Traballo Fin de Mestrado

Técnicas de ramificación e acoutamento no ámbito da optimización polinómica

Manuel Álvarez Rodríguez

Mestrado en Técnicas Estatísticas

Curso 2023-2024

Proposta de Traballo Fin de Mestrado

Título en galego: Técnicas de ramificación e acoutamento no ámbito da optimización polinómica
Título en español: Técnicas de ramificación y acotación en el ámbito de la optimización polinómica
English title: Branch and Bound Techniques in the Field of Polynomial Optimization
Modalidade: Modalidade B
Autor: Manuel Álvarez Rodríguez, Universidade de Santiago de Compostela
Director: Julio González Díaz, Universidade de Santiago de Compostela; Brais González Rodríguez, Universidade de Santiago de Compostela
Titor: Gabriel Álvarez Castro, CITMAga
Breve resumen do traballo: Este traballo girará en torno a una implementación específica de técnicas de ramificación espacial, en el ámbito de la optimización polinómica. El optimizador en cuestión, RAPOSa (González-Rodríguez y otros, 2023), ha sido desarrollado por investigadores de CITMAga basándose en la técnica RLT introducida por Sherali y Tuncbilek (1992). Tras una fase inicial de familiarización con RAPOSa y con la técnica RLT, el estudiante colaborará en el estudio y validación de posibles variaciones en el procedimiento de ramificación y acotación, con el objetivo de mejorar su eficiencia computacional.
Recomendacións:
Outras observacións:

Don Julio González Díaz, Titular de Universidade da Universidade de Santiago de Compostela, don Brais González Rodríguez, informan que o Traballo Fin de Mestrado titulado

Técnicas de ramificación e acoutamento no ámbito da optimización polinómica

realizouse baixo a súa dirección por don Manuel Álvarez Rodríguez para o Mestrado en Técnicas Estatísticas. Estimando que o traballo está terminado, dan a súa conformidade para a súa presentación e defensa ante un tribunal.

En Santiago de Compostela, a 3 de xuño de 2024.

Firmado por GONZALEZ DIAZ JULIO - ***3267** el día 03/06/2024 con un certificado emitido por AC FNMT Usuarios

GONZALEZ RODRIGUEZ BRAIS - 44661382M

Digitally signed by GONZALEZ RODRIGUEZ BRAIS - 44661382M
Date: 2024.06.03 15:25:12 +02'00'

O director:
Don Julio González Díaz

O director:
Don Brais González Rodríguez

Firmado por ALVAREZ RODRIGUEZ MANUEL - ***8315** el día 03/06/2024 con un certificado emitido por AC FNMT Usuarios

O autor:
Don Manuel Álvarez Rodríguez

Declaración responsable. Para dar cumprimento á Lei 3/2022, do 24 de febreiro, de convivencia universitaria, referente ó plaxio no Traballo Fin de Mestrado (Artigo 11, Disposición 2978 del BOE núm. 48 de 2022), **o autor declara** que o Traballo Fin de Mestrado presentado é un documento orixinal no que se tivo en conta as seguintes consideracións relativas ó uso de material de apoio desenvolto por outros/as autores/as:

- Todas as fontes usadas para a elaboración deste traballo citáronse convenientemente (libros, artigos, apuntes de profesorado, páxinas web, programas,...)
- Calquera contido copiado ou traducido textualmente púxose entre comiñas, citando a súa procedencia.
- Fíxose constar explicitamente cando un capítulo, sección, demostración,... sexa unha adaptación case literal dalgunha fonte existente.

E, acepta que, se se demostrara o contrario, se lle apliquen as medidas disciplinarias que correspondan.

Índice

Resumen

Resumen en galego

A Optimización ou Investigación Operativa, é unha destacada área das matemáticas coñecida pola súa aplicación no mundo real, especialmente na toma de decisións empresariais. Para elo, é necesario crear un modelo matemático que describa, mediante a función obxectivo, aquelo que desexamos maximizar ou minimizar, engadindo as restricións necesarias. En moitas situacións, as funcións empregadas na formulación do problema son polinomios, dando lugar a unha importante clase de problemas, os de optimización polinómica. En xeral, estes problemas adoitan ser non lineais e non convexos, polo que obter unha solución global pode ser unha tarefa complexa. Existen diversos métodos teóricos entre os que destaca o algoritmo baseado na Técnica de Reformulación-Linearización (RLT). `RAPOSa` é un recente *solver* global especialmente deseñado para problemas de optimización polinómica baseado no algoritmo RLT. Neste traballo, presentamos e estudamos o rendemento dunha nova variación no algoritmo baseado na RLT, desenvolta no marco dunhas prácticas no CITMAga, da man do equipo de desenvolvemento de `RAPOSa`.

English abstract

Optimization or Operations Research is a prominent area of mathematics known for its application in the real world, especially in business decision-making. To achieve this, it is necessary to create a mathematical model that describes, through the objective function, what we wish to maximize or minimize, adding the necessary constraints. In many situations, the function used in the formulation of the problem are polynomials, giving rise to an important class of problems: the polynomial optimization ones. In general, these problems tend to be nonlinear and non-convex, so obtaining a global solution can be a complex task. There are various theoretical methods among which the algorithm based on the Reformulation-Linearization Technique (RLT) stands out. `RAPOSa` is a recent global solver specially designed for polynomial optimization problems based on the RLT algorithm. In this work, we present and study the efficiency of a new variation of the RLT-based algorithm, developed during an internship at the CITMAga, with the support of the `RAPOSa` development team.

Prefacio

A Optimización ou Investigación Operativa é a área das matemáticas encargada de modelar situacións reais mediante problemas de maximizar (beneficios) ou minimizar (perdas) para a toma de decisións. Aínda que na antigüidade xa se falaba de minimizar ou maximizar, non é ata mediados do século pasado, co estalido da II Guerra Mundial, cando a Investigación Operativa toma especial relevancia no mundo científico. Podemos destacar ó matemático George B. Dantzig, quen deseñou o principal método de resolucións de problemas de optimización linear, o método símplex. Dende entón, e cos vertixinosos avances científicos, as técnicas de optimización son empregadas diariamente por empresas, industrias e mesmo institucións públicas.

Actualmente, malia os gran avances tecnolóxicos das últimas décadas, aínda existen algúns problemas de optimización dos cales obter unha solución óptima global é unha tarefa moi complexa. Un caso particular son os problemas polinómicos que, en xeral, son non lineais e non convexos. Ademais, esta clase de problemas teñen especial relevancia pola súa aplicación no mundo real, dende a optimización de procesos na industria química (Karia et al., 2022; Teles et al., 2013; Dua, 2015), ata a toma de decisións no sector financeiro (González-Díaz et al., 2021).

Para resolver un problema polinómico podemos atopar na literatura existente diferentes métodos teóricos, entre os que destacamos o algoritmo baseado na Técnica de Reformulación-Linearización (RLT), presentada por Sherali and Tuncbilek (1992). Partindo dun problema polinómico, onde as funcións que describen a rexión factible e a función obxectivo son polinomios, a RLT constrúe unha relaxación linear do problema, substituíndo as compoñentes non lineais das funcións por novas variables e engadindo unha serie de restricións. Este novo problema, aínda que non é equivalente ó orixinal, é facilmente resoluble con técnicas habituais, coma o método símplex, e proporciona unha cota inferior do valor óptimo do problema polinómico. O algoritmo baseado na RLT parte da versión relaxada do problema (construído coa RLT) e ramifica sucesivamente a rexión factible. Deste xeito, obtéñense cotas inferiores e superiores do valor óptimo, reducindo o *gap* de optimalidade ata acadar unha solución óptima.

Recentemente, González-Rodríguez et al. (2023) desenvolveron un novo *solver* de optimización global especialmente deseñado para problemas polinómicos. O seu deseño está baseado no algoritmo RLT, ao que engaden unha serie de melloras.

No Capítulo 1 introducimos os conceptos básicos da Optimización, presentamos as principais etiquetas coas que pode clasificarse un problema de optimizacións, centrándonos especialmente nos problemas polinómicos. No Capitulo 2 detallamos a Técnica de Reformulación-Linearización, introducimos o algoritmo baseado na RLT e probamos a súa

converxencia. No Capítulo 3 explicamos as melloras do algoritmo que se incorporan en `RAPOSa`, como os *J-sets* ou os cortes SDP. Por último, no Capítulo 4 propoñemos unha variación sobre o algoritmo baseado na RLT. No marco dunhas prácticas no CITMAGa, da man do equipo de desenvolvemento de `RAPOSa`, implementamos e estudamos computacionalmente esta variación co *solver* `RAPOSa`, co fin de mellorar a eficiencia do algoritmo.

Capítulo 1

Introdución á Optimización Polinómica

Comezamos recordando brevemente os conceptos básicos da Optimización, así como as principais clases de problemas de optimización e os métodos de resolución dalgunha delas. Empregamos como referencias para esta primeira sección González-Rodríguez (2017), Salazar-González (2001) e Rockafellar (2007). Posteriormente introducimos os problemas de optimización polinómicos, que se describen con detalle na primeira das referencias anteriores seguindo a notación introducida por Sherali and Tuncbilek (1992).

1.1. Marco teórico

1.1.1. Conceptos previos

De forma xeral, defínese un **problema de optimización** como un par (S, f), onde $S \subset \mathbb{R}^n$ é o **conxunto factible** e $f : \mathbb{R}^n \to \mathbb{R}$ é a **función obxectivo**, e consiste en atopar un punto factible $\boldsymbol{x} \in S$ tal que

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \text{ para todo } \boldsymbol{y} \in S. \tag{1.1}$$

Observación 1.1. No ámbito da Investigación Operativa empréganse indistintamente os termos optimización e programación, aínda que nós utilizaremos soamente o primeiro deles. Ademais, nótese que estamos a considerar o problema de minimizar, pois no caso de maximizar bastaría con transformar o problema como segue:

$$\text{máx}\{f(\boldsymbol{x}) \ : \ \boldsymbol{x} \in S\} = -\text{mín}\{-f(\boldsymbol{x}) \ : x \in S\}.$$

Notación 1.2. Denotaremos o vector n-dimensional columna como $\boldsymbol{x} = (x_1, \dots, x_n)^T$.

Definición 1.3. Todo punto $\boldsymbol{x} \in S$ é unha **solución factible** do problema de optimización (S, f). Se $\boldsymbol{x}$ non é unha solución factible dise que é **infactible**.

Definición 1.4. Unha **solución óptima global**, ou **óptimo global**, do problema (S, f) é aquela solución factible $\boldsymbol{x}$ que satisfai a condición (1.1).

Observación 1.5. É posible que un problema non teña unha solución óptima global e, no caso de existir, esta pode non ser única.

Definición 1.6. Defínese o **conxunto óptimo** como aquel que contén tódalas solucións óptimas globais.

Definición 1.7. Dise que un punto $\boldsymbol{x} \in S$ é unha **solución óptima local** se existe un entorno $\mathcal{U}_{\boldsymbol{x}} \subset \mathbb{R}^n$ tal que $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ para todo $\boldsymbol{y} \in S \cap \mathcal{U}_{\boldsymbol{x}}$.

Definición 1.8. Defínese o **valor óptimo** como $\inf_{\boldsymbol{y}\in S}\{f(\boldsymbol{y})\}$. Este valor pode non alcanzarse no conxunto factible.

Definición 1.9. Cando non existe ningunha solución factible, isto é, cando $S = \emptyset$ dise que (S, f) é un **problema infactible**. En tal caso, establécese por convenio que o valor óptimo de (S, f) é $+\infty$.

Definición 1.10. Se existen solucións factibles que permiten descender tanto como desexemos á función obxectivo, ou o que é o mesmo, se para todo $M \in \mathbb{R}$ existe $\boldsymbol{x} \in S$ tal que $f(\boldsymbol{x}) < M$ decimos que (S, f) é un **problema non acoutado** e o seu valor óptimo é $-\infty$.

Notación 1.11. Na literatura existente adoita atoparse un problema de optimización coa formulación estándar seguinte:

$$\begin{array}{ll} \text{minimizar} & f(\boldsymbol{x}) \\ \text{suxeito a} & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \dots, m \\ & h_j(\boldsymbol{x}) = 0, \quad j = 1, \dots, l \\ & \boldsymbol{x} \in X \subset \mathbb{R}^n, \end{array} \tag{1.2}$$

onde X se describe habitualmente mediante restricións sinxelas, como poden ser cotas superiores ou inferiores das variables. Seguindo esta formulación, o conxunto de solucións factibles S contén os puntos de X que satisfán as restricións de desigualdade, $g_i(\boldsymbol{x}) \leq 0$ e as de igualdade $h_j(\boldsymbol{x}) = 0$.

Definición 1.12. Un conxunto $\Omega \subset \mathbb{R}^n$ é unha **caixa** (*box*, en inglés) se é o produto $I_1 \times \cdots \times I_n$ de n intervalos pechados da forma $I_k = [l_k, u_k]$, onde $l_k, u_k \in \mathbb{R}$. Isto é, $\boldsymbol{x} \in \Omega$ se e só se $x_k \in [l_k, u_k]$ para $k = 1, \dots, n$. As variables que están acoutadas deste xeito denomínanse, en inglés, ***box-constrained variables***.

En ocasións, por cuestións metodolóxicas, é conveniente referirnos a unha formulación específica. Non obstante, as restricións dun problema de optimización teñen máis dunha expresión posible, podendo reformularse segundo conveña como segue:

- Calquera restrición $f_i(\boldsymbol{x}) = c_i$, $f_i(\boldsymbol{x}) \leq c_i$ ou $f_i(\boldsymbol{x}) \geq c_i$ pode transformarse noutra da forma $g_i(\boldsymbol{x}) = 0$, $g_i(\boldsymbol{x}) \leq 0$ ou $g_i(\boldsymbol{x}) \geq 0$, respectivamente, sen máis que considerar a función $g_i(\boldsymbol{x}) = f_i(\boldsymbol{x}) - c_i$.

- Escribir a restrición de igualdade $h_i(\boldsymbol{x}) = c_i$ é o mesmo que incluír na formulación as restricións $h_i(\boldsymbol{x}) \leq c_i$ e $h_i(\boldsymbol{x}) \geq c_i$.

- Toda restrición de desigualdade da forma $f_i(\boldsymbol{x}) \geq c_i$ ($f_i(\boldsymbol{x}) \leq c_i$) pode expresarse como $-f_i(\boldsymbol{x}) \leq -c_i$ ($-f_i(\boldsymbol{x}) \geq -c_i$).

- Unha restrición de desigualdade $f_i(\boldsymbol{x}) \leq c_i$ pódese transformar noutra de igualdade, $f_i(\boldsymbol{x}) + s_i = c_i$ engadindo unha variable non negativa s_i, que se coñece como variable de folgura (*slack variable*, en inglés).

- Unha variable x non restrinxida, isto é, que pode tomar valores negativos ou positivos, pode reescribirse como dúas variables non negativas substituíndo x por $x' - x''$, onde $x' \geq 0$ representa a parte positiva e $x'' \geq 0$ a parte negativa.

1.1.2. Distintas clasificacións

Dependendo das propiedades da función obxectivo e do conxunto de solucións factibles ou ben das propias variables dun problema, este pódese etiquetar como linear/non linear, convexo/non convexo, etc. Esta clasificación é moi importante, xa que existen métodos de resolución específicos para cada clase, polo que a dificultade teórica e computacional da resolución dun problema varía moito dependendo da clase na que nos encontremos. Na sección seguinte, veremos con máis detalle a clase de problemas de optimización polinómica, cos que traballaremos nos demais capítulos desta memoria.

Definición 1.13. Dise que unha función $f : \mathbb{R}^n \to \mathbb{R}$ é **afín** se existen $d_0, d_1, \dots, d_n \in \mathbb{R}$ tal que f pode expresarse como

$$f(x_1, \dots, x_n) = d_0 + d_1 x_1 + \cdots + d_n x_n.$$

Se esiximos que $d_0 = 0$, a función f é **linear**, isto é, se

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n} d_i x_i.$$

Observación 1.14. Da definición anterior dedúcese trivialmente que toda función linear é afín.

Definición 1.15. O problema (1.2), é un **problema de optimización linear** (LP) se a función f é linear (ou afín), as funcións g_i e h_j son afíns e o conxunto X é unha caixa. Noutro caso, dise que (1.2) é un **problema de optimización non linear** (NLP).

Para a resolución dun problema linear empréganse técnicas ampliamente documentadas como o método símplex, proposto por Dantzig (1963), o símplex dual ou métodos de punto interior (Dantzig and Thapa, 2003). Introduciremos en máis detalle o símplex dual no Capítulo 4, onde a dualidade sentará as bases teóricas da variación estudada sobre o algoritmo de ramificación e acoutamento espacial. A resolución de problemas de optimización linear, como o incluído no Exemplo 1.16, é sinxela.

Exemplo 1.16. O problema

$$\begin{aligned} \text{minimizar} \quad & x_1 + x_2 - 2x_3 \\ \text{suxeito a} \quad & x_1 + 2 \leq 0 \\ & x_3 - 2x_2 = 0 \\ & x_1, x_2, x_3 \geq 0, \end{aligned}$$

é un exemplo sinxelo de optimización linear (e convexa).

Definición 1.17. Un conxunto $S \subset \mathbb{R}^n$ é **convexo** se para calquera $\boldsymbol{x}, \boldsymbol{y} \in S$ se ten que $\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in S$ para todo $\lambda \in [0, 1]$.

Definición 1.18. Sexa $S \subset \mathbb{R}^n$ un conxunto convexo e non baleiro. Unha función $f : S \to \mathbb{R}$ é **convexa** se

$$f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y}),$$

para todo $\boldsymbol{x}, \boldsymbol{y} \in S$ e $\lambda \in (0, 1)$. Unha función é **cóncava** se $-f$ é convexa.

Definición 1.19. En xeral, un **problema de optimización convexo** (CP) é aquel no que tanto a función obxectivo f como o conxunto factible S son convexos. Cando algunha destas condicións non se cumpra, dirase que é un **problema de optimización non convexa**.

Proposición 1.20. *Se un problema de optimización (1.2) é tal que as funcións f e g_i son convexas, as h_j son afíns e X é un conxunto convexo, entón (1.2) é un problema de optimización convexa.*

Demostración. Posto que f é convexa, basta probar que o conxunto de solucións factibles de (1.2) é convexo. Isto é, dados $\boldsymbol{x}, \boldsymbol{y} \in S$ e $\lambda \in [0,1]$, temos que probar que $\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in S$. En primeiro lugar temos que, pola linealidade das funcións h_j,

$$h_j(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) = \lambda h_j(\boldsymbol{x}) + (1-\lambda)h_j(\boldsymbol{y}) = 0,$$

onde a última igualdade se deduce do feito que $\boldsymbol{x}, \boldsymbol{y} \in S$. Por outra banda, pola convexidade das funcións g_i temos que

$$g_i(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda g_i(\boldsymbol{x}) + (1-\lambda)g_i(\boldsymbol{y}) \leq 0.$$

□

Observación 1.21. Nótese que o recíproco da proposición anterior tamén é certo, pois bastaría considerar $X = S$.

Exemplo 1.22. O problema

$$\begin{array}{ll} \text{minimizar} & x_1 + x_2^2 - 2x_3 \\ \text{suxeito a} & x_1 + 2 \leq 0 \\ & x_1 - 2x_2 = 0 \\ & x_1, x_2, x_3 \geq 0, \end{array}$$

é un exemplo de problema de optimización convexa e non linear.

Proposición 1.23. *Sexa $S \subset \mathbb{R}^n$ un conxunto convexo e non baleiro e $f : S \to \mathbb{R}$ unha función afín (ou linear). Entón f é unha función convexa.*

Demostración. Por hipótese $S \subset \mathbb{R}^n$ é convexo e non baleiro logo, para todo $\boldsymbol{x}, \boldsymbol{y} \in S$ e $\lambda \in (0,1)$, $\boldsymbol{z} = \lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in S$. Ademais, por ser $f : S \to \mathbb{R}$ unha función afín, tense que

$$\begin{aligned} f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) = f(\boldsymbol{z}) &= d_0 + \sum_{i=1}^{n} d_i z_i = d_0 + \sum_{i=1}^{n} d_i \left[\lambda x_i + (1-\lambda)y_i\right] \\ &= d_0 + \lambda \sum_{i=1}^{n} d_i x_i + (1-\lambda) \sum_{i=1}^{n} d_i y_i \\ &= \lambda d_0 + (1-\lambda)d_0 + \lambda \sum_{i=1}^{n} d_i x_i + (1-\lambda) \sum_{i=1}^{n} d_i y_i \\ &= \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}). \end{aligned}$$

Co cal, a Definición 1.18 verificase trivialmente, logo temos probado a convexidade dunha función afín (ou linear). □

De feito, na anterior demostración temos probado que toda función afín é cóncava e convexa. Ademais, en virtude da Proposición 1.23, sabemos que todo problema de optimización linear é a súa vez convexo. É desexable que un problema de optimización sexa convexo xa que toda solución óptima localmente o é globalmente, como probaremos no seguinte resultado, e a súa resolución, en xeral, non é complexa.

Proposición 1.24. *Consideremos un problema de optimización convexa da forma*

$$\begin{aligned} &\text{minimizar} && f(\boldsymbol{x}) \\ &\text{suxeito a} && \boldsymbol{x} \in S. \end{aligned}$$

Tense que todo óptimo local do problema anterior é un óptimo global.

Demostración. Por redución ó absurdo, supoñamos que $\boldsymbol{x}^*$ é un óptimo local que non é global. Isto é, existe un entorno $\mathcal{U}_{\boldsymbol{x}^*} \subset \mathbb{R}^n$ tal que $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ para todo $\boldsymbol{x} \in S \cap \mathcal{U}_{\boldsymbol{x}^*}$, mais $f(\boldsymbol{y}) \leq f(\boldsymbol{x}^*)$ para un certo $\boldsymbol{y} \in S \setminus \mathcal{U}_{\boldsymbol{x}^*}$. Por outra banda, por ser S convexo, tense que

$$\boldsymbol{z}_\lambda = \lambda \boldsymbol{x}^* + (1-\lambda)\boldsymbol{y} \in S, \text{ para todo } \lambda \in [0,1].$$

En particular, para λ suficientemente próximo a 1 temos que $\boldsymbol{z}_\lambda \in S \cap \mathcal{U}_{\boldsymbol{x}^*}$ e

$$f(\boldsymbol{z}_\lambda) = f(\lambda \boldsymbol{x}^* + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}^*) + (1-\lambda) f(\boldsymbol{y}) < \lambda f(\boldsymbol{x}^*) + (1-\lambda) f(\boldsymbol{x}^*) = f(\boldsymbol{x}^*),$$

onde a primeira desigualdade se verifica pola convexidade de f e a segunda se deduce de $f(\boldsymbol{y}) < f(\boldsymbol{x}^*)$. Logo temos unha contradición, xa que $f(\boldsymbol{x}^*) \nleq f(\boldsymbol{z_\lambda})$, $\boldsymbol{z_\lambda} \in S \cap \mathcal{U}_{\boldsymbol{x}^*}$. Chegando polo tanto a que $\boldsymbol{x}^*$ é un óptimo global. □

Definición 1.25. Cando tódalas variables dun problema de optimización da forma (1.2) deben tomar valores enteiros, dise que é un **problema de optimización enteira** (IP). Se hai variables enteiras e non enteiras, dise que é un problema de **optimización enteira mixta** (MIP).

