de problemas

Para realizar las pruebas, se utilizaron los mecanismos de reformulación para resolver una batería de problemas. Estos problemas se generaron modificando una batería de 180 problemas de programación polinómica, presentada originalmente en [4]. Los problemas utilizados de base no contienen variables binarias. Para poder poner a prueba las heurísticas desarrolladas, en cada problema, se transformaron aleatoriamente algunas de las variables en binarias. Además, se añadieron términos de holgura para garantizar que el problema fuese factible. Tras esta transformación se obtuvo un conjunto de 180 problemas con amplia variedad en el grado de los polinomios, el número de variables, el número de restricciones y otras características.

3.4.2. Entorno de pruebas

Las ejecuciones de los problemas se realizaron utilizando la herramienta RAPOSa Cloud. Este es un software complementario a RAPOSa que facilita la ejecución de pruebas y obtención de resultados. En particular, utiliza los servidores del CESGA para ejecutar los problemas. Estos servidores tienen una gran capacidad de cómputo, con procesadores *Intel Xeon Platinum 8352Y* de 64 núcleos y 256 GB de memoria RAM. Para cada problema se estableció un tiempo máximo

de ejecución de 1 hora. Los problemas que no se resolvieron en este tiempo se consideraron no resueltos.

3.4.3. Métricas de evaluación

Para evaluar el funcionamiento de los mecanismos de reformulación se utilizaron múltiples métricas. En este trabajo se considerarán tres de ellas:

- **Geometric mean time (GMG)**: media geométrica del tiempo utilizado por el optimizador para resolver cada uno de los problemas. De esta métrica 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 propósito de utilizar la media geométrica en lugar de la media tradicional es reducir el efecto de posibles valores atípicos.
- **Geometric mean gap (GMG)**: media geométrica del los *gaps* obtenidos por el optimizador. Cuando el optimizador no es capaz de resolver un determinado problema de manera exacta, proporciona una cota inferior y una cota superior para su valor óptimo. La diferencia entre estos dos valores se denomina *gap*. Cuanto menor sea, mejor se considera la aproximación. De esta métrica se excluyen los problemas que han sido resueltos por todos los mecanismos y aquellos en los que alguno de los mecanismos no ha conseguido proporcionar un *gap*, ya que en esos casos no se puede comparar directamente los valores de forma significativa.
- **Número total de problemas resueltos**: número de problemas que el optimizador ha sido capaz de resolver utilizando cada uno de los mecanismos. Se considera que un problema ha sido resuelto si el optimizador ha sido capaz de proporcionar una solución exacta en un tiempo menor a 1 hora.

3.4.4. Resultados de las pruebas

En la tabla 3.1 se muestran los resultados obtenidos. Las columnas de la tabla se corresponden respectivamente con la versión base de `RAPOSa` (sin preprocesado del problema), preprocesado utilizando la linealización estándar, preprocesado utilizando el patrón no dominado canónico y preprocesado utilizando el patrón no dominado obtenido mediante heurística. Al la derecha de cada métrica se indica entre paréntesis el número de problemas que han sido considerados, después de descartar los problemas que no aportan información significativa como se especificó en el apartado anterior para cada métrica.

	Base	LE	PNDC	PNDH
GMT (44)	30.82	33.48	34.31	31.60
GMG (8)	0.032	0.041	0.041	0.038
Resueltos (84)	82	82	82	83

Tabla 3.1: Comparación de los resultados obtenidos por los mecanismos implementados.

Los resultados indican que de media la aplicación de los mecanismos de reformulación no supone una mejora en el tiempo de ejecución ni en los *gaps* obtenidos. Además, solo se están teniendo en cuenta 8 problemas de los 180 de la batería en el caso de la métrica de GMG, por tanto los resultados para esa métrica no son del todo representativos. Pese a esto, se observa que el mecanismo de reformulación mediante heurística ha sido capaz de resolver un problema más que el resto de los mecanismos. Esto es un indicio de que puede que los mecanismos funcionen mejor para algunas situaciones, o en problemas determinados.