Cando nun problema nos atopamos con variables enteiras, a súa resolución adoita ser máis complexa e require técnicas específicas como a de ramificación e acoutamento. Este algoritmo, que introduciremos no apartado seguinte, serve para entender a ramificación espacial, na que se basea a Técnica de Reformulación-Linearización (RLT).

Os problemas con variables enteiras tamén poden recibir outras etiquetas atendendo as características do problema relaxado, isto é, sen ter en conta o carácter enteiro das variables. Algúns exemplos son os problemas lineais enteiros (ILP) ou os enteiros mixtos non lineais (MINLP), entre outros.

Na seguinte sección, explicaremos en detalle que é un problema de optimización polinómica e porque convén distinguir esta importante clase de problemas.

1.1.3. Técnica de Ramificación e Acoutamento

Como xa comentabamos, a mesma idea na que se basea a técnica de ramificación e acoutamento para resolver problemas lineais enteiros é a que subxace no algoritmo RLT para a resolución dos problemas polinómicos. Polo que recordaremos, de forma moi resumida, en que consiste este algoritmo:

1. **Iniciación:** Partimos dunha versión relaxada do problema, suprimindo as restricións que esixen ás variables tomar valores enteiros. Este novo problema non é equivalente ó orixinal, pero pódese resolver empregando as técnicas habituais de optimización linear. Ademais, se a solución óptima da relaxación linear toma valores enteiros, entón é óptima do problema orixinal. Noutro caso, a relaxación linear proporciona unha cota inferior do valor óptimo e o algoritmo continúa.

2. **Ramificación.** Escollemos a variable cuxo valor está máis afastado de ser enteiro. Dividimos a rexión factible forzando esta variable a tomar o valor enteiro máis próximo por debaixo ou por riba, creando dous novos subproblemas lineais.

3. **Acoutamento.** Resolvemos a relaxación linear para cada subproblema. Se a solución é enteira, obtemos unha solución factible e unha nova cota superior. Se o valor asociado é menor a mellor cota superior, actualizamos a mellor solución e descartamos aqueles espazos da rexión factible onde non se pode mellorar o valor asociado á solución actual. Se non é enteiro, acadamos unha nova cota inferior. Se é maior a actual cota superior deixamos de explorar esta rexión, pois calquera solución nela é peor ca actual. Noutro caso, volvemos a ramificar.

O algoritmo remata cando se pecha o *gap* de optimalidade ou ben cando non quedan por explorar espazos da rexión factible onde se poida mellorar (reducindo) o valor da función obxectivo asociado á mellor solución. Para unha explicación máis en profundidade acerca do algoritmo de ramificación e acoutamento pode verse González-Díaz (2018).

1.2. Optimización polinómica

Na sección anterior vimos exemplos de problemas convexos (lineais e non lineais), que son facilmente resolubles, mais tamén existen, e de feito son moi habituais no mundo real, problemas que non se poden modelizar mantendo estas boas propiedades, como os incluídos nos Exemplos 1.27 e 1.26. O primeiro deles, como veremos, trátase dun problema polinómico mentres que o segundo, a priori, non o é. Incluímos a locución "a priori" xa que, en virtude do teorema clásico de Weierstrass, calquera función continua pode aproximarse por un polinomio. Este feito destaca a importancia da optimización polinómica que, como vemos, xorde dunha clase máis ampla de problemas. Ademais, os problemas polinómicos teñen aplicacións diversas; dende o deseño e optimización de diversos sistemas químicos, posto que moitos procesos da enxeñería química se describen mediante ecuacións polinómicas (Karia et al., 2022; Teles et al., 2013; Dua, 2015), ata a toma de decisións no sector financeiro (González-Díaz et al., 2021).

Exemplo 1.26. O problema

$$\begin{aligned} \text{minimizar} \quad & \operatorname{sen}(x_1) + x_2^2 - 2x_3 \\ \text{suxeito a} \quad & x_1 + 2 \leq 0 \\ & x_1 - 2x_2 = 0 \\ & x_1, x_2, x_3 \geq 0, \end{aligned}$$

é un exemplo de problema de optimización non convexa e non linear.

Exemplo 1.27. O problema

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_1x_2 - x_1x_2^2 \\ \text{suxeito a} \quad & x_1x_2 \geq 3 \\ & x_1^2 - 3x_2 = 1 \\ & 1 \leq x_1, x_2 \leq 10, \end{aligned}$$

é outro exemplo de problema de optimización non convexa e non linear.

En xeral, un problema de optimización polinómica é aquel no que a función obxectivo e as restricións son polinomios e as variables son non negativas e están acoutadas superior e inferiormente. Aínda que esta descrición é clara e comprensible, cómpre incluír unha definición formal. Para elo, comezamos precisando algúns conceptos previos.

Definición 1.28. Sexa $\boldsymbol{x} = (x_1, \dots, x_n)^T$ un vector de variables reais. Dados $\alpha \in \mathbb{R}$ e $\sigma_j \in \mathbb{N} \cup \{0\}$, $j \in \{1, \dots, n\}$, dise que a expresión $m(\boldsymbol{x}) = \alpha \prod_{j=1}^{n} x_j^{\sigma_j}$ é un **monomio** e o seu **grao** é $\sum_{j=1}^{n} \sigma_j$.

Definición 1.29. Dise que un **polinomio** é a suma dun número finito de monomios. Isto é, sexa $\boldsymbol{x} = (x_1, \dots, x_n)^T$ un vector de variables reais, dados $T \subset \mathbb{N}$ o conxunto de índices asociado ós monomios, $\alpha_t \in \mathbb{R}$, para todo $t \in T$, e $\sigma_{tj} \in \mathbb{N} \cup \{0\}$, para todo $t \in T$ e $j \in \{1, \dots, n\}$, entón a expresión

$$\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\sigma_{tj}}$$

é un polinomio. Ademais defínese o seu **grao** como $\max_{t \in T} \left\{ \sum_{j=1}^{n} \sigma_{tj} \right\}$.

Exemplo 1.30. A expresión

$$\phi(x_1, x_2, x_3) = 2 + x_1x_2^2x_3 \tag{1.3}$$

é un polinomio formado pola suma dos monomios 2, de grao 0, e $x_1x_2^2x_3$ de grao 4. Logo o grao do polinomio (1.3) é 4.

Definición 1.31. Dado un conxunto finito N e unha aplicación $p : N \to \mathbb{N} \cup \{0\}$ denominamos **multiconxunto** ó par (N, p), onde a aplicación p indica a multiplicidade de cada elemento, isto é, o número de veces que un elemento é membro do conxunto N. Defínese o **cardinal** dun multiconxunto como $|(N, p)| = \sum_{j \in N} p(j)$.

Notación 1.32. Dado o conxunto $N = \{1, \dots, n\}$, ó que nos referiremos a partir de agora soamente como N, denotamos o multiconxunto (N, p) como

$$\left\{ 1, \overset{p(1)}{\dots}, 1, \dots, n, \overset{p(n)}{\dots}, n \right\}.$$

En particular, cando p é a aplicación constante δ denotamos, facendo un pequeno abuso de notación, o multiconxunto (N, p) por N^δ.

Exemplo 1.33. Sexa $n = 3$ e $\delta = 2$. O multiconxunto N^δ é $\{1, 1, 2, 2, 3, 3\}$ e o seu cardinal é

$$|N^\delta| = \delta |N| = 2 \cdot 3 = 6.$$

Grazas a este novo concepto somos capaces de reformular as expresións alxébricas de monomio e polinomio, coas que traballaremos ó longo da memoria. Sexa $m(\boldsymbol{x}) = \alpha \prod_{j=1}^{n} x_j^{\sigma_j}$ un monomio de grao δ, se consideramos o multiconxunto $J \subset N^{\delta}$ dado por

$$J = \{\underbrace{1, \overset{\sigma_1}{\ldots}, 1}, \ldots, \underbrace{n, \overset{\sigma_n}{\ldots}, n}\},$$

tense que $m(\boldsymbol{x}) = \alpha \prod_{j \in J} x_j$.

Observación 1.34. Nótese que cando a variable x_j non aparece no monomio, é dicir, cando a súa multiplicidade é $\sigma_j = 0$, o elemento j non aparece no multiconxunto J.

De xeito análogo, dado un polinomio $\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\sigma_{tj}}$ de grao δ, podemos expresalo como

$$\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j \in J_t} x_j,$$

sendo J_t o multiconxunto asociado ó monomio t.

Agora xa estamos en condicións de definir rigorosamente os problemas de optimización polinómica.

Definición 1.35. Un **problema de optimización polinómica** (POP ou PP) é un problema de optimización da forma:

$$\begin{array}{lll} \text{minimizar} & \phi_0(\boldsymbol{x}) & \\ \text{suxeito a} & \phi_r(\boldsymbol{x}) \geq \beta_r, & r = 1, \ldots, R_1 \\ & \phi_r(\boldsymbol{x}) = \beta_r, & j = R_1 + 1, \ldots, R \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^n, & \end{array} \qquad (PP(\Omega))$$

onde:

- $\phi_r(\boldsymbol{x}) = \sum_{t \in T_r} \alpha_{rt} \prod_{j \in J_t} x_j$ é un polinomio de grao δ_r, sendo T_r o conxunto de índices, $\alpha_{rt} \in \mathbb{R}$ os coeficientes correspondentes a cada monomio e $J_{rt} \subset N^{\delta_r}$ multiconxuntos tales que $|J_{rt}| \leq \delta_r$, para todo $r \in \{0, \ldots, R\}$.
- O conxunto Ω é unha caixa da forma

  $$\Omega = [l_1, u_1] \times \cdots \times [l_n, u_n] \subset \mathbb{R}^n,$$

  onde os l_j e u_j son cotas inferiores e superiores, respectivamente, das variables x_j tales que

  $$0 \leq l_j \leq x_j \leq u_j < \infty,$$

  para todo $j \in \{1, \ldots, n\}$.

Definición 1.36. Defínese o **grao dun problema de optimización polinómica**, que se denota por δ, como o maior grao dos polinomios que forman o problema, ou o que é o mesmo, o maior grao dos monomios que figuran no problema. Formalmente temos que

$$\delta = \max_{r \in \{0,\ldots,R\}} \delta_r = \max_{r \in \{0,\ldots,R\},\ t \in T} |J_{rt}|.$$

Para facilitar a comprensión desta notación, que empregaremos nos vindeiros capítulos, incluímos o Exemplo 1.27 onde traballamos coas Definicións 1.35 e 1.36.

Exemplo 1.37. Vexamos que o problema

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_1x_2 - x_1x_2^2 \\ \text{suxeito a} \quad & x_1x_2 \geq 3 \\ & x_1^2 - 3x_2 = 1 \\ & 0 \leq x_1, x_2 \leq 10, \end{aligned}$$

é, por definición, un problema de optimización polinómica. En primeiro lugar, o número de variables é 2 logo $N = \{1, 2\}$. Comprobamos que a función obxectivo é da forma

$$\phi_0(\boldsymbol{x}) = 2x_1 + x_1x_2 - x_1x_2^2 = \sum_{t \in T_0} \alpha_{0t} \prod_{j \in J_{0t}} x_j,$$

para uns certos T_0, α_{0t} e J_t. O número de monomios da función obxectivo é 3 logo $T_0 = \{1, 2, 3\}$. O primeiro monomio é $2x_1$ e correspóndese con $J_{01} = 1$ e $\alpha_{01} = 3$. O segundo, x_1x_2, correspóndese con $J_{02} = \{1, 2\}$ e $\alpha_{02} = 1$. Por último, temos que

$$-x_1x_2^2 = -x_1x_2x_2 = \alpha_{02} \prod_{j \in J_{03}} x_j,$$

para $J_{03} = \{1, 2, 2\}$ e $\alpha_{03} = -1$. De xeito análogo, podemos describir as restricións do problema como

$$\phi_1(\boldsymbol{x}) = x_1x_2 = \alpha_{11} \prod_{j \in J_{11}} x_j,$$

para $\alpha = 1$ e $J_{11} = 1, 2$, e

$$\phi_2(\boldsymbol{x}) = x_1^2 - 3x_2 = \sum_{t \in T_2} \alpha_{2t} \prod_{j \in J_{2t}} x_j,$$

para $T_2 = \{1, 2\}$, $\alpha_{21} = 1$, $J_{21} = \{1, 1\}$, $\alpha_{2,2} = -3$ e $J_{22} = \{2\}$. O grao do problema é o maior dos graos dos monomios que se atopan nel, neste caso correspóndese co monomio $x_1x_2^2$, logo $\delta = 3$. Polo tanto $N^\delta = \{1, 1, 1, 2, 2, 2\}$ e contén a tódolos multiconxuntos posibles asociados ós monomios do problema.Para rematar, as variables son non negativas e están acoutadas superiormente por 10, logo temos que $\Omega = [0, 10] \times [0, 10]$.

Aínda que hai problemas polinómicos convexos (e incluso lineais), en xeral os problemas de optimización polinómica son non lineais e non convexos. Por este motivo, obter unha solución global resulta unha tarefa moi complexa pois, de feito, trátanse de problemas $\mathcal{NP}$-duros. Non entraremos máis en detalle na complexidade computacional mais poden consultarse Papadimitriou and Steiglitz (1998), onde se explica en detalle este concepto, e Laurent (2009), quen comenta e exemplifica a complexidade desta clase de problemas. Tamén existen problemas de optimización polinómica con variables enteiras, non obstante nos próximos capítulos suporemos que as variables son continuas. Ademais consideraremos $\delta \geq 2$, centrando o noso estudo en problemas non lineais.

Capítulo 2

Técnica de Reformulación-Linearización (RLT)

2.1. Introdución

Para obter solucións locais dun problema de optimización polinómica existen moitos métodos e software dispoñibles. Non obstante, obter unha solución global desta clase de problemas é, en xeral, unha tarefa complexa. Existen moitas investigacións nesta liña, a meirande parte delas baséanse en transformar o problema orixinal noutro para o cal sexamos capaces de obter unha solución. Por exemplo, mediante unha relaxación linear ou convexa. Agora ben, este novo problema pode non ser equivalente ó orixinal. Neste caso, é necesario realizar algún procedemento iterativo, como resolver unha xerarquía de problemas ou técnicas de ramificación e acoutamento, para finalmente obter a solución global do problema orixinal.

Neste capítulo introduciremos a Técnica de Reformulación-Linearización (RLT, do inglés *Reformulation-Linearization Tecnique*), presentada por Sherali and Tuncbilek (1992) e estudada en González-Rodríguez (2022) e González-Rodríguez et al. (2023). Este método parte dunha relaxación linear do problema polinómico, substituíndo os monomios non lineais por novas variables e engadindo unhas novas restricións. A continuación, divide sucesivamente a rexión factible mediante un algoritmo de ramificación espacial e acoutamento obtendo cotas superiores e inferiores do problema orixinal. A idea esencial da ramificación espacial é semellante á empregada en optimización enteira, se ben neste último caso se ramifica atendendo ás infactibilidades das variables segundo o seu carácter enteiro, agora faise tendo en conta a equivalencia das novas variables coa sua expresión non linear correspondente.

Aínda que nos centraremos na Técnica de Reformulación-Linearización, na que se basea o *solver* global `RAPOSa` (González-Rodríguez et al., 2023) e que detallaremos formalmente máis adiante, é interesante coñecer que outros métodos teóricos existen para resolver un problema polinómico. Polo tanto, comentaremos brevemente a idea dalgúns deles:

1. Unha forte liña de investigación neste ámbito busca transformar de xeito eficiente un problema polinómico noutro de grao dous, máis concretamente, nun problema cuadrático con restricións cuadráticas (QCQP). Resolver un problema cuadráti-

co pode ser máis sinxelo que un polinómico dun grao maior. Non obstante, segue tratándose de problemas $\mathcal{NP}$-duros (Karia et al., 2022), polo que é necesario empregar posteriormente técnicas como a RLT ou relaxacións convexas para resolvelos. Por este motivo, os métodos de redución de grao son complementarios ás demais técnicas de resolución de problemas polinómicos, como a RLT ou as liñas de investigacións seguintes.

Para facer unha transformación cuadrática modifícanse as restricións non lineais usando variables auxiliares e engádense novas restricións, de xeito que o problema orixinal sexa equivalente a outro de grao dous. Pódese consultar Dalkiran and Ghalami (2018), onde detallan diferentes maneiras de levar a cabo esta cuadrificación. En exemplos moi sinxelos é posible atopar, de maneira intuitiva, algunhas transformacións que permiten pasar do problema orixinal a un de grao dous equivalente. Non obstante, en xeral, esta tarefa require algoritmos específicos, como o deseñado por Karia et al. (2022). Recentemente, González-Rodríguez and Naoum-Sawaya (2024) propuxeron un novo procedemento, baseado na RLT, que permite reducir o grao dun problema polinómico. Isto é, transformar o problema orixinal noutro dun grado menor desexado, en particular, nun cuadrático. Ademais, implementaron e estudaron este método de redución de grao co *solver* `RAPOSa` (González-Rodríguez et al., 2023).

2. Outro enfoque dende o punto de vista da álxebra xeométrica, trouxo importantes avances na optimización polinómica. Algúns exemplos son Chang and Wah (1994) e Hägglöf et al. (1995) quen transforman o problema orixinal empregando multiplicadores de Lagrange e bases de Gröbner para finalmente resolver o problema resultante con métodos coñecidos como a eliminación de Gauss.

3. Outros resultados baséanse na resolución dunha xerarquía de relaxacións convexas como poden ser os problemas semidefinidos positivos (SDPs) ou os expresados como suma de cadrados de polinomios (SOS). Deste xeito, obtéñense novamente cotas do valor óptimo do problema polinómico orixinal. Unha vantaxe dalgúns destes métodos é a converxencia teórica nun número finito de pasos para algúns problemas (Nie, 2014). Non obstante, a día de hoxe, a resolución de SDPs é moito menos eficiente computacionalmente que a de LPs, como é o caso na RLT. Unha referencia básica é Lasserre (2001), quen propón unha relaxación SDP para un problema de optimización polinómica con rexión factible compacta, cunha posterior mellora presentada en Lasserre et al. (2017). Outro exemplo é Nie (2013), quen deseña outra relaxación SDP para o caso xeral, chegando no caso particular anterior, no que a rexión factible é compacta, ós mesmos resultados que os obtidos por Lasserre (2001).

4. Outros métodos teñen en conta a relación dos Problemas de Optimización Polinómica cos problemas signomiais (SPs), onde interveñen polinomios con expoñentes reais e variables positivas (signomios), ou xeométricos (GPs), onde ademais os coeficientes son tamén positivos (posinomios) cos problemas polinómicos. Algúns exemplos son Karaca et al. (2017) e Teles et al. (2013).

Observación 2.1. Como curiosidade, os resultados dalgunhas das liñas de investigación anteriormente mencionadas están fortemente relacionados coa teoría dos polinomios non negativos, amplamente coñecida e representada polo décimo-sétimo problema de Hilbert.

2.2. Aspectos teóricos

Antes de presentar o algoritmo baseado na Técnica de Reformulación-Linearización cómpre introducir algúns conceptos teóricos previos. Os principais resultados que se inclúen neste capítulo, así como as demostracións correspondentes, recóllense nas referencias González-Rodríguez (2022) e Sherali and Tuncbilek (1992).

Definición 2.2. Consideremos un problema de optimización polinómica da forma $(PP(\Omega))$. Podemos escribir as restrición de cota das variables como $x_j - l_j \geq 0$ e $u_j - x_j \geq 0$, para todo $j \in N$, e reciben o nome de ***bounding factors***.

Como xa comentamos, para formular a relaxación linear do problema $(PP(\Omega))$ substitúese cada monomio non linear por unha nova variable, que chamaremos variable RLT.

Definición 2.3. Sexa $J \subset N^\delta$ un monomio con $|J| \geq 2$ correspondente a un problema polinómico $(PP(\Omega))$ de grao δ. Defínese a **identidade RLT** correspondente como

$$X_J = \prod_{j \in J} x_j,$$

onde X_J é a **variable RLT** asociada.

Notación 2.4. Co fin de homoxeneizar a notación e facilitar a lectura empregaremos, cando conveña, $X_{\{j\}}$ ou X_j para referirnos a variable "orixinal" x_j.

Definición 2.5. Sexa $\boldsymbol{x}$ un punto, factible ou non, do problema $(PP(\Omega))$, definimos a súa extensión como

$$E(\boldsymbol{x}) = (X_J)_{J \subset N^\delta},$$

onde $X_J = \prod_{j \in J} x_j$, para todo $J \subset N^\delta$.

É dicir, a extensión dun punto do problema polinómico é un vector que contén os valores correspondentes a tódolos posibles monomios de grao menor ou igual ó do problema. En particular, contén o valor das variables orixinais do problema $(PP(\Omega))$, isto é, ó propio punto $\boldsymbol{x}$.

Definición 2.6. Sexa $\phi(\boldsymbol{x})$ unha función polinómica, definimos a súa **linearización** $[\phi(\boldsymbol{x})]_L$ como o polinomio resultante de substituír tódolos monomios non lineais polas súas variables RLT correspondentes.

Este concepto é un dos piares fundamentais da relaxación linear do problema polinómico. O outro é engadir novas restricións formadas polos *bounding factors*, que definimos a continuación.

Definición 2.7. Dado un problema de optimización polinómica da forma $(PP(\Omega))$. Defínense o conxunto de restricións ***bound factors*** como

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

para todo $J_1 \cup J_2 \subset N^\delta$ tal que $|J_1 \cup J_2| = \delta$.

Pode parecer trivial definir estas restricións, xa que son redundantes en $(PP(\Omega))$. Non obstante, na relaxación linear, cando substituímos as non linealidades polas variables RLT, as restricións *bound factors* non se deducen das demais. Deste xeito, no problema linealizado, aínda que non supoñen o cumprimento das igualdades RLT, esixen unha importante relación entra as variables orixinais e as RLT, fundamental para o funcionamento do algoritmo. Ademais, incluír todas estas restricións permite obter unha formulación máis axustada das sucesivas linealizacións. Pola contra, pode supoñer unha dificultade engadida á hora de buscar un algoritmo eficiente pois a medida que aumenta o número de variables e o grao do problema, o conxunto de *bound factors* aumenta exponencialmente, como demostra o seguinte resultado. Máis adiante veremos que é posible reducir o número de *bound factors*.

Proposición 2.8. *Dado un problema de optimización polinómica da forma $(PP(\Omega))$. O número de restricións bound factors do problema é*

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

Demostración. Queremos saber cantas restricións distintas se poden escribir da forma

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

para todo $J_1 \cup J_2 \subset N^\delta$ tal que $|J_1 \cup J_2| = \delta$. Isto é, queremos saber cantas eleccións distintas podemos facer do par de conxuntos J_1 e J_2 tales que os seus elementos están en N e a súa unión ten exactamente δ elementos. Ou o que é o mesmo, cal é o número de combinacións posibles de tamaño δ dun conxunto de $2n$ elementos con repetición, n posibles índices (que se poden repetir) para o conxunto J_1 e n para J_2. Sabemos que o número de combinacións posibles de m elementos tomados de l en l ven dado por

$$CR_m^l = \binom{m+l-1}{l},$$

logo o número de *bound factors* é

$$CR_{2n}^\delta = \binom{2n+\delta-1}{\delta}.$$

□

De xeito análogo podemos probar o seguinte resultado.

Proposición 2.9. *Dado un problema de optimización polinómica da forma $(PP(\Omega))$, o número de varibles RLT é*

$$\binom{n+\delta}{\delta} - (n+1).$$

Demostración. De acordo a Definición 2.3 e tendo en conta que o grao dos monomios é como moito δ, queremos saber cantos multiconxuntos $J \subset N^\delta$ distintos se poden construír de forma que $2 \leq |J| \leq \delta$. Posto que os monomios de grao 1 se corresponden coas n variables do problema, o número de multiconxuntos tales que $|J| < 2$ é $n+1$ (tendo en conta o conxunto baleiro). Polo tanto, pode resultar máis sinxelo achar o número de multiconxuntos tales que $|J| \leq \delta$, e restar posteriormente os $n+1$ anteriores. Por outra banda, se consideramos o 0 como un posible índice, podemos escribir os multiconxuntos asociados ós

monomios de grao menor que δ como multiconxuntos de tamaño δ engadindo tantos ceros como sexa necesario. Por exemplo, o monomio $x_1^2x_3$ asóciase con $J = \{0, 1, 1, 3\}$. Deste xeito, basta con calcular o número de combinacións con repetición de $n+1$ elementos tomados de δ en δ e finalmente restar os $n+1$ posibles multiconxuntos de tamaño 1. Con todo, temos que o número de variables RLT é

$$CR^{\delta}_{n+1} - (n+1) = \binom{n+\delta}{\delta} - (n+1),$$

como queriamos probar. □

Continuamos agora co problema polinómico do Exemplo 1.37 ilustrando os resultados vistos neste capítulo.

Exemplo 2.10. Consideremos o problema de optimización polinómica seguinte

$$\begin{array}{ll} \text{minimizar} & 2x_1 + x_1x_2 - x_1x_2^2 \\ \text{suxeito a} & x_1x_2 \geq 3 \\ & x_1^2 - 3x_2 = 1 \\ & 1 \leq x_1, x_2 \leq 10. \end{array}$$

Neste caso $n = 2$ e $\delta = 3$, logo o número de variables RLT é

$$\binom{2+3}{3} - (2+1) = 10 - 3 = 7$$

e o de *bound factors* é

$$\binom{2 \cdot 2 + 3 - 1}{3} = 20.$$

As variables RLT son

$$X_{11} = x_1x_1, \qquad X_{12} = x_1x_2, \qquad X_{22} = x_2x_2,$$

$$X_{111} = x_1x_1x_1, \quad X_{112} = x_1x_1x_2, \quad X_{122} = x_1x_2x_2 \text{ e}$$

$$X_{222} = x_2x_2x_2.$$

Algúns dos *bound factors* son

$$(x_1-1)^3 \geq 0, \qquad (x_1-1)^2(x_2-1) \geq 0, \qquad (x_1-1)(x_2-1)^2 \geq 0,$$

$$(x_2-1)^3 \geq 0, \qquad (x_1-1)^2(10-x_1) \geq 0, \qquad (x_1-1)^2(10-x_2) \geq 0,$$

$$(x_1-1)(x_2-1)(10-x_1) \geq 0, \quad (x_1-1)(x_2-1)(10-x_2) \geq 0, \qquad \ldots$$

Definición 2.11. Sexa

$$\begin{array}{ll} \text{minimizar} & \phi_0(\boldsymbol{x}) \\ \text{suxeito a} & \phi_r(\boldsymbol{x}) \geq \beta_r, \quad r = 1, \ldots, R_1 \\ & \phi_r(\boldsymbol{x}) = \beta_r, \quad j = R_1 + 1, \ldots, R \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^n \end{array}$$

o problema de optimizacións polinómica ($PP(\Omega)$), definimos a súa **relaxación linear** como

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

Se consideramos un punto factible $\boldsymbol{x}$ do problema ($PP(\Omega)$), a súa extensión $E(\boldsymbol{x})$ é claramente factible do problema ($LP(\Omega)$). Partindo desta idea Sherali and Tuncbilek (1992) probaron os seguintes tres resultados, que sentan as bases teóricas do algoritmo RLT.

Lema 2.12. *Dado un problema polinómico da forma ($PP(\Omega)$) con valor óptimo $v\,[PP(\Omega)]$ e relaxación linear ($LP(\Omega)$), cuxo valor óptimo é $v\,[LP(\Omega)]$, tense que*

$$v\,[LP(\Omega)] \leq v\,[PP(\Omega)]\,.$$

É máis, se $(\boldsymbol{x}^*, \boldsymbol{X}^*)$ é unha solución óptima de ($LP(\Omega)$) verificando as identidades RLT, entón $\boldsymbol{x}^*$ é unha solución óptima de ($PP(\Omega)$).

Demostración. Denotamos por $\phi_0[PP(\Omega)](\cdot)$ e $\phi_0[LP(\Omega)](\cdot)$ as funcións obxectivo de ($PP(\Omega)$) e ($LP(\Omega)$), respectivamente. Sexa $\bar{\boldsymbol{x}}$ unha solución factible de ($PP(\Omega)$), entón $E(\bar{\boldsymbol{x}})$ é unha solución factible de ($LP(\Omega)$) e $\phi_0[PP(\Omega)](\bar{\boldsymbol{x}}) = \phi_0[LP(\Omega)](E(\bar{\boldsymbol{x}}))$, logo

$$v\,[LP(\Omega)] \leq v\,[PP(\Omega)]\,. \qquad (2.1)$$

Sexa $(\boldsymbol{x}^*, \boldsymbol{X}^*)$ unha solución óptima de ($LP(\Omega)$) verificando as identidades RLT, isto é, tal que

$$X_J = \prod_{j\in J} x_j, \text{ para todo } |J| \geq 2,$$

entón $(\boldsymbol{x}^*, \boldsymbol{X}^*) = E(\boldsymbol{x}^*)$ e $\boldsymbol{x}^*$ é factible de ($PP(\Omega)$). Ademais, como vimos antes, tense que

$$\phi_0[PP(\Omega)](\boldsymbol{x}^*) = \phi_0[LP(\Omega)](E(\boldsymbol{x}^*)).$$

Polo tanto, tendo en conta a desigualdade (2.1) e o feito de que $E(\boldsymbol{x}^*)$ é óptimo de ($LP(\Omega)$), dedúcese que $\boldsymbol{x}^*$ é unha solución óptima de ($PP(\Omega)$). □

Lema 2.13. *As restricións bound factors da forma $[F_\delta(J_1, J_2)]_L \geq 0$, $J_1 \cup J_2 \subset N^\delta$, tales que $|J_1 \cup J_2| < \delta$ dedúcense das restricións $[F_\delta(J_1, J_2)]_L \geq 0$ con $J_1 \cup J_2 \subset N^\delta$ e $|J_1 \cup J_2| = \delta$.*

Demostración. Sexan δ' tal que $1 \leq \delta' < \delta$, $j \in N$ e J_1, J_2 tales que $J_1 \cup J_2 \subset N^\delta$ e $|J_1 \cup J_2| = \delta$. Consideremos as restricións $[F_{\delta'+1}(J_1 \cup \{j\}, J_2)]_L \geq 0$ e $[F_{\delta'+1}(J_1, J_2 \cup \{j\})]_L \geq 0$, entón tense que

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

Posto que $u_j - l_j \geq 0$, a restrición $[F_{\delta'}(J_1, J_2)]_L \geq 0$ dedúcese de $[F_{\delta'+1}(J_1 \cup \{j\}, J_2)]_L \geq 0$ e $[F_{\delta'+1}(J_1, J_2 \cup \{j\})]_L \geq 0$. En virtude do principio de indución, temos probado o enunciado. □

O Lema 2.13 afirma que todas as restricións que se poden formar multiplicando ata $\delta - 1$ *bounding factors* xa están incluídas polas restricións *bound factor*.

Corolario 2.14. *Sexa ($LP(\Omega)$) o problema relaxado dun problema de optimización polinómica da forma ($PP(\Omega)$). Entón, todas as variables do problema ($LP(\Omega)$) están acoutadas. Ademais, se o problema ($LP(\Omega)$) é factible, este ten óptimo finito.*

Demostración. Sexa X_J unha variable RLT arbitraria, con $X_J \subset N^\delta$. Vexamos por indución sobre $|J|$ que X_J ten cotas inferior e superior finitas. Se $|J| = 2$, con $J = \{j_1, j_2\}$, en virtude do Lema 2.13 sabemos que

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

logo $X_J \geq l_{j_2}x_{j_1} + l_{j_1}x_{j_2} - l_{j_1}l_{j_2}$. Posto que as variables x_{j_1} e x_{j_2} están acoutadas, dedúcese que a variable RLT ten unha cota inferior finita. De xeito análogo, considerando $J_1 = j_1$ e $J_2 = j_2$ tense que

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

logo $X_J \leq u_{j_2}x_{j_1} + l_{j_1}x_{j_2} - l_{j_1}u_{j_2}$, de onde se deduce que X_J está acoutado superiormente. Supoñamos que se verifica para $|J| < \delta'$ e vexamos que se cumpre para $|J| = \delta'$, con $\delta' \leq \delta$ e $J = \{j_1, \ldots, j_{\delta'}\}$. Do mesmo xeito que para o primeiro caso, grazas o Lema 2.13, tense que

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

sendo $f(\boldsymbol{x}, \boldsymbol{X})$ unha función linear cuxos monomios asociados ás variables RLT con coeficientes non nulo teñen grao menor que *delta'*, polo que están acoutadas. Logo $X_J \geq -f(\boldsymbol{x}, \boldsymbol{X})$ e X_J ten unha cota inferior finita. Por último, se consideramos $J_1 = J \setminus \{j_{\delta'}\}$ e $J_2 = \{j_{\delta'}\}$, temos que

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

onde $g(\boldsymbol{x}, \boldsymbol{X})$, por un razoamento análogo o anterior, está acoutada e $X_J \leq g(\boldsymbol{x}, \boldsymbol{X})$. Logo X_J está acoutado superiormente. □

Lema 2.15. *Sexa $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ unha solución factible do problema $(LP(\Omega))$. Se $\bar{x}_p = l_p$ para algún $p \in N$, entón*

$$\bar{X}_{J \cup \{p\}} = l_p \bar{X}_J,$$

para todo $J \subset N^\delta$ tal que $1 \leq |J| \leq \delta - 1$. Analogamente, se $\bar{x}_p = u_p$, entón

$$\bar{X}_{J \cup \{p\}} = u_p \bar{X}_J,$$

para todo $J \subset N^\delta$ tal que $1 \leq |J| \leq \delta - 1$.

Demostración. Sexa $\bar{x}_p = l_p$ para algún $p \in N$. Imos probar este resultado por indución sobre $|J|$. Comezamos considerando $|J| = 1$, logo $J = \{q\}$ para algún $q \in N$. En virtude do Lema 2.13, sabemos que as restricións *bound factors* de maior grao do problema $(LP(\Omega))$ implican que

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

e

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

Logo, tense que

$$l_q(x_p - l_p) + l_q x_q \leq X_{\{p,q\}} \leq l_p x_q + u_q(x_p - l_p).$$

Avaliando a expresión anterior no punto $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ e tendo en conta que $\bar{x}_p = l_p$, temos que $\bar{X}_{J\cup\{p\}} = \bar{X}_{\{p,q\}} = l_p \bar{x}_q$. Supoñamos que isto se verifica para $1 \leq |J| \leq t-1$ e vexámolo para $|J| = t$, con $2 \leq t \leq \delta - 1$. Para todo $q \in J$, polo Lema 2.13 sabemos que

$$[(x_p - l_p)(x_q - l_q) \prod_{j \in J\setminus\{q\}} (x_j - l_j)]_L \geq 0$$

e

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

Reescribindo $\left[\prod_{j\in J\setminus\{q\}} (x_j - l_j)\right]_L$ como $X_{J\setminus\{q\}} + \phi(\boldsymbol{x})$, onde $\phi(\boldsymbol{x})$ é un polinomio de grao menor ou igual a $t-2$, tense que

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

e

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

Pola hipótese de indución tiñamos que $\bar{X}_{J\setminus\{q\}} = l_p \bar{X}_{J\setminus\{q\}}$, logo avaliando as desigualdades anteriores en $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ e tendo en conta que $\bar{x}_p = l_p$, temos que

$$0 \leq X_{J\cup\{p\}} - l_p X_J \leq 0.$$

Polo tanto, $\bar{X}_{J\setminus\{p\}} = l_p \bar{X}_J$ como queiramos probar.

Se $\bar{x}_p = u_p$ a demostración é análoga, polo que non a incluiremos. □

2.3. Algoritmo baseado na RLT

Os resultados anteriores, ademais de probar algúns aspectos máis técnicos, son fundamentais para o funcionamento do algoritmo. En particular, o Lema 2.13 afirma que a solución óptima do problema linealizado, obtida mediante as técnicas habituais de optimización linear, xera unha cota inferior (LB, do inglés *lower bound*) de $(PP(\Omega))$. No mellor dos casos, se a solución óptima de $(LP(\Omega))$ $(\boldsymbol{x}^*, \boldsymbol{X}^*)$ cumpre as igualdades RLT, a solución é óptima de $(PP(\Omega))$. Noutro caso, a infactibilidade débese a que, polo menos, unha igualdade RLT non se satisfai. Entón, chegados a este punto, buscamos unha variable x_p que maximice dalgunha maneira as violacións das igualdades RLT para ramificar sobre ela. Sherali and Tuncbilek (1992) definiron o criterio de ramificación dado por

$$p \in \arg\max_{j \in N}\{\theta_j\}, \tag{2.2}$$

onde

$$\theta_j = \max_{J \subset N^\delta, 1 \leq |J| \leq \delta - 1} \left\{ |\bar{X}_{J \cup \{j\}} - \bar{x}_j \bar{X}_J| \right\},$$

para todo $j \in N$. Posteriormente, dividimos Ω nos hiperrectángulos Ω^1 e Ω^2 resultantes de engadir a restrición $x_p \leq x_p^*$ e $x_p \geq x_p^*$, respectivamente. Esta etapa, que se coñece como ramificación espacial, permite descartar un óptimo de $(LP(\Omega))$ que non é factible (nin óptimo) de $PP(\Omega)$, xerando dous novos problemas $LP(\Omega^1)$ e $LP(\Omega^2)$ e a súa vez novas cotas. Pois ben, este procedemento de xeito iterativo é o que se coñece como Algoritmo de Reformulación-Linearización e se detalla a continuación:

Etapa 0: Inicialización.

- Definimos a mellor solución actual $x^* = \emptyset$ e o mellor valor obxectivo como $v^* = \infty$. Fixamos $k = 1$, $t = 1$, $tot = 1$, $Q_1 = \{1\}$ e $\Omega^{1,1} = \Omega$.

- Resolvemos $LP(\Omega^{1,1})$, obtendo unha solución óptima $(\bar{\boldsymbol{x}}^{1,1}, \bar{\boldsymbol{X}}^{1,1})$ de $LP(\Omega^{1,1})$. Se $LP(\Omega^{1,1})$ é infactible, entón o problema orixinal $(PP(\Omega))$ tamén o é e o algoritmo remata.

- Seleccionamos a variable x_p sobre a que ramificamos de acordo ó criterio previamente fixado. Se $\theta_p^{1,1} = 0$, $\boldsymbol{x}^{1,1}$ é unha solución óptima de $(PP(\Omega))$. Polo tanto, actualizamos $\boldsymbol{x}^* = \boldsymbol{x}^{1,1}$ $v^* = v\left[LP\left(\Omega^{1,1}\right)\right]$ e o algoritmo remata. En caso contrario, pasamos á Etapa 1.

Etapa 1: Ramificación.

Consideramos o problema $LP\left(\Omega^{k,t}\right)$, para o cal se seleccionou x_p como variable de ramificación con $\theta_p^{k,t} > 0$.

- Defínese a partición de $\Omega^{k,t}$:

$$\begin{aligned} \Omega^{k,t_1} &= \Omega^{k,t} \cap \left\{ x \in \mathbb{R}^n : l_p^{k,t} \leq x_p \leq x_p^{k,t} \right\}, \\ \Omega^{k,t_2} &= \Omega^{k,t} \cap \left\{ x \in \mathbb{R}^n : x_p^{k,t} \leq x_p \leq u_p^{k,t} \right\}, \end{aligned} \tag{2.3}$$

  sendo $t_1 = tot + 1$ e $t_2 = tot + 2$. Nótese que os novos conxuntos están ben definidos xa que, en virtude do Lema 2.15, por ser $\theta_p^{k,t} > 0$ tense que $l_p^{k,t} < x_p^{k,t} < u_p^{k,t}$ sendo $l_p^{k,t}$ e $u_p^{k,t}$ as cotas inferior e superior, respectivamente, de x_p en $LP(\Omega^{k,t})$.

- Actualizamos $Q_k = (Q_k \backslash \{t\}) \cup \{t_1, t_2\}$, $tot = tot + 2$, e continuamos á Etapa 2.

Etapa 2a: Acoutamento.

Resolvemos $LP\left(\Omega^{k,t_1}\right)$. Se é infactible, actualizamos $Q_k = Q_k \backslash \{t_2\}$ e continuamos á Etapa 2b. Noutro caso, temos que $(\bar{\boldsymbol{x}}^{k,t_1}, \bar{\boldsymbol{X}}^{k,t_1})$ é a solución óptima de $LP(\Omega^{k,t_1})$ cun valor óptimo de $LB_{k,t_1} = v[LP(\Omega^{k,t_1})]$. Seleccionamos a variable de ramificación x_p.

- Se $\theta_p^{k,t_1} = 0$, entón $\bar{\boldsymbol{x}}^{k,t_1}$ é factible de $(PP(\Omega))$, o que nos proporciona unha cota superior de $(PP(\Omega))$. Actualizamos $Q_k = Q_k \backslash \{t_1\}$. Se $v[LP(\Omega^{k,t_1})] < v^*$, entón $\boldsymbol{x}^* = \boldsymbol{x}^{k,t_1}$ e $v^* = LB_{k,t_1}$.

Etapa 2b: Acoutamento.

Resolvemos $LP\left(\Omega^{k,t_2}\right)$. Se é infactible, actualizamos $Q_k = Q_k \backslash \{t_2\}$ e continuamos á Etapa 3. Noutro caso, temos que $(\bar{\boldsymbol{x}}^{k,t_2}, \bar{\boldsymbol{X}}^{k,t_2})$ é a solución óptima de $LP(\Omega^{k,t_2})$ cun valor óptimo de $LB_{k,t_2} = v[LP(\Omega^{k,t_2})]$. Seleccionamos a variable de ramificación x_p.

- Se $\theta_p^{k,t_2} = 0$, entón $\bar{\boldsymbol{x}}^{k,t_2}$ é factible de $(PP(\Omega))$, o que nos proporciona unha cota superior de $(PP(\Omega))$. Actualizamos $Q_k = Q_k \backslash \{t_2\}$. Se $v[LP(\Omega^{k,t_2})] < v^*$, entón $\boldsymbol{x}^* = \boldsymbol{x}^{k,t_2}$ e $v^* = LB_{k,t_2}$.

Etapa 3: Poda.

Actualizamos $Q_{k+1} = Q_k \backslash \{t \in Q_k : LB_{k,t} \geq v^*\}$.

- Se $Q_{k+1} = \emptyset$, o algoritmo remata.
- Noutro caso, para cada $t \in Q_{k+1}$, actualizamos $\Omega^{k+1,t} = \Omega^{k,t}$, $\left(\boldsymbol{x}^{k+1,t}, \boldsymbol{X}^{k+1,t}\right) = \left(\boldsymbol{x}^{k,t}, \boldsymbol{X}^{k,t}\right)$, $LB_{k+1,t} = LB_{k,t}$, $\theta_p^{k+1,t} = \theta_p^{k,t}$ e $k = k+1$.

Etapa 4: Selección do nodo.

Seleccionamos (k,t) tal que $t \in \arg\min_{t \in Q_k} \{LB_{k,t}\}$ e volvemos á Etapa 1.