Existen distintas formas de analizar específicamente en qué caso cada mecanismo ha funcionado mejor. Se podrían comparar los resultados para cada problema en específico, pero esto es algo tedioso y no aporta una visión global. Se puede tomar otra perspectiva, en lugar de observar directamente los resultados obtenidos por cada mecanismo, se puede calcular un cómputo general tomando para cada problema el resultado del mecanismo que obtuvo la solución en menos tiempo o con el mejor *gap*. En la tabla 3.2 se muestran los resultados obtenidos con esta perspectiva (repitiendo la columna relativa a la versión base de RAPOSa para comparar).

	Base	Menor tiempo	Menor *gap*
GMT (44)	30.82	19.41	29.74
GMG (8)	0.032	0.028	0.028
Resueltos (84)	82	84	84

Tabla 3.2: Comparación de resultados agregados.

En este caso, el tiempo obtenido es significativamente menor que el obtenido al utilizar RAPOSa sin modificaciones. Esto significa que existen problemas en los que utilizar los mecanismos de reformulación mejora considerablemente el tiempo de ejecución.

3.5. Trabajo futuro

Aunque las pruebas realizadas indican que los mecanismos de reformulación pueden suponer una cierta mejora, sería necesaria una investigación en profundidad para deducir en qué casos

se mejora y cuál es el grado de mejora. En primer lugar, sería necesario ampliar o modificar la batería de problemas utilizada. Ahora mismo se están descartando muchos de los problemas, por resolverse demasiado rápido o por no obtenerse solución. Esto lleva a que las métricas no sean del todo representativas. En segundo lugar, habría que analizar en mayor profundidad todas las métricas disponibles en `RAPOSa`. En este trabajo solo se han expuesto tres, ya que son las más fáciles de entender y reflejan el comportamiento del optimizador, pero teniendo en cuenta las otras métricas se obtendría una mejor intuición de cuando son mejores unos mecanismos de reformulación u otros. Por último, se podría estudiar el grado de efectividad de la heurística. Su objetivo es reutilizar el máximo número de variables posibles, por tanto se debería comparar el número de variables utilizadas en cada uno de los mecanismos para ver si la heurística cumple su objetivo.

Capítulo 4

Conclusiones

En este trabajo se han presentado una serie de resultados sobre reformulaciones de problemas de optimización polinómica con variables binarias. Se han extendido estos resultados al caso más general de problemas de optimización polinómica con variables no binarias y se han adaptado estas reformulaciones para su inclusión en un software de optimización, mostrando los resultados obtenidos.

En el capítulo 1 se han presentado reformulaciones que permiten transformar problemas de optimización polinómica con variables binarias en problemas de optimización lineal y entera. Estas reformulaciones permiten utilizar técnicas conocidas para la resolución de este último tipo de problemas. Además se han mostrado algoritmos para escoger una reformulación entre todas las posibles, siguiendo criterios como escoger la reformulación que genere el problema más ajustado entre todos (`MaxBound`) o la que añada el menor número de variables (`ND-MinVar`).

En el capítulo 2 se han ampliado algunos de los resultados presentados en el capítulo 1 para problemas de optimización polinómica con variables no binarias. Este es un trabajo original que permite utilizar las reformulaciones del capítulo anterior en contextos más generales.

En el capítulo 3 se han puesto en práctica las reformulaciones presentadas en el capítulo anterior, adaptándolas para su inclusión en un software de optimización. Se han presentado resultados obtenidos al utilizarlas para resolver una batería de problemas, concluyendo que existen problemas en los cuales el tiempo de resolución se reduce al usar las reformulaciones.

En resumen, en este trabajo se realiza una contribución al estado del conocimiento en el contexto de la resolución de problemas de optimización polinómica con variables binarias, aportando resultados originales que permiten el uso de estas técnicas para la resolución en un contexto de problemas algo más general.

Bibliografía

[1] Sourour Elloumi y Zoé Verchère. «Efficient linear reformulations for binary polynomial optimization problems». En: *Computers & Operations Research* 155 (2023), pág. 106240.

[2] Brais González Rodríguez. «Advances in polynomial optimization». Tesis doct. Universidade de Santiago de Compostela, 2022.

[3] Brais González-Rodríguez et al. «Computational advances in polynomial optimization: RAPOSa, a freely available global solver». En: *Journal of Global Optimization* 85.3 (2023), págs. 541-568.

[4] Evrim Dalkiran y Hanif D Sherali. «RLT-POS: Reformulation-Linearization Technique-based optimization software for solving polynomial programming problems». En: *Mathematical Programming Computation* 8.3 (2016), págs. 337-375.