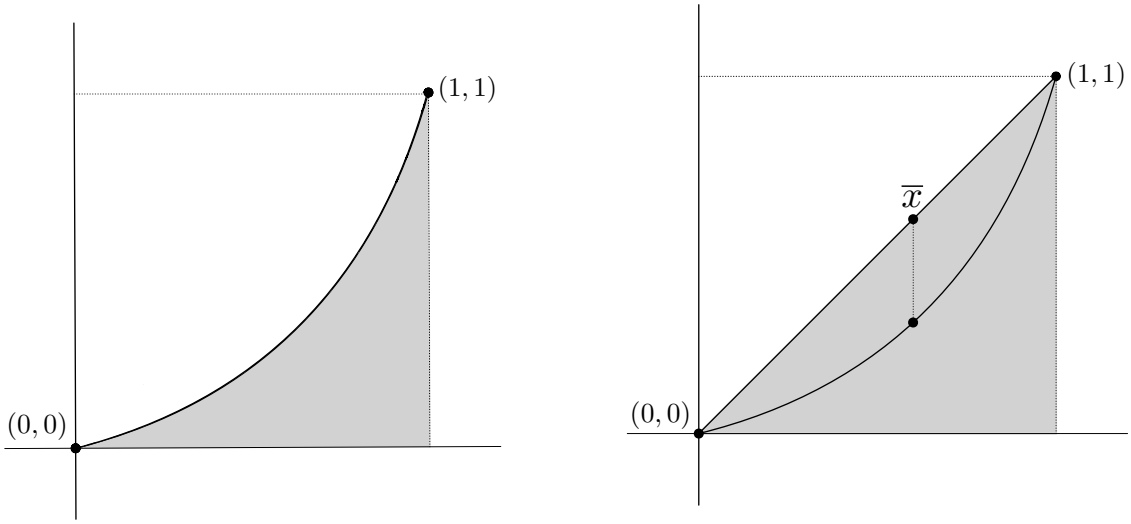
2.4. Interpretación gráfica

Nótese que a ramificación espacial non consiste en engadir directamente a restrición $x_p \leq x_p^*$ ou $x_p \geq x_p^*$ ó problema linealizado, senón que actualiza a cota inferior ou superior da variable en cuestión. Isto implica que algunhas restricións *bound factors* tamén cambien, reducindo aínda máis a rexión factible do problema linealizado. Imos ilustrar esta idea.

Sexa un problema polinómico de grao dous da forma $(PP(\Omega))$ con $\Omega = [0,1]^2$ dado. Supoñamos que a rexión factible é o espazo sombreado representado na Figura 2.1a. O problema relaxado terá 5 variables, as orixinais x_1 e x_2 máis as variables RLT X_{11}, X_{22} e X_{12}. Logo a rexión factible deste problema será un subconxunto de $\mathbb{R}^5$. Non obstante, se consideramos a súa proxección sobre o plano que contén as variables x_1 e x_2, podemos imaxinar que será semellante á rexión sombreada na Figura 2.1b. Como vemos, a linealización ou convexificación dun problema non linear ou non convexo, tamén ten unha representación gráfica moi clara. Ademais, unha vez resolvemos a relaxación linear, acadando o óptimo no punto $\bar{\boldsymbol{x}}$, ramificamos sobre unha das súas variables. Isto é, dividimos Ω actualizando unha das cotas da variable. Esto supón un cambio nas restricións *bound factor* e polo tanto na sua linealización. Polo tanto, non só se divide o espazo factible da relaxación linear, senón que se forma un novo espazo factible asociado a cada subproblema linear, como se representa na Figura 2.2.

2.5. Converxencia do algoritmo

Agora debemos preguntarnos se este algoritmo é capaz de obter unha solución óptima nunha cantidade finita de pasos, ou se polo menos temos asegurada a converxencia do mesmo. Para dar resposta a esta cuestión Sherali and Tuncbilek (1992) presentan e demostran



(a) Rexión factible de $PP(\Omega)$ (b) Rexión factible de $LP(\Omega)$

Figura 2.1: Interpretación gráfica da linearización dun problema polinómico empregada na RLT.

o Teorema 2.17. No presente traballo, incluiremos a demostración de González-Rodríguez (2022), pois fai explícitos algúns argumentos que a orixinal non.

Lema 2.16. *En calquera iteración $k \in K$ do algoritmo baseado na RLT, sendo K un conxunto non necesariamente finito, tense que*

$$LB_{k,t(k)} \leq v[PP(\Omega)],$$

onde $t(k) \in Q_k$.

Demostración. Imos ver que, pola propia construción do algoritmo, nunca seleccionamos nodos con un valor óptimo maior que o do problema $(PP(\Omega))$. En cada iteración k do algoritmo, tense que:

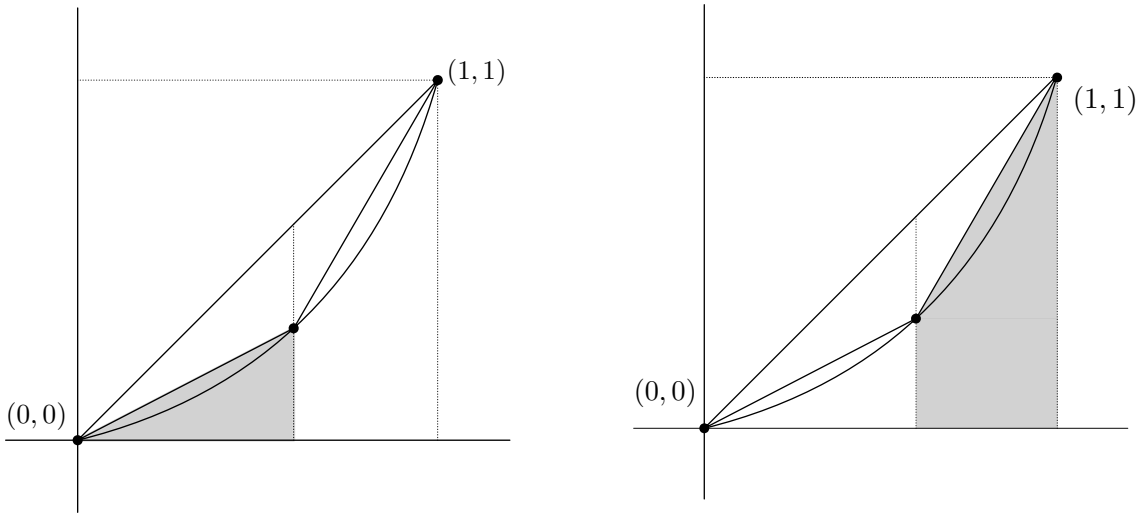
1. Todo punto $E(\boldsymbol{x})$ factible en $LP(\Omega^{k,t})$, e polo tanto $\boldsymbol{x}$ factible en $(PP(\Omega))$, é factible en $LP(\Omega^{k,t_1})$ ou en $LP(\Omega^{k,t_2})$.

2. Deixa de ramificarse nun nodo $LP(\Omega^{k,t(k)})$ cando é infactible ou cando o óptimo de $LP(\Omega^{k,t(k)})$ xa é factible de $(PP(\Omega))$.

3. Na Etapa 3, actualizamos $Q_{k+1} = Q_k \backslash \{t \in Q_k : LB_{k,t} \geq v^*\}$ de forma que descartamos aqueles nodos con valor óptimo maior ou igual que o da mellor solución ata ese momento.

4. Por último, seleccionamos o nodo sobre o que ramificamos escollendo en Q_k aquel co valor óptimo máis baixo.

□



(a) Rexión factible de $LP(\Omega_1)$ (b) Rexión factible de $LP(\Omega_2)$

Figura 2.2: Interpretación gráfica da ramificación espacial empregada no algoritmo baseado na RLT.

Teorema 2.17. *Cando empregamos o algoritmo RTL para resolver un problema de optimización da forma ($PP(\Omega)$) cúmprese sempre unha das seguintes afirmacións:*

1. *O algoritmo RLT remata nunha cantidade finita de pasos obtendo unha solución óptima de ($PP(\Omega)$).*
2. *Dada unha sucesión infinita de iteracións e unha rama con infinitos nodos xerada pola mesma, entón todo punto de acumulación da sucesión de variables $\boldsymbol{x}$ xerada pola resolución das correspondentes relaxacións lineais é un óptimo global de ($PP(\Omega)$).*

Demostración. Nótese que, na Etapa 1, os puntos factibles en $LP(\Omega^{k,t})$ que son factibles en $(PP(\Omega))$ tamén o son en $LP(\Omega^{k,t_1})$ ou en $LP(\Omega^{k,t_2})$. Polo tanto, a ramificación está ben definida, isto é, todo punto da rexión factible de $(PP(\Omega))$ foi estudado ou está pendente de selo. Ademais, o algoritmo remata (nunha cantidade finita de iteracións) cando $Q_k = \emptyset$. É dicir, cando despois de k iteracións o algoritmo resolveu tódolos nodos, salvo aqueles cuxos valores correspondentes son maiores ó óptimo actual. Logo, v^* é valor óptimo de $(PP(\Omega))$ e $\boldsymbol{x}^*$ a solución correspondente.

Supoñamos agora que o algoritmo RLT non remata nunha cantidade finita de iteracións. Consideremos unha rama con infinitos nodos da árbore de ramificación e acoutamento, dada pola sucesión $\{\Omega^{k,t(k)}\}_{k\in K}$ de particións aniñadas, onde $t(k) \in Q_k$ para todo $k \in K \subset \mathbb{N}$, sendo K o conxunto infinito de iteracións. Denotemos por $(\boldsymbol{x}^{k,t(k)}, \boldsymbol{X}^{k,t(k)})$ a solución óptima de $LP(\Omega^{k,t(k)})$. A sucesión $\{\boldsymbol{x}^{k,t(k)}\}$ está acoutada, posto que todos os elementos da sucesión $\boldsymbol{x}^{k,t(k)}$ pertencen a Ω, que está acoutado. Polo tanto, en virtude do Teorema de Bolzano-Weierstrass, existe unha subsucesión $\{\boldsymbol{x}^{k,t(k)}\}_{k\in K_1}$, con K_1 un conxunto infinito, converxente a un punto $\bar{\boldsymbol{x}}$, ou o que é o mesmo, $\bar{\boldsymbol{x}}$ é un punto de acumulación. Basta probar que $\bar{\boldsymbol{x}}$ é precisamente unha solución óptima de $(PP(\Omega))$.

Por ser $\left\{\Omega^{k,t(k)}\right\}_{k\in K_1}$ unha sucesión infinita, existe unha variable x_p sobre a que o algoritmo ramifica nun conxunto $K_2 \subset K_1$ de infinitas iteracións. É máis, existe un subconxunto $K_3 \subset K_2$ e un conxunto de índices $J' \subset N^\delta$, con $1 \leq |J'| \leq \delta - 1$, tal que

$$\max_{j\in N} \theta_j^{k,t(k)} = \theta_p^{k,t(k)} = |X_{J'\cup\{p\}}^{k,t(k)} - x_p^{k,t(k)} X_{J'}^{k,t(k)}|,$$

para todo $k \in K_3$. Entón, tense que,

$$\left|X_{J'\cup\{p\}}^{k,t(k)} - x_p^{k,t(k)} X_{J'}^{k,t(k)}\right| \geq \left|X_{J\cup\{j\}}^{k,t(k)} - x_j^{k,t(k)} X_J^{k,t(k)}\right| \tag{2.4}$$

para todo $k \in K_3, J \subset N^\delta$, $1 \leq |J| \leq \delta - 1$, e $j \in N$. Por outro lado, en virtude do Corolario 2.14, a sucesión

$$\left\{\left(\boldsymbol{x}^{k,t(k)}, \boldsymbol{X}^{k,t(k)}, \boldsymbol{l}^{k,t(k)}, \boldsymbol{u}^{k,t(k)}\right)\right\}_{k\in K_3}$$

está acoutada, logo existe unha subsucesión converxente

$$\left\{\left(\boldsymbol{x}^{k,t(k)}, \boldsymbol{X}^{k,t(k)}, \boldsymbol{l}^{k,t(k)}, \boldsymbol{u}^{k,t(k)}\right)\right\}_{k\in K_4} \to (\overline{\boldsymbol{x}}, \overline{\boldsymbol{X}}, \overline{\boldsymbol{l}}, \overline{\boldsymbol{u}})$$

onde $K_4 \subset K_3$. Ademais, tense que $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{X}})$ é factible en $LP(\bar{\Omega})$ sendo

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

Agora, por como se define a ramificación en (2.3), tense que $x_p^{k,t(k)} \notin \left(l_p^{k',t(k')}, u_p^{k',t(k')}\right)$, para todo $k, k' \in K_4$ con $k' > k$. Polo tanto, posto que $\bar{x}_p \in \left[\bar{l}_p, \bar{u}_p\right]$, necesariamente $\bar{x}_p = \bar{l}_p$ ou $\bar{x}_p = \bar{u}_p$. Logo, en virtude do Lema 2.15, sabemos que $\bar{X}_{J'\cup\{p\}} = \bar{x}_p \bar{X}_{J'}$. Entón, se tomando $k \to \infty$ e levamos (2.4) ó límite, temos que $\bar{X}_{J\cup\{j\}} = \bar{x}_j \bar{X}_J$ para todo $J \subset N^\delta, 1 \leq |J| \leq \delta - 1$, e $j \in N$. Con isto, temos probamos que $\bar{x}$ é factible en $(PP(\Omega))$, logo

$$[\phi_0]_L (\overline{\boldsymbol{x}}, \overline{\boldsymbol{X}}) = \phi_0(\overline{\boldsymbol{x}}) \geq v[PP(\Omega)].$$

Por último, empregando o Lema 2.16 e tomando novamente límites, temos que

$$v[PP(\Omega)] \geq [\phi_0]_L (\overline{\boldsymbol{x}}, \overline{\boldsymbol{X}}) = \phi_0(\overline{\boldsymbol{x}}).$$

Polo tanto, $\overline{\boldsymbol{x}}$ é un óptimo global de $(PP(\Omega))$, como queriamos ver.

□

A partir deste resultado dedúcese que, na práctica, non sempre podemos atopar unha solución óptima do noso problema. Non obstante, nestes casos un obxectivo razoable pode ser obter, nun tempo previamente fixado, un *gap* de optimalidade reducido, resultante de xerar cotas superiores e inferiores do valor óptimo.

2.6. Exemplo

A continuación, incluímos un detallado exemplo no que, mediante o algoritmo baseado na RLT, resolvemos un sinxelo problema de optimización polinómica.

Exemplo 2.18. Consideremos o problema de optimización polinómica

$$\begin{array}{ll} \text{minimizar} & x_2x_3 + x_2 + x_3 - x_1x_3 \\ \text{suxeito a} & -4x_1x_3 - x_2 \geq -12 \\ & 3x_1 - x_2 \geq -1 \\ & 0 \leq x_1, x_2, x_3 \leq 6. \end{array} \tag{2.5}$$

Logo $N = \{1, 2, 3\}$, $\Omega = [0, 6]^3$, o grao do problema é $\delta = 2$ e as variables RLT son

$$X_{11} = x_1x_1, \quad X_{12} = x_1x_2, \quad X_{13} = x_1x_3,$$

$$X_{22} = x_2x_2, \quad X_{23} = x_2x_3, \quad X_{33} = x_3x_3.$$

Construímos a relaxación do problema anterior engadindo as restricións *bound factors* e substituíndo os monomios non lineais polas variables RLT correspondentes:

$$\begin{array}{lll} \text{minimizar} & X_{23} + x_2 + x_3 - X_{13} & \\ \text{suxeito a} & -4X_{13} - x_2 \geq -12 & \\ & 3x_1 - x_2 \geq -1 & \\ & \left[\prod_{j\in J_1}(x_j - l_j)\prod_{j\in J_2}(u_j - x_j)\right]_L \geq 0, & \begin{array}{r}\forall J_1 \cup J_2 \subset N^2 \\ |J_1 \cup J_2| = 2\end{array} \\ & \boldsymbol{x} \in \Omega^{1,1} \subset \mathbb{R}^3 & \end{array} \tag{$LP(\Omega_{1,1})$}$$

- Etapa 0.
  - Inicializamos con $\boldsymbol{x} = \emptyset$ e $v^* = \infty$. Fixamos a primeira etapa, $k = 1$, o primeiro e único nodo t_1, $tot = 1$, $Q_1 = \{1\}$ e o conxunto $\Omega_{1,1} = \Omega$.
  - Resolvemos $(LP(\Omega_{1,1}))$ en AMPL empregando o *solver* `Gurobi`. A solución óptima é $(\boldsymbol{x}^{1,1}, \boldsymbol{X}^{1,1}) = (0.5, 0, 0.5, 0, 0, 3, 0, 0, 0)$.
  - Para determinar a variable sobre a que ramificaremos. Empregamos o criterio dado por Sherali and Tuncbilek (1992), para $(\boldsymbol{x}, \boldsymbol{X}) = (\boldsymbol{x}^{1,1}, \boldsymbol{X}^{1,1})$, calculamos:

    $$\theta_1 = \text{máx}\{|X_{11} - x_1^2|, |X_{12} - x_2x_1|, |X_{13} - x_3x_1|\} = \text{máx}\{0.25, 0, 2.75\} = 2.75,$$
    $$\theta_2 = \text{máx}\{|X_{12} - x_2x_1|, |X_{22} - x_2^2|, |X_{23} - x_2x_3|\} = \text{máx}\{0, 0, 0\} = 0,$$
    $$\theta_3 = \text{máx}\{|X_{13} - x_1x_3|, |X_{23} - x_2x_3|, |X_{33} - x_3x_3|\} = \text{máx}\{2.75, 0, 0.25\} = 2.75.$$

    Entre as variables x_1 e x_3, que maximizan as violacións, escollemos, por exemplo, a variable $x_p = x_1$.

- **Etapa 1**. Temos que $x_1^{1,1} = 0.5$, logo creamos a partición

$$\Omega^{1,2} = \Omega^{1,1} \cap \left\{x \in \mathbb{R}^3 : 0 \leq x_1 \leq 0.5\right\},$$
$$\Omega^{1,3} = \Omega^{1,1} \cap \left\{x \in \mathbb{R}^3 : 0.5 \leq x_1 \leq 6\right\}.$$

  Actualizamos $Q_1 = (Q_1 \backslash \{1\}) \cup \{2, 3\}$ e $tot = 3$.

- **Etapa 2a**. Resolvemos $LP\left(\Omega^{1,2}\right)$. Temos que

$$\left(\bar{\boldsymbol{x}}^{1,2}, \bar{\boldsymbol{X}}^{1,2}\right) = (0,0,0,0,0,0,0,0,0)$$

é a solución óptima de $LP(\Omega^{1,2})$ é $LB_{1,2} = v[LP(\Omega^{1,2})] = 0$. Ademais, $\bar{\boldsymbol{x}}^{1,2} = (0,0,0)$ é factible de $(PP(\Omega))$, pois $\theta_p^{1,2} = 0$, entón actualizamos $Q_1 = Q_1 \backslash \{2\} = \{3\}$. Posto que $v[LP(\Omega^{1,2})] = 0 < \infty = v^*$, actualizamos tamén $\boldsymbol{x}^* = (0,0,0)$ e $v^* = 0$.

- **Etapa 2b**. Repetimos o proceso para $\Omega^{1,3}$. Temos que

$$(\bar{\boldsymbol{x}}^{1,3}, \bar{\boldsymbol{X}}^{1,3}) = (0.96, 0, 0.5, 0.71, 0, 3, 0, 0, 0)$$

é a solución óptima de $LP(\Omega^{1,3})$ cun valor óptimo de $LB_{1,3} = v[LP(\Omega^{1,3})] = -2.5$. Escollemos a variable de ramificación $x_p = x_3$ tendo en conta que $\theta_1 = \theta_3 = 2.52$ e $\theta_2 = 0$.

- **Etapa 3**. Non é necesario podar a árbore pois temos que $Q_1 = \{3\}$ e o LB de $\Omega_{1,3}$ é menor co valor óptimo actual, isto é, podemos atopar unha solución mellor á actual neste espazo da rexión factible. Actualizamos $k = 2$, $\Omega^{2,3} = \Omega^{1,3}$, $\left(\boldsymbol{x}^{2,3}, \boldsymbol{X}^{2,3}\right) = \left(\boldsymbol{x}^{1,3}, \boldsymbol{X}^{1,3}\right)$, $LB_{2,3} = LB_{1,3}$ e $\theta_p^{2,3} = \theta_p^{1,3}$.

- **Etapa 4**. O único nodo pendente de explorar é o $(1,3)$ logo non é necesario facer unha selección.

- **Etapa 1** (k=2). Consideramos o problema $LP(\Omega_{2,3})$. A variable de ramificación é x_3 sendo $x_3^{2,3} = 0.5$, de acordo ó obtido en etapas anteriores, polo tanto definimos

$$\Omega^{2,4} = \Omega^{2,3} \cap \left\{x \in \mathbb{R}^3 : 0 \leq x_3 \leq 0.5\right\},$$
$$\Omega^{2,5} = \Omega^{2,3} \cap \left\{x \in \mathbb{R}^3 : 0.5 \leq x_3 \leq 6\right\}.$$

Actualizamos $Q_2 = (Q_2 \backslash \{3\}) \cup \{4,5\}$ e $tot = 5$.

- **Etapa 2a** (k=2). Resolvemos $LP\left(\Omega^{2,4}\right)$. Temos que

$$(\bar{\boldsymbol{x}}^{2,4}, \bar{\boldsymbol{X}}^{2,4}) = (6, 0, 0.5, 36, 0, 3, 0, 0, 0.25)$$

é a solución óptima de $LP(\Omega^{2,4})$ cun valor óptimo de $LB_{2,4} = v[LP(\Omega^{2,4})] = -2.5$. Comprobamos que $\bar{\boldsymbol{x}}^{2,4} = (6,0,0.5)$ é factible de $(PP(\Omega))$ e actualizamos $Q_2 = Q_2 \backslash \{4\} = \{5\}$. Ademais $v[LP(\Omega^{2,4})] = -2.5 < 0 = v^*$, logo actualizamos $\boldsymbol{x}^* = (6,0,0.5)$ e $v^* = -2.5$. En realidade, chegados este punto xa pechamos o *gap* de optimalidade a 0. Non obstante, posto que non fixamos un criterio de parada neste sentido, continuamos co algoritmo.

- **Etapa 2b** (k=2). Para $\Omega^{2,5}$ o resultado é o mesmo, pois neste caso a solución atópase na interxección dos subconxuntos $\Omega_{2,4}$ e $\Omega_{2,5}$, polo que actualizamos $Q_2 = Q_2 \backslash \{5\} = \emptyset$.

- **Etapa 3** (k=2). $Q_2 = \emptyset$ logo non quedan nodos por explorar. Rematamos o algoritmo obtendo a solución óptima $\boldsymbol{x}^* = (6,0,0.5)$ do problema (2.5) cun valor óptimo de -2.5.

Na Figura 2.3 ven representada a árbore de ramificación resultante de aplicar o algoritmo baseado RLT no Exemplo 2.18.

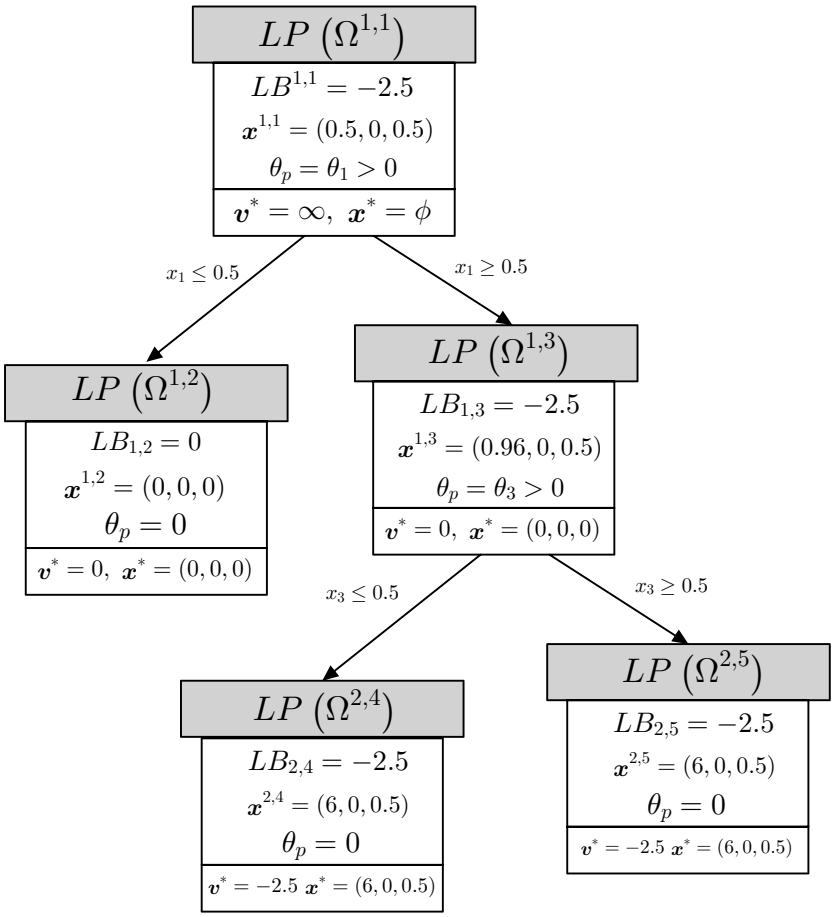


Figura 2.3: Esquema do algoritmo baseado na RLT no Exemplo 2.18

Capítulo 3

RAPOSa

3.1. Introdución

Partindo do algoritmo baseado na RLT, que acabamos de presentar, González-Rodríguez et al. (2023) implementan en `C++` un novo *solver* global especificamente deseñado para problemas de optimización polinómica chamado `RAPOSa` (do inglés, **R**eformulation **A**lgorithm for **P**olynomial **O**ptimization - **Sa**ntiago). Esta ferramenta, baseada no algoritmo RLT introducido por Sherali and Tuncbilek (1992), nútrese tamén das sucesivas melloras que os mesmos autores publicaron posteriormente, entre outras. De feito, Dalkiran and Sherali (2016) presentaron a súa propia implementación do algoritmo chamada `RLT-POS`, aínda que non está dispoñible actualmente. Malia as moitas investigacións no campo da optimización polinómica que atopamos na literatura existente, que nós coñezamos, `RAPOSa` é o único software gratuíto específico para optimización polinómica que é competitivo con outros *solvers* globais comerciais, no ámbito da optimización polinómica.

3.2. Melloras do algoritmo orixinal baseado na RLT

Como xa comentamos, González-Rodríguez et al. (2023) incorporan algunhas melloras ó algoritmo orixinal baseado na Técnica de Reformulación como o uso de J-sets, inicio en quente na resolución da relaxación linear ou cambios na regra de ramificación. A continuación, comentaremos brevemente algunhas destas ideas.

3.2.1. J-sets

No Capítulo 2 xa mencionabamos a complicación que supón introducir tódalas restricións *bound factors* na formulación da relaxación linear. Dalkiran and Sherali (2013) propoñen uns novos conxuntos, chamados J-sets, que permiten filtrar unha certa cantidade de restricións *bound factors*, dependendo do problema concreto, de xeito que se manteña a converxencia á un óptimo global. A idea é sinxela, incluír unicamente as restricións *bound factors* correspondentes ós monomios que figuren na formulación do problema polinómico orixinal. Máis formalmente, preséntase o seguinte resultado.

Proposición 3.1. *O Teorema 2.17 segue sendo certo se incluímos soamente as restricións bound factors*

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

para todo J_1 e J_2 tales que $J_1 \cup J_2 = J$ e o monomio $\prod_{j\in J} x_j$ aparece no problema ($PP(\Omega)$).

Demostración. Para probar a converxencia, na demostración do Teorema 2.17, requírese que, para un punto $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ factible do problema linear no que as variables orixinais toman o valor da cota superior ou inferior, se satisfagan tódalas as igualdades RLT (para comprobar que $\bar{\boldsymbol{x}}$ é factible de $(PP(\Omega))$). En realidade, bastaría con que se cumpran as igualdades RLT correspondentes ós monomios que aparecen no problema $(PP(\Omega))$. Polo tanto, basta con probar que isto se cumpre cando incluímos na formulación unicamente as *bound factor* do enunciado. Para elo, empregando as probas dos Lemas 2.13 e 2.15, comprobamos que a colección de restricións *bound factors*

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

para todo J_1 e J_2 tales que $J_1 \cup J_2 = J$ e o monomio $\prod_{j\in J} x_j$ aparece no problema $(PP(\Omega))$, garanten que o punto $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ satisfaga as igualdades RLT asociadas a tódolos monomios que figuran nestas restricións. En particular, cúmprense as igualdades RLT asociadas ós monomios incluídos no problema, logo $\bar{\boldsymbol{x}}$ é factible de $(PP(\Omega))$ como queriamos probar. □

É máis, o Lema 2.13 permítenos excluír tamén as restricións *bound factor* asociadas ós monomios que están xa incluídos noutras restricións. Isto é, podemos excluír aquelas restricións *bound factor* asociadas ós multiconxuntos $J' \subsetneq J$, sendo $\prod_{j\in J} x_j$ un monomio presente no problema $(PP(\Omega))$. Deste xeito, redúcese notablemente o número de restricións da relaxación linear $LP(\Omega)$ e dos subproblemas lineais asociados. Esto implica unha considerable redución do tempo que require o algoritmo para procesar e resolver en cada iteración o problema linear. Pola contra, excluír estas restricións da relaxación linear fai que a formulación sexa menos axustada. Polo tanto, aínda tendo asegurada a converxencia do algoritmo, esta pode ser máis lenta xa que as cotas inferiores, resultantes da resolución dos problemas lineais, poden non ser tan boas. A pesar deste inconveniente, o uso dos J-sets supón unha clara mellora no rendemento do algoritmo, como se pode comprobar no detallado estudo computacional elaborado por González-Rodríguez (2022).

Warm start dos J-sets

Por outra banda, no algoritmo baseado na RLT, un problema linear e o subproblema correspondente diferéncianse nas cotas da variable sobre a que ramifica e nas restricións *bound factor* asociadas á mesma. Deste xeito, na implementación do algoritmo, para crear cada un dos subproblemas lineais González-Rodríguez et al. (2023) identifican e actualizan unicamente as *bound factors* necesarias, en lugar de construír tódalas restricións *bound factor* que indica inicialmente o J-set. Con esta mellora, conseguen reducir notablemente o tempo que o software precisa internamente para xerar cada un dos sucesivos problemas lineais. Polo tanto, redúcese o tempo que `RAPOSa` require para obter un óptimo global, ou no seu defecto para, fixado un tempo máximo, reducir o *gap* de optimalidade.

Observación 3.2. Nótese que isto non supón un cambio no algoritmo de optimización no que está baseado, polo que a árbore de ramificación non varía. Isto é, non se trata dunha variación teórica senón dun mellora no rendemento resultante da propia implementación.

3.2.2. Produto de restricións

Para obter unhas relaxacións lineais máis axustadas Sherali and Tuncbilek (1992) propoñen a construción de novas restricións. Por exemplo, incluír aquelas resultantes de

multiplicar *bounding factors* (Definición 2.2) con restricións de desigualdade da forma $\phi_r(\boldsymbol{x}) \geq \beta_r$, que non superen o grao do problema. Outra idea, tamén mencionada en Sherali and Tuncbilek (1997), é considerar o produto de restricións de igualdade da forma $\phi_r(\boldsymbol{x}) = \beta_r$ con variables x_j, $j = 1, \ldots, n$, ata obter un polinomio do mesmo grao que o problema.

Observación 3.3. Véxase que, aínda que estas restricións son máis esixentes que as orixinais en $(PP(\Omega))$, isto non ten porque ser certo ao facer a linearización nas diversas relaxacións lineais. Por iso, malia que se engadan estas novas restricións, é necesario incluír na formulación do problema linear todas aquelas que figuren orixinalmente en $(PP(\Omega))$.

Unha formulación máis axustada, implica un crecemento máis rápido da cota inferior do valor óptimo. Non obstante, unha complicada formulación da relaxación linear podería dificultar a resolución dos sucesivos problemas lineais e así aumentar o tempo de resolución. Polo tanto, do mesmo xeito que ocorría coas restricións *bound factors*, é importante controlar axeitadamente o número de restricións que se engaden ao deseñar a relaxación linear.

Para implementar estas ideas en `RAPOSa`, González-Rodríguez et al. (2023) levan a cabo dúas estratexias. A primeira delas é multiplicar restricións, xa sexan de desigualdade con *bounding factors* ou ben de igualdade con varias variables, que teñan tantas variables en común como sexa posible. Pola contra, outra posibilidade é multiplicar restricións que teñen o menor número de variables en común, de maneira que se establezan novas relacións entre estas. En tódolos casos, baixo as premisas de non xerar novas variables RLT, o que daría lugar a máis restricións *bound factor*, e non aumentar o grao do problema. Finalmente, incorpórase a segunda estratexia, cun maior grao de mellora, ao deseño final do *solver*.

3.2.3. Cortes SDP

A idea de formar novas restricións pódese xeneralizar, dando lugar ó que e coñece como cortes válidos.

Definición 3.4. Unha restrición linear da forma $f(\boldsymbol{x}, \boldsymbol{X}) \geq 0$ (ou $f(\boldsymbol{x}, \boldsymbol{X}) = 0$) é un **corte válido** para $(PP(\Omega))$ se, para cada punto factible $\boldsymbol{x}$ en $(PP(\Omega))$, tense que $f(\boldsymbol{x}, \boldsymbol{X}) \geq 0$ $(f(\boldsymbol{x}, \boldsymbol{X}) = 0)$.

Ademais, incluír cortes válidos non compromete a converxencia do algoritmo baseado na RLT, como se pode comprobar no seguinte resultado, extraído de González-Rodríguez (2022).

Teorema 3.5. *Sexa* $\{f_s(\boldsymbol{x}, \boldsymbol{X}) \geq 0\}_{s \in S_1}$ *e* $\{f_s(\boldsymbol{x}, \boldsymbol{X}) = 0\}_{s \in S_2}$ *unha colección de cortes válidos. Entón, pódese engadir calquera subcolección destes cortes válidos a calquera dos problemas lineais xerados polo algoritmo baseado na RLT, mantendo a converxencia ó óptimo global.*

Demostración. En primeiro lugar, vexamos que o Lema 2.12 segue sendo certo. Incluír cortes válidos non supón un cambio na función obxectivo de $(LP(\Omega))$. Ademais, para calquera $\boldsymbol{x}$ factible de $(PP(\Omega))$, $E(\boldsymbol{x})$ segue sendo factible de $(LP(\Omega))$, pois un punto factible $\boldsymbol{x}$ satisfai, por definición, calquera corte válido. O Lema 2.13 tamén é certo, independentemente da subcolección de cortes válidos engadida, pois este só involucra as restricións *bound factor*. O mesmo ocorre co Lema 2.15, cuxa proba só depende das restricións da relaxación linear de tipo *bound factor*, logo segue sendo un resultado válido. Por último,

o Lema 2.16 tamén é valido pois os argumentos empregados na súa demostración tamén o son. En particular, o argumento 1 é certo xa que, por definición de corte válido, un punto factible satisfai calquera corte válido. Con todo, temos probado que os argumentos empregados na demostración do Teorema 2.17 seguen sendo certos, logo temos asegurada a converxencia do algoritmo. □

Algúns cortes válidos poden obterse como resultado de multiplicar restricións xa existentes, como vimos nos apartados anteriores. Pero tamén existen outras estratexias, como os cortes semidefinidos positivos (SDP). Esta mellora, introducida por Sherali and Fraticelli (2002) e desenvolta por Sherali et al. (2012), parte da construción dunha matriz da forma $\boldsymbol{M} = [\boldsymbol{y}\boldsymbol{y}^T]$, sendo $\boldsymbol{y}$ un vector definido empregando variables ou produto destas. Así construída, esta matriz é semidefinida positiva en calquera solución factible, por ser o produto de un vector por si mesmo. Intercambiando os monomios de M polas súas variables RLT correspondentes obtense a matriz $\boldsymbol{M}_L = [\boldsymbol{y}\boldsymbol{y}^T]_L$. Avaliando a matriz $\boldsymbol{M}_L$ nunha solución dunha relaxación linear que non sexa factible de $(PP(\Omega))$, obtense unha matriz $\bar{\boldsymbol{M}}_L$ que non é necesariamente semidefinida positiva, ou o que é o mesmo, pode existir un vector $\boldsymbol{\alpha}$ tal que $\boldsymbol{\alpha}^T\bar{\boldsymbol{M}}_L\boldsymbol{\alpha} < 0$. Neste caso, engádese a restrición $\boldsymbol{\alpha}^T\bar{\boldsymbol{M}}_L\boldsymbol{\alpha} \geq 0$ á relaxación linear.

González-Rodríguez et al. (2023) implementan e analizan coidadosamente o uso de cortes SDP, utilizando diferentes vectores $\boldsymbol{y}$ de xeito que non se requira engadir novas variables RLT. Os resultados experimentais indican que, se ben o uso desta variación individualmente pode significar unha mellora do algoritmo, incluír cortes SDP xunto con outras melloras ten o efecto contrario.

Na versión de `RAPOSa` que se vai empregar no próximo capítulo non se empregaran cortes SDP polo que non afondaremos máis nesta ferramenta. Para máis detalle, poden consultarse as referencias anteriormente citadas.

3.2.4. NLP solver

Outra das propostas que mencionan Dalkiran and Sherali (2016) é o uso dun *solver* local non linear en certos nodos da árbore de ramificación, empregando como solución inicial a correspondente á menor cota inferior actual. Isto permite xerar novas cotas superiores e, polo tanto, reducir o *gap* de optimalidade con maior velocidade. Deste xeito, tamén se reduce así o tempo de resolución, como se demostra numericamente en González-Rodríguez et al. (2023). En `RAPOSa`, emprégase por defecto o *solver* local non lineal `Ipopt` (Wächter and Biegler, 2006) periodicamente despois de explorar unha certa cantidade de nodos, usándoo sempre no nodo raíz. Como observación, no estudo computacional González-Rodríguez et al. (2023) comprobaron que a elección do *solver* non linear empregado non influía na eficiencia do algoritmo.

3.2.5. LP solver

Para a resolución dun problema polinómico, o algoritmo baseado na RLT require en moitas ocasións da resolución dunha gran cantidade de problemas lineais. Por iso, aínda que individualmente o tempo que leva resolver un problema linear é relativamente pequeno, convén reducir o tempo de resolución destes problemas. Para elo, González-Rodríguez et al. (2023) fixeron varias probas. En primeiro lugar, comparouse o funcionamento do software alternando o *solver* linear empregado. O *solver* cun mellor rendemento foi `Gurobi`

(Gurobi Optimization, LLC, 2023), que se establece como o *solver* non linear por defecto.

Do mesmo xeito que se aproveitou a forte relación entre un problema linear e o subproblema seguinte para mellorar a eficiencia perfeccionando a implementación, González-Rodríguez et al. (2023) estudan o impacto computacional ó resolver cada subproblema linear tendo en conta a solución do problema do que parte. No caso de `Gurobi`, ao igual que outros *solvers*, dispón de unha opción chamada *LPWarmStart* que permite empregar a solución óptima ou a base asociada para a resolución dos subproblemas dun problema linear. O estudo computacional sobre o uso do *warm start* parece indicar que o seu uso está recomendado unicamente en problemas sinxelos, polo que non se empregará esta opción.

Outra opción interesante, que figura por defecto en `Gurobi`, é o *presolve*. Esta, permite que o *solver* linear transforme o problema noutro equivalente, máis sinxelo de resolver, para obter finalmente a solución do problema orixinal. Por defecto en `RAPOSa`, esta opción non está desactivada. Non obstante, como xa comentaremos, no Capítulo 4 non se empregará esta opción do *solver* linear.

3.2.6. Regra de ramificación

Recordemos que no algoritmo orixinal baseado na RLT, Sherali and Tuncbilek (1992) definiron, dada unha solución $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ da relaxación linear, o criterio de ramificación dado por

$$p \in \arg\max_{j \in N}\{\theta_j\}, \tag{3.1}$$

onde

$$\theta_j = \max_{J \subset N^\delta, 1 \leq |J| \leq \delta - 1} \left\{ |\bar{X}_{J\cup\{j\}} - \bar{x}_j \bar{X}_J| \right\},$$

para todo $j \in N$. Posteriormente, Dalkiran and Sherali (2016) presentan unha regra de ramificación similar á anterior, calculando θ_j como o sumatorio das violacións, en lugar do máximo destas, e multiplicándoo pola distancia mínima de $\bar{x}_j$ á cota superior ou inferior no nodo actual. Isto é, considerando tódalas violacións e dando un peso maior a aquelas variables máis afastadas das súas cotas.

González-Rodríguez et al. (2023) definen varias regras de ramificación, empregando en todas elas o sumatorio das violacións, aínda que con pesos diferentes. Dada unha solución $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ da relaxación linear,González-Rodríguez et al. (2023) propoñen ramificar sobre a variable $p \in \arg\max_{j\in N}\{\theta_j\}$, onde

$$\theta_j = \sum_{J \subset N^\delta, 1 \leq |J| \leq \delta - 1} w(j, J) |\bar{X}_{J\cup\{j\}} - \bar{x}_j \bar{X}_J|,$$

sendo $w(j, J)$ o peso correspondente a unha das seguintes opcións:

1. **Pesos constantes.** $w(j, J) = 1$ para todo j e J. Esta regra correspóndese coa opción "sum".

2. **Coeficientes.** $w(j, J) \equiv w(J)$, onde $w(J)$ é a suma do valor absoluto dos coeficientes do monomio J no problema linear.

3. **Valores duais.** $w(j, J)$ defínese como a suma do valor absoluto dos valores duais asociados ás restricións do problema linear que conteñen a J.

4. **Rango da variable.** Neste caso,

$$w(j, J) \equiv w(j) = \frac{u'_j - l'_j}{u_j - l_j},$$

onde u'_j e l'_j son as cotas superior e inferior, respectivamente, da variable x_j no nodo actual. Isto é, asígnase un peso maior a aquelas variables cuxos rangos se reduciron menos, respecto ó rango inicial.

5. **Densidade da variable.** $w(j, J) \equiv w(j)$, onde $w(j)$ é proporcional ó número total de monomios do problema nos que aparece a variable j. Deste xeito, reciben un peso maior as variables máis "activas", isto é, aquelas que máis se empregan para formular o problema.

Para asegurar a converxencia do algoritmo, cando se obteña $\theta_j = 0$ (ou moi próximo a cero), para todo $j \in N$, empregando unha regra de ramificación diferente á orixinal ou a "sum", iniciarase de novo este paso utilizando "sum" como regra de ramificación, nesa iteración.

3.2.7. Axuste das cotas

Unha técnica amplamente empregada no deseño de *solvers* globais é a chamada *bound tightening* ou axuste das cotas. Este procedemento consiste en axustar as cotas das variables co fin de reducir o dominio de busca do algoritmo. Isto permite obter formulacións máis axustadas das sucesivas relaxacións lineais, obtendo mellores cotas inferiores do valor óptimo de $(PP(\Omega))$ e, polo tanto, pechando máis rapidamente o *gap* de optimalidade.

Un dos métodos de *bound tightening*, estudado en González-Rodríguez et al. (2023), é o baseado na factibilidade (FBBT, do inglés *Feasibility-Based Bound Tightening*), que explota a relación das restricións que definen a rexión factible para reducir, cando sexa posible, o rango das variables. Nótese que este método é, en definitiva, un caso particular de cortes válidos, pois engade novas restricións de cota sen eliminar ningún punto factible de $(PP(\Omega))$. González-Rodríguez et al. (2023) tamén incorporan en `RAPOSa` o método baseado na optimalidade (OBBT, do inglés *Optimality-Based Bound Tightening*), que permite reducir as cotas das variables mediante a resolución dunhas certas relaxacións do problema orixinal con lixeiras variacións. Este método, a diferencia do anterior, pode eliminar puntos da rexión factible, despois de comprobar que segue existindo polo menos unha solución factible cuxo valor é óptimo. Ademais, o OBBT require un custo computacional maior fronte o FBBT, polo que é habitual empregalo só no nodo raíz. Para máis detalle, pode consultarsePuranik and Sahinidis (2017) ou Gómez-Casares et al. (2024), onde inclúen un estudo computacional pormenorizado do uso destas técnicas.

Capítulo 4

Variación no algoritmo RLT

A continuación presentaremos e estudaremos o rendemento dunha nova variación no algoritmo baseado na RLT, desenvolta no marco dunhas prácticas no CITMAga (*Galician Centre for Mathematical Research and Technology*), da man do equipo de desenvolvemento de `RAPOSa`. O código, as instancias e os resultados necesarios para reproducir a meirande parte das gráficas desta memoria están dispoñibles en https://github.com/ManuelAlvarezR/TFM.

4.1. Estudo teórico

Cando empregamos o algoritmo baseado na RLT para obter un óptimo global dun problema polinómico, a maior parte do tempo que ocupamos correspóndese coa resolución dos sucesivos problemas lineais, como probaremos computacionalmente na Subsección 4.1.2. Por iso, para obter unha solución global nun menor tempo, é importante tratar de reducir o tempo que adicamos na obtención das cotas inferiores. Atendendo a este principio, propoñemos deter o *solver* linear antes de chegar ó óptimo (da relaxación linear). Desta forma, esperamos obter unha cota inferior suficientemente boa, isto é, moi próxima ao valor óptimo real do problema lineal, reducindo notablemente o tempo empregado. Para poder desenvolver esta idea, que detallaremos na Subsección 4.1.2, sustentámonos nalgúns resultados de dualidade en optimización linear. Polo tanto, cómpre recapitular brevemente estes resultados. Así mesmo, introduciremos o método símplex dual, que se empregará para a resolucións dos problemas lineais.

4.1.1. Resultados teóricos previos

Para poder falar de dualidade e símplex dual, cómpre recordar algúns resultados da optimización linear e o método símplex. Dado que non son obxecto de estudo, non afondaremos moito nestes resultados. Non obstante, é importante comprender ben a teoría na que se fundamentan, polo que se remite ás referencias Bazaraa et al. (2010), Dantzig and Thapa (2003), Salazar-González (2001) e González-Díaz (2018), empregadas para escribir esta subsección.

Optimización Linear

No Capítulo 1 xa falabamos de problemas de optimización linear (Definición 1.15) e presentabamos a formulación habitual para un problema de optimización xeral (1.2). Non obstante, tendo en conta as propiedades lineais das funcións e as reformulacións que

prestabamos na páxina 2, un problema linear pode escribirse na forma estándar,

$$\begin{aligned} \text{minimizar} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{suxeito a} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{aligned} \tag{4.1}$$

onde $\boldsymbol{c} \in \mathbb{R}^n$ é o **vector de custos**, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ a **matriz de restricións** e $\boldsymbol{b} \in \mathbb{R}^m$ é o **vector de lados dereitos**, ou tamén na forma canónica

$$\begin{aligned} \text{minimizar} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{suxeito a} \quad & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned} \tag{P}$$

Notación 4.1. Denotaremos por $\boldsymbol{A}_i^f$ a fila i-ésima da matriz de restricións $\boldsymbol{A}$ e por $\boldsymbol{A}_j^c$ a columna j-ésima.

Proposición 4.2. *A rexión factible dun problema linear é un poliedro (convexo), acoutado ou non.*

Definición 4.3. Dado un conxunto convexo $S \subset \mathbb{R}^n$, un vector non nulo $\boldsymbol{d} \in \mathbb{R}^n$ é unha **dirección** de S se $\boldsymbol{x} + \lambda\boldsymbol{d} \in S$, para todo $\boldsymbol{x} \in S$ e $\lambda \geq 0$. Se S é un poliedro da forma $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}\}$, o conxunto de direccións pode representarse como

$$D = \{\boldsymbol{d} : \boldsymbol{A}\boldsymbol{d} \geq \boldsymbol{0}, \boldsymbol{1}\boldsymbol{d} = 1, \boldsymbol{d} \geq \boldsymbol{0}\}.$$

Definición 4.4. Sexa S un poliedro da forma $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$. Dise que $\boldsymbol{z} \in S$ é un **punto extremo** se non é posible representalo como combinación convexa de dous puntos distintos de S. Analogamente, $\boldsymbol{d} \in D$ é unha **dirección extrema** se non pode escribirse como combinación cónica de dúas direccións de S.

Definición 4.5. Dado un problema da forma (P), dise que unha restrición $\boldsymbol{A}_i^f\boldsymbol{x} \geq b_i$ está **saturada** no punto $\bar{\boldsymbol{x}} \in S$ se $\boldsymbol{A}_i^f\bar{\boldsymbol{x}} = b_i$

Proposición 4.6. *Sexa S un poliedro da forma $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$. Un punto $\bar{\boldsymbol{x}} \in S$ é un punto extremo se e só se en $\bar{\boldsymbol{x}}$ se saturan n restricións independentes das que definen o conxunto S.*

Demostración. Pode verse en González-Díaz (2018). □

Proposición 4.7. *Dado un problema de optimización linear con rexións factible $S \neq \emptyset$. Entón, tense que*

1. *Se o problema é acoutado, entón al menos un dos puntos extremos de S é óptimo.*
2. *O problema é non acoutado se e só se existe unha dirección extrema $\boldsymbol{d} \in S$ tal que $\boldsymbol{c}^T\boldsymbol{d} < 0$.*

Demostración. Pode verse en González-Díaz (2018). □

Pois ben, un dos métodos empregados para resolver problemas de optimización linear é o símplex. Este método baséase principalmente nos resultados anteriores aínda que ten, como veremos, moita relación coa dualidade. Do mesmo xeito ocorre co seguinte resultado, que presenta as condicións de optimalidade dun problema linear e establece a relación entre un problema e o seu dual, como veremos máis adiante.

Teorema 4.8. *(Condicións de optimalidade de Karush-Kuhn-Tucker) Dado un problema de programación linear da forma (P), $\bar{\boldsymbol{x}}$ é unha solución óptima de (P) se e só se existe $\bar{\boldsymbol{u}} = (\bar{\boldsymbol{w}}, \bar{\boldsymbol{v}}) \in \mathbb{R}^{m+n}$ tal que*

$$\boldsymbol{A}\bar{\boldsymbol{x}} \geq \bar{b}, \quad \bar{\boldsymbol{x}} \geq \boldsymbol{0}, \tag{FP}$$

$$\boldsymbol{A}^T\bar{\boldsymbol{w}} + \bar{\boldsymbol{v}} = \boldsymbol{c}, \quad \bar{\boldsymbol{w}} \geq \boldsymbol{0}, \bar{\boldsymbol{v}} \geq \boldsymbol{0}, \tag{FD}$$

$$\bar{\boldsymbol{w}}^T(\boldsymbol{A}\bar{\boldsymbol{x}} - \boldsymbol{b}) = 0, \quad \bar{\boldsymbol{v}}^T\bar{\boldsymbol{x}} = 0. \tag{FC}$$

Demostración. Pode verse en González-Díaz (2018). □

Polo de agora, estes resultados poden non parecer relevantes. Non obstante, cando introduzamos a dualidade en programación linear, entenderemos a súa gran importancia.

Dualidade

Dado un problema de optimización linear, que chamaremos **primal**, este pode asociarse con un novo problema, denominado **dual**. Este par de problemas, como veremos, teñen unha estreita relación. Dado o problema primal ($LP(\Omega)$), coa formulación canónica, o problema dual asociado é

$$\begin{aligned} \text{minimizar} \quad & \boldsymbol{b}^T\boldsymbol{w} \\ \text{suxeito a} \quad & \boldsymbol{A}^T\boldsymbol{w} \leq \boldsymbol{c} \\ & \boldsymbol{w} \geq \boldsymbol{0}, \end{aligned} \tag{D}$$

ou ben, empregando variables de folgura,

$$\begin{aligned} \text{minimizar} \quad & \boldsymbol{b}^T\boldsymbol{w} \\ \text{suxeito a} \quad & \boldsymbol{A}^T\boldsymbol{w} + \boldsymbol{v} = \boldsymbol{c} \\ & \boldsymbol{w} \geq \boldsymbol{0} \\ & \boldsymbol{v} \geq \boldsymbol{0}. \end{aligned} \tag{D}$$

Observación 4.9. Nótese que $\boldsymbol{b}^T\boldsymbol{w} = \boldsymbol{w}^T\boldsymbol{b}$ e $\boldsymbol{A}^T\boldsymbol{w} \leq \boldsymbol{c}$ se e só se $\boldsymbol{w}^T\boldsymbol{A} \leq \boldsymbol{c}^T$.

A forma na que se constrúa o dual depende da formulación escollida no problema primal. No Cadro 4.1, extraído de González-Díaz (2018), recóllense as relacións entre a formulación dos problemas primal e dual.

Unha vez presentada a formulación (D) dun problema dual, podemos comprender mellor as condicións de optimalidade de Karush-Kuhn-Tucker (Teorema 4.8). A condición (FP) non é nova, pois require a factibilidade da solución do primal. A condición (FD) correspóndese precisamente, coas restricións do problema (D), isto é, coa factibilidade da solución dual. Por último, a restrición de Folguras Complementarias (FC) establece unha relación entre as restricións dun problema coas variables do outro. Esta condición esixe que, cando unha restrición non está saturada no primal (ou no dual), a variable correspondente do problema dual (primal) é nula.

A continuación presentamos os principais resultados da teoría de dualidade, que nos permitirán entender a forte relación que existe entre ambos problemas.

Proposición 4.10. *Dados un problema primal (P) e o seu dual (D), tense que o dual de (D) coincide co problema primal (P).*

Proposición 4.11. *(Dualidade débil) Dados un par de problemas primal (P) e dual (D). Para calquera solucións factibles $\bar{\boldsymbol{x}}$ e $\bar{\boldsymbol{w}}$ de (P) e (D), respectivamente, tense que*

$$\bar{\boldsymbol{w}}^T\boldsymbol{b} \leq \boldsymbol{c}^T\bar{\boldsymbol{x}}.$$

Primal			**Dual**		
mín	$\boldsymbol{c}^\top \boldsymbol{x}$		máx	$\boldsymbol{w}^\top \boldsymbol{b}$	
suxeito a	$\boldsymbol{A}_i^{\mathrm{f}} \boldsymbol{x} \geq b_i,$	$i \in M_1$	suxeito a	$w_i \geq 0,$	$i \in M_1$
	$\boldsymbol{A}_i^{\mathrm{f}} \boldsymbol{x} \leq b_i,$	$i \in M_2$		$w_i \leq 0,$	$i \in M_2$
	$\boldsymbol{A}_i^{\mathrm{f}} \boldsymbol{x} = b_i,$	$i \in M_3$		w_i non restrinxido,	$i \in M_3$
	$\boldsymbol{x}_j \geq 0,$	$j \in N_1$		$\boldsymbol{w}^\top \boldsymbol{A}_j^{\mathrm{c}} \leq c_j,$	$j \in N_1$
	$\boldsymbol{x}_j \leq 0,$	$j \in N_2$		$\boldsymbol{w}^\top \boldsymbol{A}_j^{\mathrm{c}} \geq c_j,$	$j \in N_2$
	$\boldsymbol{x}_j$ non restrinxido,	$j \in N_3$		$\boldsymbol{w}^\top \boldsymbol{A}_j^{\mathrm{c}} = c_j,$	$j \in N_3$

Cadro 4.1: Relacións entre as formulacións dun problema primal e o seu dual

Demostración. Por ser $\bar{\boldsymbol{x}}$ factible do primal tense que $\bar{\boldsymbol{b}} \leq \boldsymbol{A}^T \bar{\boldsymbol{x}}$ e $\bar{\boldsymbol{w}} \geq 0$ por ser factible do dual, logo $\bar{\boldsymbol{w}}^T \boldsymbol{b} \leq \bar{\boldsymbol{w}}^T (\boldsymbol{A}^T \bar{\boldsymbol{x}})$. Pola factibilidade dual de $\bar{\boldsymbol{w}}$ tense que $\bar{\boldsymbol{w}}^T \boldsymbol{A} \leq \boldsymbol{c}$ e $\bar{\boldsymbol{x}} \geq 0$ por ser factible do primal, co cal $(\bar{\boldsymbol{w}}^T \boldsymbol{A}^T) \bar{\boldsymbol{x}} \leq \boldsymbol{c}^T \bar{\boldsymbol{x}}$. Polo tanto, xuntando desigualdades, temos que

$$\bar{\boldsymbol{w}}^T \boldsymbol{b} \leq \bar{\boldsymbol{w}}^T \boldsymbol{A}^T \bar{\boldsymbol{x}} \leq \boldsymbol{c}^T \bar{\boldsymbol{x}},$$

como queriamos probar. □

Corolario 4.12. *Se $\bar{\boldsymbol{w}}$ é unha solución factible do dual, entón $\boldsymbol{b}^T \bar{\boldsymbol{w}}$ é unha cota inferior do valor óptimo do primal. Analogamente, toda solución factible do primal $\bar{\boldsymbol{x}}$ xera unha cota superior $\boldsymbol{c}^T \bar{\boldsymbol{x}}$ do valor óptimo do dual.*

Corolario 4.13. *Se $\bar{\boldsymbol{x}}$ e $\bar{\boldsymbol{w}}$ son solucións factibles do primal e do dual tales que*

$$\boldsymbol{c}^T \boldsymbol{x} = \boldsymbol{b}^T \bar{\boldsymbol{w}},$$

entón $\bar{\boldsymbol{x}}$ e $\bar{\boldsymbol{w}}$ son solucións óptimas do primal e dual, respectivamente.

Corolario 4.14. *Dado un par de problemas (P) e (D), se un é non acoutado entón o outro é infactible.*

A Proposición 4.11, malia a súa sinxeleza, será fundamental para poder aplicar a variación desexada no algoritmo baseado na RLT, pois permitiranos obter cotas inferiores do problema linear e, polo tanto, do polinómico. Non obstante, compre presentar algún resultado máis antes de introducir o símplex dual.

Teorema 4.15. *(Dualidade forte) Dado un par de problemas (P) e (D) factibles. Entón, existen solucións óptimas $\bar{\boldsymbol{x}}$ e $\bar{\boldsymbol{w}}$ de (P) e (D), respectivamente, e son tales que os seus valores óptimos coinciden, isto é,*

$$\boldsymbol{c}^T \bar{\boldsymbol{x}} = \bar{\boldsymbol{w}}^T \boldsymbol{b}.$$

Demostración. Pode verse en Dantzig and Thapa (2003). □

Por último, presentamos o Teorema de Folguras Complementarias.

Teorema 4.16. *Dado un par de problemas (P) e (D) con solucións factibles* $\bar{\boldsymbol{x}}$ *e* $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{v}})$*, respectivamente. Entón* $\bar{\boldsymbol{x}}$ *e* $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{v}})$ *son solucións óptimas se e só se*

$$w_i(\boldsymbol{A}_i^f \boldsymbol{x} - b_i) = 0 \text{ para todo } i \in \{1, \dots, n\} \text{ e}$$
$$(c_j - \boldsymbol{w}^T \boldsymbol{A}_j^c) x_j = 0 \quad (v_j x_j = 0) \text{ para todo } j \in \{1, \dots, m\}.$$

Demostración. Pode verse en González-Díaz (2018). □

Este resultado presenta as tres condicións suficientes e necesarias de optimalidade, que son factibilidade primal, factibilidade dual e folguras complementarias. Esta última, como xa comentamos, esixe que toda variable asociada a unha restrición que está saturada sexa nula. É máis, o Corolario 4.13 dinos que podemos intercambiar a condición suficiente de folguras complementarias pola condición $\boldsymbol{c}^T \boldsymbol{x} = \boldsymbol{b}^T \bar{\boldsymbol{w}}$. Os métodos símplex e símplex dual, que presentamos a continuación, baséanse nestes resultados para atopar solucións óptimas dos problemas primal e dual.

Símplex

Dado un problema linear en forma estándar (4.1), tal que $\text{rango}(\boldsymbol{A}) = \text{rango}(\boldsymbol{A}, \boldsymbol{b}) = m$ e sexa B un conxunto de m índices distintos de $\{1, \dots, n\}$. Dise que B é unha **base** de (4.1) se $\boldsymbol{A}_B^c \equiv \boldsymbol{A}_B$ ten rango m. As variables $\boldsymbol{x}_B$ chámanse **variables básicas** e $\boldsymbol{x}_N$ son as **variables non básicas**, onde $N = \{1, \dots, n\} \backslash B$. Agora, podemos reescribir o sistema do problema primal (4.1) como

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}_B \boldsymbol{x}_B + \boldsymbol{A}_N \boldsymbol{x}_n = b. \tag{4.2}$$

En virtude da Proposición 4.6 e da Proposición 4.7, en calquera solución óptima do problema (4.1) se saturan n restricións. Posto que as m restricións de igualdade se saturan en toda solución factible, necesariamente $(n-m)$ restricións da forma $x_i \geq 0$ se teñen que saturar. Logo toda solución do sistema (4.2) será da forma

$$\boldsymbol{x}_B = \boldsymbol{A}_B^{-1} \boldsymbol{b} \text{ e } \boldsymbol{x}_N = \boldsymbol{0}.$$

Se ademais $\boldsymbol{x}_B \geq 0$, esta denomínase **solución factible** e a base B é **factible primal**. O método símplex comeza cunha base factible primal, obtén a solución básica factible $\boldsymbol{x}_B$ e calcula, para as $(n-m)$ variables non básicas, os custos reducidos $c_j - z_j$ onde

$$z_j \equiv \boldsymbol{c}_B^T \boldsymbol{y}^j \equiv \boldsymbol{w}_B^T \boldsymbol{A}_j^c := \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{A}_j^c.$$

Os custos reducidos $c_j - z_j$, $j \in N$, indícannos como cambiaría o valor da función obxectivo (por cada unidade que aumente x_j) se incluímos o índice j na base, isto é, se facemos a variable j básica. Se $c_j - z_j < 0$, para algún $j \in N$, interésanos introducir o índice k con menor custo reducido na base, pois reduciremos ó máximo posible (na etapa actual) o valor da función obxectivo. Neste caso, o vector $\boldsymbol{y}^k$ indícanos como teñen que cambiar os valores das variables básicas para facer este cambio mantendo a factibilidade primal. Se $\boldsymbol{y}^k \leq \boldsymbol{0}$, podemos aumentar a variable x_k tanto como nós queiramos, polo que o problema será non acoutado. Noutro caso, a variable x_i que primeiro se anule ao aumentar o valor de x_k será a que se converta en non básica. Isto é, intercambiamos o índice

$$i = \text{argmin} \left\{ \frac{(\boldsymbol{x}_B)_l}{y_l^k} : y_l^k > 0 \right\}$$

polo índice k na base B. Actualízase $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ e os custos reducidos $c_j - z_j$, $j \in N$. Cando os custos reducidos sexan $c_j - z_j \geq 0$, para toda variable non básica $j \in N$, a solución factible $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ será óptima. Inclúese o método descrito no Algoritmo 1.

Observación 4.17. Nótese que non se precisa o cálculo da solución inicial factible. Para obtela existen diferentes métodos como o da M-grande ou o das dúas fases. Posto que non imos entrar en máis detalles, pois non é necesario para esta memoria, déixase como referencia para estes métodos o Capítulo 4 do libro Bazaraa et al. (2010).

Algoritmo 1 Algoritmo Símplex

INICIALIZACIÓN
Parte dunha base primal factible B con solución básica factible asociada $(\boldsymbol{x}_B, \boldsymbol{x}_N)$, onde $\boldsymbol{x}_B = \boldsymbol{A}_B^{-1}b$ e $\boldsymbol{x}_N = \boldsymbol{0}$, e con custos reducidos $c_j - z_j$, onde

$$z_j = \boldsymbol{w}_B^T \boldsymbol{A}_j^c = \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{A}_j^c,$$

para todo $j \in N$.
PASO 1: Criterio de entrada.
Sexa $k = \operatorname{argmin}\{c_j - z_j : j \in N\}$.
if $c_k - z_k \geq 0$ **then**
  B é óptima, FIN.
else
  Continuar ó PASO 2
end if
PASO 2: Criterio de saída.
Calculamos $\boldsymbol{y}^k = \boldsymbol{A}_B^{-1}\boldsymbol{A}_k^c$
if $\boldsymbol{y}^k \leq \boldsymbol{0}$ **then**
  O problema (P) é non acoutado, FIN.
else
  Sexa

$$i = \operatorname{argmin}\left\{\frac{(\boldsymbol{x}_B)_l}{y_l^k} : y_l^k > 0\right\}.$$

  Continuamos ó PASO 3.
end if
PASO 3: Actualización.
Actualizar B (intercambiando o elemento i por k), $\boldsymbol{x}_B$ (usando $\boldsymbol{y}^k$) e $c_j - z_j$, $j \in N$.
Volver ó PASO 1.

Proposición 4.18. *Dado un problema lineal da forma (4.1) e sexa $\bar{\boldsymbol{x}}$ unha solución óptima con base asociada B, entón*

$$\bar{\boldsymbol{w}} := \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1}$$

é unha solución óptima do problema dual.

Demostración. Para cada $j \in N$, temos que $\bar{\boldsymbol{w}}\boldsymbol{A}_j^c = z_j$, sendo $\bar{\boldsymbol{w}} := \boldsymbol{c}_B^T\boldsymbol{A}_B^{-1}$. Pola condición de optimalidade do primal sabemos que $z_j - c_j \leq 0$, ou o que é o mesmo, $\bar{\boldsymbol{w}}\boldsymbol{A}_j^c = z_j \leq c_j$, que é precisamente a condición de factibilidade do dual. Por outro lado,

$$\boldsymbol{c}^T\bar{\boldsymbol{x}} = \boldsymbol{c}_B^T\bar{\boldsymbol{x}}_B + \boldsymbol{c}_N^T\bar{\boldsymbol{x}}_N = \boldsymbol{c}_B^T\boldsymbol{A}_B^{-1}\boldsymbol{b} = \bar{\boldsymbol{w}}^T\boldsymbol{b},$$

logo, en virtude do Corolario 4.13, tense que $\bar{\boldsymbol{w}}$ é unha solución óptima do problema dual. □

Noutras palabras, o método símplex parte dunha solución factible primal e constrúe unha solución dual $\boldsymbol{w}$ que satisfai a condición $\boldsymbol{c}_B^T\boldsymbol{x} = \boldsymbol{w}^T\boldsymbol{b}$. Continúa iterando, mellorando

o valor da función obxectivo, ata que a solución dual sexa factible, isto é, $c_j - \boldsymbol{w}_b^T \boldsymbol{A}_j^c \geq 0$. Nese momento, ambas solucións serán óptimas.

Símplex Dual

O símplex dual, en cambio, parte dunha base B que sexa **factible dual**, isto é, tal que $c_j - z_j = c_j - \boldsymbol{w}_b^T \boldsymbol{A}_j^c \geq 0$, para todo $j \in N$, onde $\boldsymbol{w}_b := \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1}$. Deste xeito, obtén unha solución dual factible e unha solución primal cumprindo a igualdade $\boldsymbol{c}_B^T \boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{b}$. Se ademais $\boldsymbol{x}_B \geq 0$, a solución $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ é factible. Entón, en virtude do Corolario 4.13, chegamos ó óptimo e rematamos. En caso contrario, existe polo menos unha variable básica $x_l < 0$, $l \in B$. Neste caso, escollemos aquela que maximice a violación de non negatividade, isto é, seleccionamos o índice

$$i = \operatorname{argmin}\{x_l : l \in B\},$$

que sae da base. No seu lugar, entra o índice $j \in N$ cuxo custo reducido se anule antes ao incrementar o valor da variable non básica x_k. Se podemos aumentar tanto como queiramos o valor desta variable, sen perder a factibilidade das variables duais, o problema dual é non acoutado e rematamos.

Polo tanto, primeiro probamos que o método símplex se move de solución factible en solución factible do primal, construíndo en cada iteración unha solución do dual baixo a condición $\boldsymbol{c}_B^T \bar{\boldsymbol{x}} = \bar{\boldsymbol{w}}^T \boldsymbol{b}$. No caso do símplex dual vimos que, en cada iteración, o algoritmo dispón dunha solución primal (que non é factible, pero si satisfai a condición anterior) e da solución dual factible asociada. No primeiro método, a condición de optimalidade é $c_j - z_j \geq 0$, isto é, factibilidade dual. Pola contra, o segundo ten como condición de optimalidade a factibilidade primal, $\boldsymbol{x}_B \geq 0$. Ademais, mentres o método símplex vai reducindo o valor da función obxectivo de (P) ata chegar o óptimo, o símplex dual vai incrementando o valor da función obxectivo de (D). Polo tanto, cando detemos o algoritmo dual antes de rematar, obtemos unha cota inferior do valor óptimo.

4.1.2. Variación no algoritmo RLT

Como xa comentabamos ó inicio desta subsección, co fin de reducir o tempo empregado para a resolución das sucesivas relaxacións lineais, propoñemos unha variación no algoritmo baseado na RLT. A idea é sinxela, despois de resolver o nodo raíz, resolvemos as seguintes relaxacións lineais detendo o algoritmo do símplex dual linear antes de chegar ó óptimo.

Obxectivo

Cando resolvemos un problema linear co método símplex ou o símplex dual, ao rematar o algoritmo o valor da función obxectivo é óptimo. Non obstante, nas últimas iteracións do método, o valor da función obxectivo está moi preto de ser óptimo. De feito, pode ocorrer que este coincida co valor óptimo, aínda que non se acaden as condicións de optimalidade, isto é, a solución non sexa factible (primal ou dual). Pois ben, fixando un límite de iteracións no algoritmo do método de resolución linear o que se pretende é explotar este comportamento do algoritmo para reducir o tempo empregado na resolución das relaxacións lineais, coa desvantaxe de empeorar lixeiramente a cota inferior obtida. Outra posibilidade, limitar o número máximo de iteracións de maneira moi agresiva. Deste xeito é posible facer unha ramificación inicial moi rápida, con unhas cotas inferiores claramente peores, pero explorando moitos nodos en pouco tempo. En calquera caso, é esperable que

o número de nodos aumente, malia iso, esperamos atopar un equilibrio axeitado entre este feito e unha diminución do tempo de resolución ou do *gap* de optimalidade, fixado un tempo máximo de resolución.

Elección do método para a resolucións dos LPs

Para poder empregar correctamente o algoritmo RLT detendo nalgúns nodos a resolución do problema linear antes de chegar á solución óptima, necesitamos obter, no defecto do valor óptimo, unha cota inferior deste. Por este motivo, o método símplex non é válido pois, como xa vimos, vai reducindo o valor da función obxectivo ata chegar ó óptimo. Polo tanto, proporciona cotas superiorese do valor óptimo do problema linear, non inferiores, logo non serve para reducir o *gap* de optimalidade do problema polinómico.

Pola contra, os resultados vistos na subsección anterior asegúrannos que, nunha iteración intermedia do símplex dual, este método proporciónanos unha cota inferior do valor óptimo do problema lineal (e polo tanto do polinómico). Esto permítenos seguir pechando o *gap* de optimalidade. Como inconveniente, a solución primal que obtemos viola as condicións de factibilidade, o que provocará, como veremos, algúns obstáculos na converxencia do algoritmo baseado na RLT.

Parámetros necesarios

Se detemos o algoritmo do símplex dual nunha iteración intermedia en tódolos nodos da árbore de ramificación, nunca chegaremos ó óptimo dos subproblemas lineais, e polo tanto tampouco de $(PP(\Omega))$. Entón, o que faremos será, unha vez resolto o nodo raíz, aplicar esta variación sobre un número de iteracións do algoritmo RLT previamente fixado. Por exemplo, podemos establecer que despois das 10 primeiras iteracións a modificación deixe de estar activa, isto é, se resolva de novo co algoritmo RLT orixinal. A primeira vista, non é sinxelo decidir cal é o número adecuado de iteracións nas que debe resolverse sen chegar ó óptimo, ou se un mesmo número de iteracións é adecuado para distintos problemas. Por exemplo, se aplicamos esta variación nun número pequeno de iteracións, a redución do tempo esperada pode non ser significativa. Pola contra, fixar un número moi grande pode facer que problemas que se resolvían nas primeiras iteracións, agora non poidan facelo. No estudo computacional, empregaremos diferentes procedementos para fixar a cantidade de iteracións nas que estará activa esta modificación.

Por outro lado, tamén debemos establecer o número máximo de iteracións que pode empregar o método símplex dual para resolver as relaxacións lineais. Do mesmo xeito que ocorría no caso anterior, tampouco está claro cal é a cantidade ideal, pois non coñecemos cal é o número de iteracións que require cada nodo para chegar ó óptimo. Aínda coñecendo esta cifra, podemos decidir se aplicamos unha modificación máis ou menos agresiva. Estudaremos diferentes opcións no estudo computacional.

Dificultades derivadas da variación no algoritmo RLT

Como xa adiantabamos, obter unha solución infactible cuxo valor asociado non é óptimo supón unha serie de dificultades engadidas ó empregar o algoritmo de ramificación espacial e acoutamento, estudado na Subsección 2.3. que debemos analizar e resolver.

Supoñamos que nos atopamos na Etapa 1 do algoritmo RLT nun nodo cuxa relaxación linear se resolveu detendo o algoritmo do símplex dual antes de chegar ó óptimo. Nesta

etapa atopámonos co primeiro obstáculo, pois os lemas previos ó Teorema de converxencia non teñen porque ser válidos. En particular, pode ocorrer que na solución obtida o valor da variable escollida para ramificar non está no seu rango actual, isto é,

$$x_p^{k,t} \notin (l_p^{k,t}, u_p^{k,t}).$$

O que facemos neste caso, sen comprometer ningún aspecto teórico, é ramificar sobre o punto medio do intervalo. Outra opción é escoller a seguinte variable ramificadora, que maximice as violacións RLT, e se atope dentro do seu rango. Non obstante, pode ocorrer que moitas ou tódalas variables non cumpran esta condición, o que suporía un custo computacional maior, polo que descartamos esta opción.

Supoñamos agora que resolvemos na Etapa 2 os subproblemas lineais coa modificación activa, isto é, detendo o símplex dual antes de cumplirse as condicións de optimalidade. Ao empregar o símplex dual, o problema dual pode non ser acoutado, en cuxo caso o Corolario 4.12 implica que o problema linear é infactible. Se o problema dual é infactible, pode ocorrer que o problema linear sexa infactible ou ben que o problema sexa non acoutado. Non obstante, posto que o problema do nodo raíz está acoutado, os subproblemas tamén. Escollemos entón a variables sobre a que ramificaremos. Se $\theta_p^{k,t_1} = 0$, a solución $\boldsymbol{x}^{k,t_1}$ satisfai as igualdades RLT mais non é factible de $(PP(\Omega))$, pois non o é da relaxación linear. Esto supón un grave inconveniente pois, no caso de ser $LB_{k,t_1} < v^*$, non poderemos actualizar a cota superior do valor óptimo nin a solución actual. De feito, non é posible pechar a rama, é dicir, non podemos actualizar o conxunto $Q_k = Q_k \backslash \{t_1\}$. A solución para poder seguir explorando esta espazo da rexión factible, coa modificación activa, é ramificar atendendo tamén as infactibilidades do problema linear.

Na Etapa 3, esta variación do algoritmo non interfire coa poda das ramas, isto é, ao actualizar o conxunto $Q_{k+1} = Q_k \backslash \{t \in Q_k : LB_{k,t} \geq v^*\}$. Os resultados vistos ao comezo deste capítulo, garántennos que o LB obtido coa nosa modificación nunca é maior ó valor óptimo da relaxación linear. Polo tanto, non pode ocorrer que podemos unha rama que non debía ser podada.

Na Etapa 4, posto que o $LB_{k,t}$ obtido coa variación é xeralmente menor estrito que o resultante do algoritmo orixinal, é posible que alteremos lixeiramente o criterio de selección do nodo a explorar. Pois, como xa comentamos, a cota inferior actual dun nodo, obtida coa modificación activa, pode ser menor á cota inferior real. Non obstante, explorar unha rama cunha cota menor que outra, non significa que sexa máis probable atopar a solución óptima nesta, senón que hai un maior marxe de mellora, polo que mantemos o mesmo criterio.

4.2. Estudo Computacional

4.2.1. Implementación e entorno da proba

Todas as execucións que se recollen nesta memoria, a excepción das presentados no apartado 4.2.4, foron realizadas no supercomputador Finisterrae II, proporcionado polo Centro de Supercomputación de Galicia (CESGA). En concreto, utilizamos nodos computacionais alimentados con 2 CPUs Intel Haswell 2680v3 deca-core con 128 GB de RAM conectados a través dunha rede Infiniband FDR e con 1 TB de disco duro. As variacións sobre o algoritmo baseado na RLT foron implementadas en C++ con `RAPOSa`.

As instancias do apartado 4.2.4 resolvéronse mediante o *solver* `Gurobi` 10.0.2 na interface AMPL (Fourer et al., 1990). O código foi executado nunha CPU Intel i7-7500 de dous núcleos con 8 GB de RAM.

En canto ás instancias empregadas para o estudo computacional, utilizamos tres conxuntos de problemas diferentes. O primeiro, denotado como DS, está tomado de Dalkiran and Sherali (2016) e consta de 180 problemas de programación polinómica xerados aleatoriamente de diferentes graos, número de variables e densidade. O segundo conxunto test recolleuse da coñecida librería de *benchmark*, MINLPLib (Bussieck et al., 2003), seleccionando aquelas instancias que son problemas de programación polinómica con variables continuas, resultando un total de 168. Nótese que estes problemas son especialmente interesantes pois trátase de modelos aplicados no mundo real. O último conxunto, QPLib, recóllese da librería Furini et al. (2019), onde se inclúen problemas QPQC con variables mixtas enteiras. Estas instancias proveñen de estudos teóricos e de casos prácticos reais. En tódolos casos, trátase de problemas de gran dificultade. Para esta memoria só nos interesan os problemas polinómicos de grao dous con variables continuas, polo que empregamos un subconxunto desta colección de 63 instancias.

O *solver* linear empregado é `Gurobi` (Gurobi Optimization, LLC, 2023) e o non linear local `Ipopt` (Wächter and Biegler, 2006). En `RAPOSa`, estableceuse un límite de tempo de resolución de 10 minutos nas comparacións entre diferentes configuracións.

4.2.2. Configuración estándar de RAPOSa

Para facer as probas computacionais da variación no algoritmo baseado na RLT fíxose unha implementación en C++ no código de RAPOSa. Cabe destacar que non empregamos tódalas melloras do algoritmo baseado na RLT, mencionadas na Sección 3.2. Hai opcións que, como xa se comentou, non supoñen unha mellora no rendemento do algoritmo, como o *warm start*, polo que, en principio, non se empregan. Outras, como as opcións *bound tightening* e os cortes SDP, que si poden mellorar o rendemento do algoritmo, non se inclúen inicialmente para non interferir co estudo específico da variación que implementamos. Polo mesmo motivo, limitarase a un o número de núcleos que pode empregar o *solver* linear. Por outro lado, o criterio de ramificación por defecto en `RAPOSa` emprega os valores duais. Non obstante, salvo nos primeiros resultados da Subsección 4.2.5, utilizamos o criterio de ramificación "sum", en lugar dos valores duais, pois ao empregar o símplex dual este último criterio pode verse afectado. Isto non resulta prexudicial para o rendemento do algoritmo, pois demostrouse computacionalmente que o criterio de ramificación "sum" ten un comportamento moi similar ó dual ou ó do rango. Por último o *presolve*, aínda que é beneficioso para o rendemento na configuración orixinal, por cuestións técnicas da propia implementación de `Gurobi`, non pode estar activa se queremos obter a solución do símplex dual fixando un límite de iteracións.

4.2.3. Tempos LP na configuración estándar

Cando resolvemos un problema polinómico en `RAPOSa`, recóllense datos que serven para estudar o comportamento do algoritmo, como o tempo total de resolución, o número de iteracións, o gap relativo, etc. En particular, mídense os tempos relacionados coa resolución dos problemas lineais, que se recollen nas seguintes variables:

- **Tempo linear**. Tempo total de interacción de `RAPOSa` co *solver* linear. Correspóndese co tempo que o *solver* linear require para crear o modelo, resolver o problema e

procesar a solución.

- **Tempo linear de xeración**. Tempo total da xeración dos problemas lineais, isto é, tempo empregado polo *solver* linear para cargar o modelo. (Non confundir co tempo que require a técnica RLT para xerar a linearización.)
- **Tempo linear de resolución**. Tempo total da resolución dos problemas lineais.
- **Tempo linear de resolución obtido polo *solver***. Tempo total da resolución dos problemas lineais, medido polo propio *solver*.
- **Tempo linear de posprocesado**. Tempo total empregado para procesar a solución posterior á resolución dos problemas lineais.

Para estudar como se distribúen os tempos relacionados coas relaxacións lineais, escollemos aqueles problemas que non se resolveron no tempo fixado, isto é, que tardan máis de 10 minutos en obter unha solución óptima. Ademais, seleccionamos aquelas instancias que requiren máis de 50 iteracións de `RAPOSa`. Estes problemas poden ser máis interesantes de cara a un estudo computacional, pois canto maior sexa o número de iteracións, máis patente poderá ser o comportamento desta modificación do algoritmo de ramificación espacial e acoutamento. O número de problemas que cumpren estas condicións son 138.

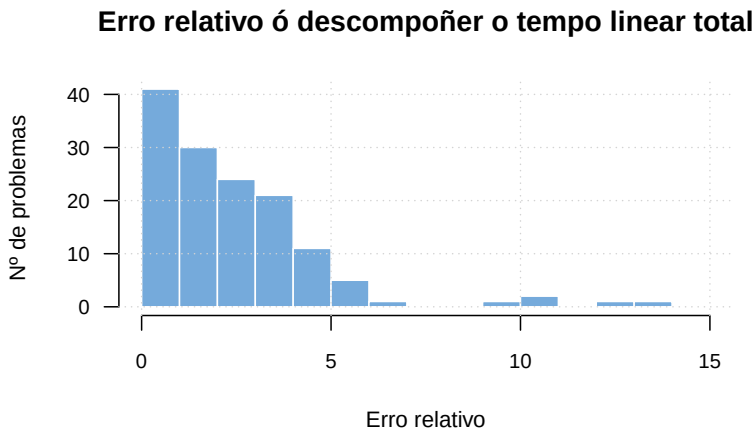


Figura 4.1: Histograma do erro relativo, respecto ó tempo linear total, ao medir os tempos dos subprocesos de xeración, resolución e posprocesado, que xuntos conforman o tempo linear total.

Por cada problema resolto con `RAPOSa`, obtemos un arquivo en formato JSON onde se recollen tódolos datos. Con axuda do software estatístico R (R Core Team, 2021), onde realizamos tódolos cálculos e gráficas presentes nesta subsección, importamos, filtramos e seleccionamos os datos desexados. Comezamos agora comprobando que o tempo linear total se recolla nos tempos de xeración, resolución e posprocesado, isto é, que non exista outro proceso que requira un tempo significativo e non esteamos a medir. Para elo, calculamos a diferencia relativa, respecto ó tempo linear total, entre este tempo e o empregado polos tres subprocesos. Observamos no histograma da Figura 4.1 que os erros son, en xeral, menores do 5 % respecto ó tempo linear total. De feito, só nun 5 % dos problemas, o error relativo e maior do 5.6 %. Polo tanto, non existe outro subproceso significativo que

debamos ter en conta.

Tamén comprobamos co histograma da Figura 4.2 que o tempo linear de resolución calculado por nós e o devolto por `Gurobi` son case idénticos.

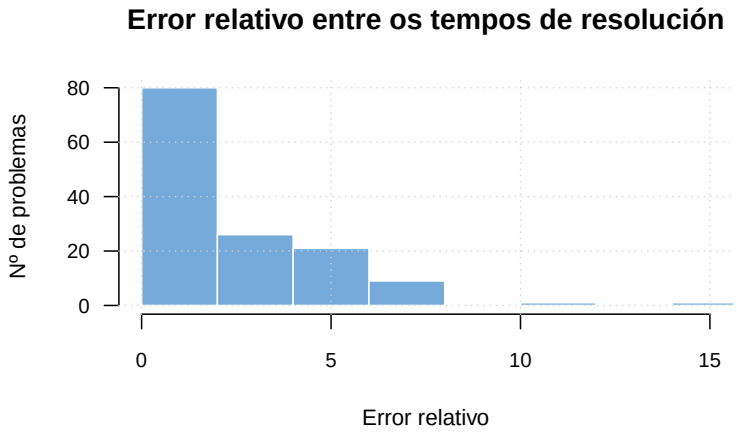


Figura 4.2: Histograma do erro relativo entre o tempo linear de resolución devolto por `Gurobi` fronte o obtido na implementación de `RAPOSa`.

Agora podemos estudar como de importante é a proporción do tempo total que adicamos as relaxacións lineais e, en particular, a súa resolución. Para elo, incluímos os histogramas das Figura 4.4 e 4.3. Na segunda figura observamos que, na gran maioría das instancias, o tempo empregado expresamente para a resolución das relaxación lineais é máis da metade do tempo total. Esto confirma o gran potencial desta modificación do algoritmo, coa que tratamos de reducir o tempo linear de resolución.

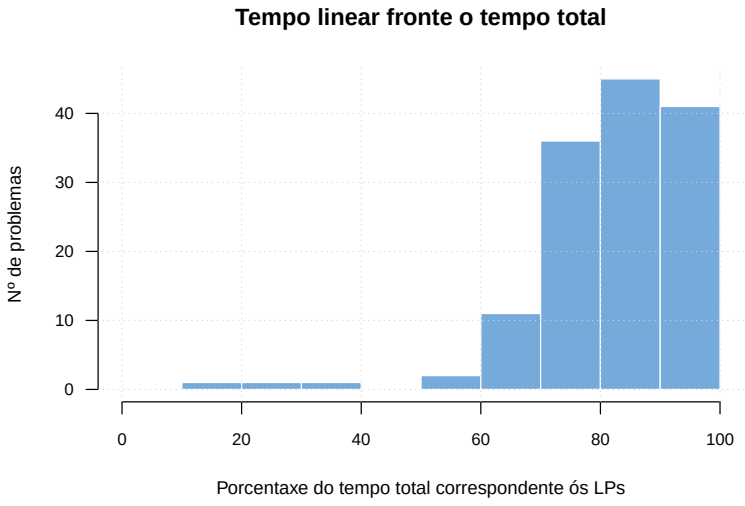


Figura 4.3: Histograma da porcentaxe que representa o tempo linear fronte o total.

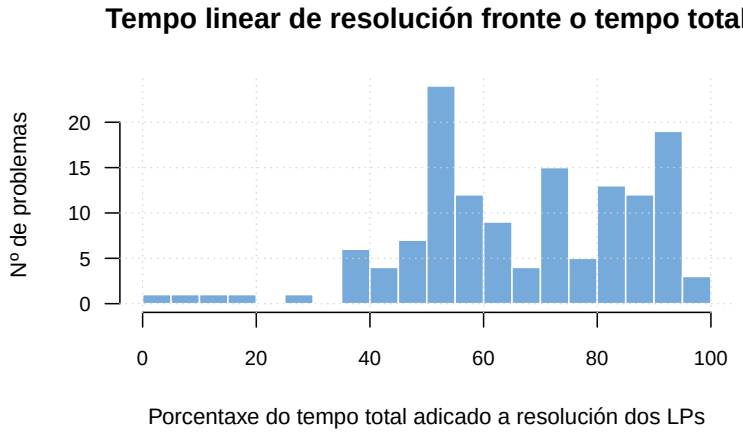


Figura 4.4: Histograma da porcentaxe que representa o tempo linear de resolución fronte o tempo total.

4.2.4. Probas preliminares

Nesta subsección incluímos unhas probas preliminares e máis rudimentarias, sen empregar o entorno de `RAPOSa`, estudando o comportamento que ten a nosa variación aplicada a un único nodo, neste caso o nodo raíz. Escollemos un conxunto de problemas polinómicos inicial e resolvemos a relaxación linear do nodo raíz correspondente dende AMPL con `Gurobi`. Primeiro, facémolo ata chegar a optimalidade, gardando o número de iteracións do símplex dual empregadas en cada problema. Despois, resolvemos de forma iterativa fixando como número máximo de iteración ó $10, 20, 30, \ldots, 90\,\%$ das iteracións totais. Pretendemos estudar dúas cousas, como se reducen os tempos de resolución nas diferentes configuracións e como avanza o valor da función obxectivo a medida que itera o símplex dual.

Tempo de resolución no nodo raíz

Escollemos, de entre os problemas da Subsección 4.2.3, aqueles que teñen un tempo medio por nodo maior a 1 segundo. Pois doutro xeito, os tempos serían tan pequenos que as gráficas resultarían moi imprecisas. Para cada problema, resolvemos coa versión estándar, isto é, resolvendo co símplex dual ata cumprir os criterios de optimalidade, e coas novas configuracións, parando no $10, 20, 30, \ldots, 90\,\%$ das iteracións respecto á versión estándar. Na Figura 4.5 representamos, para cada configuración, un *box-plot* onde se mide a porcentaxe do tempo total empregado. Con esta gráfica observamos que a porcentaxe do tempo total empregado en cada configuración coincide coa porcentaxe das iteración totais establecida. Polo tanto, cando paramos a resolución dun problema na metade das execucións do símplex dual, por exemplo, reducimos un $50\,\%$ o tempo empregado. Loxicamente, estas conclusión non se poden xeneralizar en termos do tempo linear total, posto que se asume que o número de nodos aumenta. Si son asumibles en termos do tempo linear de resolución medio por nodo. En calquera caso, como estudo preliminar, é un resultado moi prometedor.

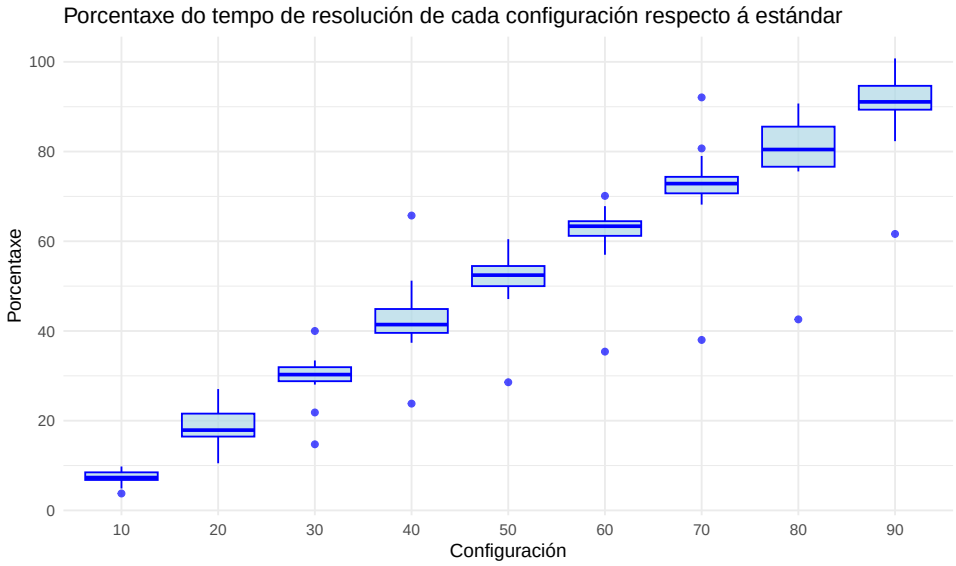


Figura 4.5: Porcentaxe do tempo empregado para a resolución de cada configuración respecto á versión estándar.

Evolución do valor da función obxectivo

Imos estudar agora a evolución da función obxectivo a medida que itera o símplex dual. Escollemos un problema calquera, e resolvemos a relaxación do nodo raíz seguindo o mesmo procedemento que no apartado anterior. Pode ocorrer que o valor da función obxectivo creza moito nas primeiras iteracións do símplex dual, aproximándose ó valor óptimo máis lentamente nas últimas iteracións. Entón, parando moito antes de acadar o valor óptimo, teremos unha moi boa cota inferior reducindo notablemente o tempo. Pola contra, se é nas últimas iteracións do símplex dual cando o algoritmo é capaz de crecer ata o óptimo, a variacións que presentamos non tería sentido. Pois de pouco serve reducir o tempo empregado na resolución da relaxación linear, se o valor da función obxectivo está moi afastado do óptimo e non pechamos o *gap* de optimalidade do problema polinómico. Na figura 4.6 incluímos a evolución do valor da función obxectivo a medida que itera o símplex dual en catro problemas distintos. Para ilustrar as diferentes situacións que poden ocorrer, escollemos catro instancias con comportamentos ben diferenciados. Na Figura 4.6a, o valor da función obxectivo crece de maneira linear, o que poderiamos denominar un comportamento normal. Un resultado desfavorecedor, no que o valor da función obxectivo crece moi rápido nas últimas iteracións, é o que se observa na Figura 4.6b. Pola contra, un comportamento ideal é o incluído na Figura 4.6d, onde despois das primeiras iteracións o valor da función obxectivo é moi próximo ó óptimo. Un caso máis atípico, pero positivo de cara ó noso estudo, é o representado na Figura 4.6c, onde o valor da función obxectivo na solución inicial é óptimo (aínda que a solución asociada non sexa factible do primal).

Para poder comparar a evolución de varios problemas nunha mesma gráfica, calculamos o *gap* relativo entre o valor óptimo e o valor da función obxectivo na iteración correspondente como

$$\frac{|v^*_{100} - v^*_p|}{\text{máx}\{|v^*_{100} - v^*_{10}|, 1\}},$$

onde v^*_p é o valor da función obxectivo obtido ao fixar como límite de iteracións do símplex

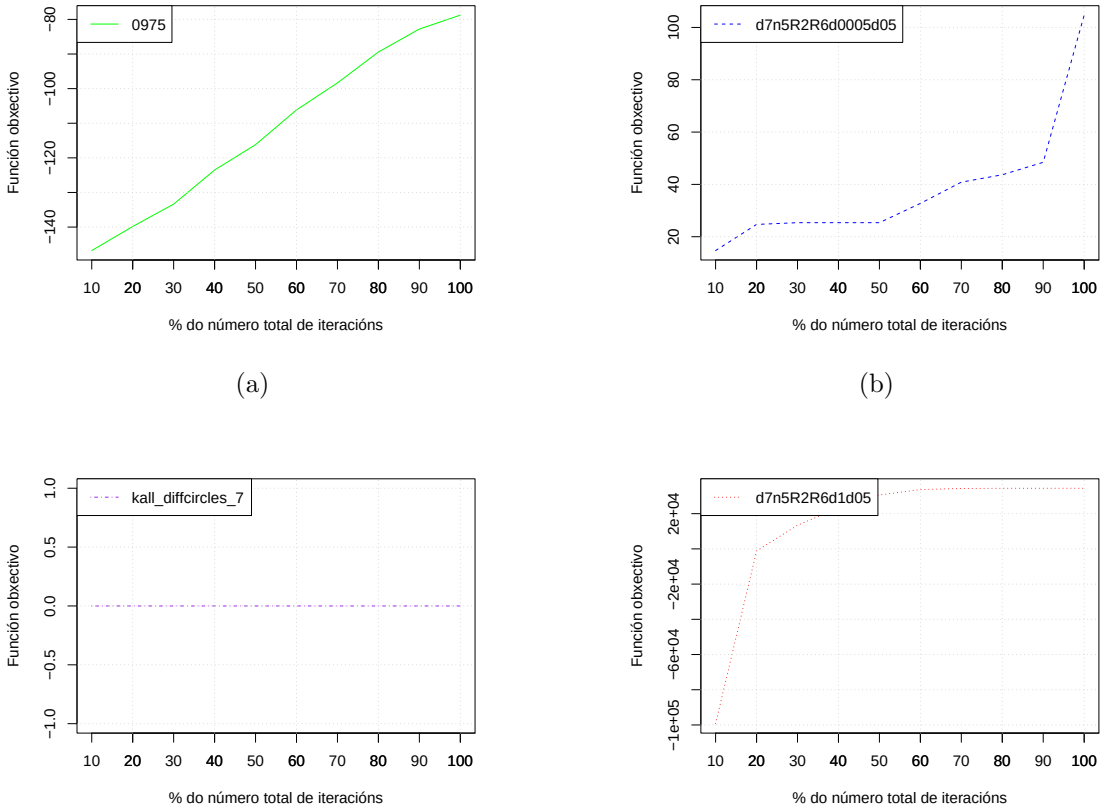


Figura 4.6: Evolución do valor da función obxectivo a medida que itera o símplex dual.

dual o $p\,\%$ das iteracións necesarias para cumprir as condicións de optimalidade. É dicir v^*_{100} é o valor óptimo, mentres que v^*_{10} é o valor da función obxectivo cando o símplex dual leva un $10\,\%$ das iteracións totais necesarias. Desta forma, medimos como se pecha o *gap* entre o valor da función obxectivo na primeira parada e o óptimo, isto é, como evoluciona o valor da función obxectivo. Para calquera problema tense que executadas as primeiras $10\,\%$ iteracións do símplex dual, o *gap* relativo será do $100\,\%$ e ao rematar do $0\,\%$. Nótese que, cando a diferencia na primeira parada é moi próxima a cero, defínese o *gap* absoluto para evitar problemas numéricos. Incluímos na Figura 4.7 a evolución do *gap* nos catro problemas anteriores. Vemos que, con esta representación, somos quen de extraer as mesmas conclusións que na Figura 4.6 pero incluíndo todos os problemas nunha mesma gráfica.

Repetimos este procedemento para un conxunto de máis de 100 problemas. Posto que o gráfico do *gap* relativo con tantos problemas pode resultar algo confuso, incluímos na Figura 4.8 un gráfico semellante empregando un *box-plot* asociado a cada porcentaxe das iteracións totais. Como vemos, a tendencia xeral é que o símplex dual peche o *gap* máis rapidamente na primeira metade das iteracións. De feito, no último $20\,\%$ das iteración o valor da función obxectivo é case óptimo na ampla maioría das instancias.

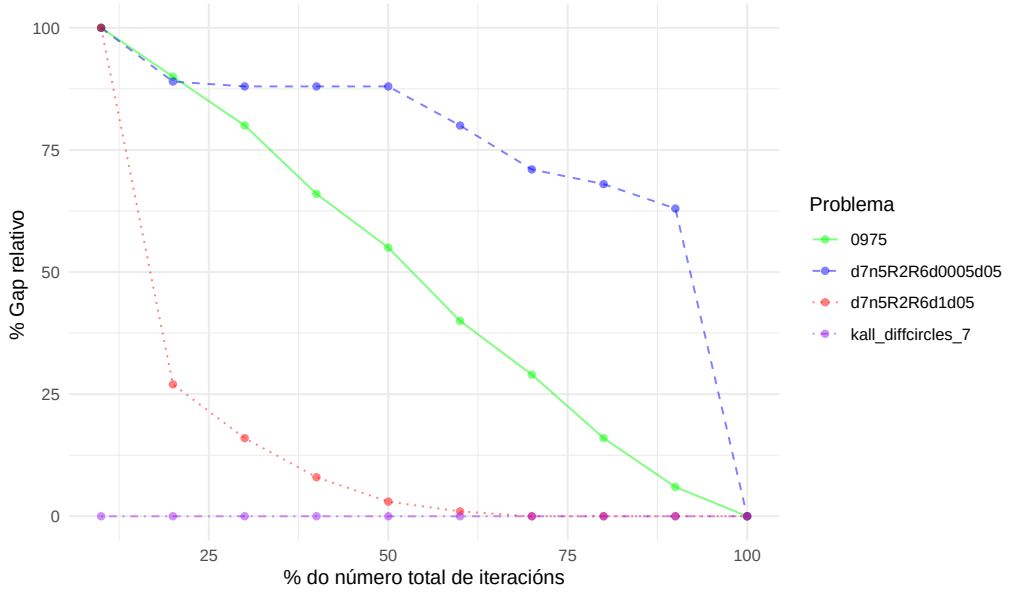


Figura 4.7: Evolución do *gap* relativo entre o valor óptimo e o valor da función obxectivo ao parar por límite de iteracións.

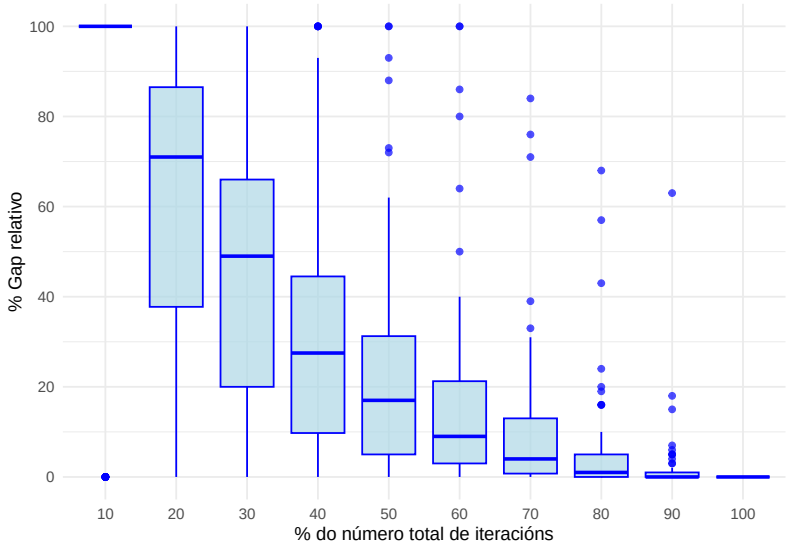


Figura 4.8: Evolución do *gap* relativo en problemas da librería DS segundo a súa densidade.

Para estudar se este comportamento depende do grao de dificultade do problema, incluímos na Figura 4.9, a evolución do *gap* relativo en varios problemas da librería DS, diferencia por cores segundo a densidade dos seus polinomios. A partir da metade das iteracións, podemos comprobar que aqueles problemas especialmente densos (1 e 0.5) teñen un *gap* relativo máis pequeno. É dicir, en problemas complexos a evolución da función obxectivo ao iterar o método símplex parece máis favorable cara ó noso estudo. De feito, esta conclusión coincide co comportamento dos dous exemplos incluídos nas Figuras 4.6 e 4.7.

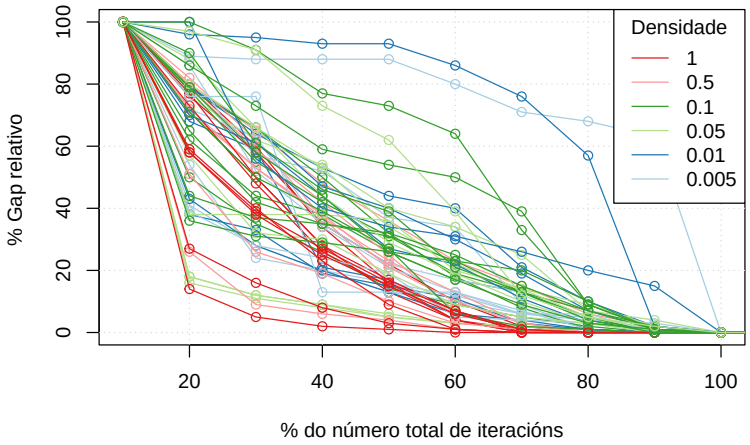


Figura 4.9: Evolución do *gap* relativo entre o valor óptimo e o valor da función obxectivo ao parar por límite de iteracións, coloreado segundo a densidade dos polinomios do problema correspondente.

4.2.5. Análise computacional das diferentes configuracións

Con todos os análises anteriores, temos argumentos suficientes para implementar e estudar a variación sobre o algoritmo de ramificación espacial e acoutamento, mediante o *solver* `RAPOSa`. Non obstante, nestas probas iniciais coñecíamos previamente o número de iteracións que requiría o símplex dual en cada problema linear para chegar á solución óptima. Posto que na realidade isto non é así, o que facemos é resolver inicialmente o nodo raíz ata óptimalidade. Despois, resolvemos fixando como límite de iteración unha porcentaxe determinada das empregadas no nodo raíz, durante unha cantidade de iteracións de `RAPOSa` previamente establecida. Tendo en conta que un problema linear é moi semellante ós subproblemas seguintes, o comportamento da variación do algoritmo aplicando esta técnica non debería variar do ilustrado na subsección anterior. Ademais, como xa comentamos, o número de iteracións de `RAPOSa` na que a modificación está activa pode ser moi relativa entre diferentes problemas. Por iso, tamén faremos probas fixando unha cantidade variable, como o número de variables do problema. Así, problemas con menos variables, que en principio son máis sinxelos e requiren menos iteracións, terán activa esta variación en menos. Deste xeito, evitando algúns casos nos que, instancias nas que a converxencia se produce nas primeiras iteracións, a nosa variación atrase a obtención da solución óptima.

Unha vez feita a implementación en C++, o primeiro que facemos é comprobar que non hai erros derivados do código. Para elo, activamos a modificación fixando como límite de iteración do símplex dual unha cantidade suficientemente grande (por exemplo, 10^{15}) para que o algoritmo resolva ata optimalidade. Resolvemos as instancias dos tres conxuntos de problemas e comprobamos que coa versión estándar e a modificación activa, nestas condicións, obtemos resultados idénticos.

Diferentes comparacións co criterio de ramificación por defecto

Agora xa estamos en condicións de facer as primeiras probas determinantes. Probamos fixando o límite de iteracións do símplex dual nun 75 % das empregadas no nodo raíz durante as 10 e 100 primeiras iteracións. Tamén utilizamos o número de variables como número de iteracións nas que está activa esta modificación. Probamos tamén a, fixado en 10 o número de iteracións nas que está activa a modificación, baixar do 75 ao 50 % respecto o nodo raíz no símplex dual. Incluímos nos Cadros 4.2 e 4.3 un resumen moi reducido dos resultados das configuracións variando o número de iteracións nas que está activa a variación e variando o límite (en porcentaxe respecto ó nodo raíz) de iteracións do símplex dual coas librerías DS e MINLPLib, respectivamente.

Librería		**Estándar**	**75 %, 10 iter.**	**75 %, 100 iter.**
DS	resoltos (132/180)	132	127	85
	gap (96)	0.0051	+23.56 %	+1758.0 %
	mellor *gap* (96)	85	49	3
	tempo (106)	10.70	+45.41 %	+1643.84 %
	nodos (84)	120.24	+66.38 %	+696.87 %
	tempo linear (84)	0.0115	-9.13 %	+2.02 %
	mellor tempo linear (84)	0	9	42
Librería		**Estándar**	**50 %, 10 iter.**	**75 %, N.Var. iter.**
DS	resoltos (132/180)	132	125	128
	gap (96)	0.0051	+54.92 %	+ 23.9 %
	mellor *gap* (96)	85	43	53
	tempo (106)	10.70	+72.02 %	+38.31 %
	nodos (84)	120.24	+85.71 %	+87.59 %
	tempo linear (84)	0.0115	-14.74 %	-9.83 %
	mellor tempo linear (84)	0	26	7

Cadro 4.2: Comparación de varias configuracións de `RAPOSa` coa libraría DS.

Observación 4.19. Para os cálculos dos datos representados nas táboas excluíronse aqueles problemas que se resolveron en menos de cinco segundos en tódalas configuracións. O cálculo do *gap*, tempo e nodos fíxose tendo en conta os problemas que tiñan o dato correspondente para tódalas configuración. Por exemplo, se para un problema se obtivo un *gap* nunha configuración pero noutra non, non se ten en conta para o cálculo do mesmo. Tódalas medias son xeométricas, para evitar a descompensación na media pola disparidade de medidas entre problemas diferentes.

Os resultados, como podemos observar nos Cadros 4.2 e 4.3, non son favorables. Como esperabamos, o tempo linear reduciuse, aínda que non de xeito significativo. Por un lado, o valor do tempo linear obtense tendo en conta tódolos nodos, non só aqueles nos que a variación está activa. Tamén esperabamos que o número de nodos aumentase, pero non de xeito tan notorio, o que implica a resolución de máis subproblemas lineais. Con todo, está claro que a modificación funciona reduce o tempo linear, xa que a versión estándar acada tempos lineais máis altos en tódolos problemas da batería DS e , salvo tres instancias, de MINLPLib. Ademais, cantas máis iteracións está activa esta variación, ou máis agresiva é detendo o algoritmo do símplex dual, máis patente se fai este comportamento. Véxase que, aínda que o tempo linear aumente en media xeométrica na versión co 75 %

Librería		**Estándar**	**75 %, 10 iter.**	**75 %, 100 iter.**
MINLPLib	resoltos (118/166)	116	115	114
	gap (47)	0.4860	-2.74 %	+18.73 %
	mellor *gap* (47)	27	25	18
	tempo (35)	10.6776	+22.48 %	+105.99 %
	nodos (111)	107.0381	+82.76 %	+368.27 %
	tempo linear (111)	0.0007	-21.8 %	-13.64 %
	mellor tempo linear (111)	3	10	57
Librería		**Estándar**	**50 %, 10 iter.**	**75 %, N.Var. iter.**
MINLPLib	resoltos (118/166)	116	114	116
	gap (47)	0.4860	-2.4 %	+9.32 %
	mellor *gap* (47)	27	30	20
	tempo (35)	10.6776	+35.65 %	+30.49 %
	nodos (111)	107.0381	+145.23 %	+89.99 %
	tempo linear (111)	0.0007	-31.51 %	-13.37 %
	mellor tempo linear (111)	3	31	10

Cadro 4.3: Comparación de varias configuracións de `RAPOSa` coa libraría MINLPLib.

no símplex dual e 100 iteracións en `RAPOSa` no Cadro 4.2, esta configuración é a que mellor tempo linear ten en 42 dos 84 problemas. Logo, o feito de que en media xeométrica aumente o tempo linear respecto a configuración estándar non debe confundirnos.

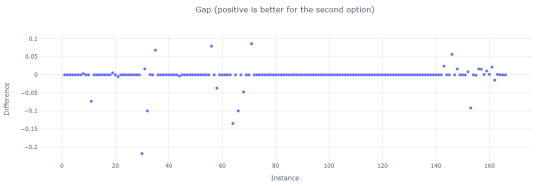
Como desvantaxe desta variacións vemos que o cómputo do tempo total aumenta, ata tal punto que na libraría DS pasamos de resolver 132 problemas a soamente 85 na versión co 75 % no símplex dual e 100 iteracións en `RAPOSa` nos 10 minutos fixados. O mesmo ocorre co *gap* nos problemas que non se resolveron en tódalas configuracións, que aumenta a medida que o fai o número de iteracións nos que está funcionando esta variación do algoritmo RLT.

En canto á versión onde o número máximo de iteracións nos que está activa esta variación é o número de variables do problema, os resultados son semellantes. A mellora no tempo linear e o aumento no resto de medidas son lixeiramente inferiores, pero isto débese principalmente a que o número de variables é unha cantidade pequena respecto ó total de iteracións.

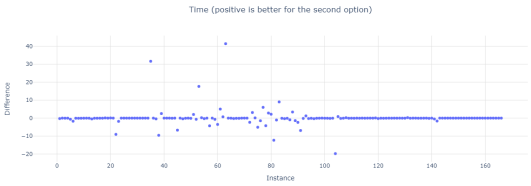
Se comparamos os resultados do Cadro 4.2 cos do Cadro 4.3, vemos que son máis favorables no segundo caso. Algo positivo, se temos en conta que se trata de problemas reais, non xerados aleatoriamente (como ocorre na libraría DS). De feito, nos problemas que non se resolveron no tempo fixado, na libraría MINLPLib o *gap* reduciuse lixeiramente, en media xeométrica, na configuración do 75 % nas 10 primeiras iteracións respecto a versión estándar. Se temos en conta o número de problemas nos que unha configuración obtén o mellor *gap*, vemos que co 50 % nas 10 primeiras se mellora esta cifra respecto a versión estándar.

Na Figura 4.10 incluímos catro gráficos comparando, para cada instancia, o *gap*, tempo total de resolución, número de nodos explorados e tempo linear entre a configuración

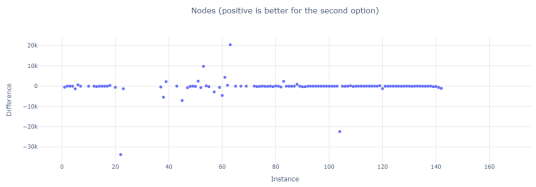
limitando nun 50 %, respecto o nodo raíz, o símplex dual nas 10 primeiras iteracións de `RAPOSa` e a estándar. Malia obter unhas conclusións claras coas medias xeométricas, coas diferencias problema a problema, podemos ver que existe unha certa variabilidade nos resultados (a excepción da Figura 4.10d). Por exemplo, hai problemas nos que a árbore de ramificación resultante ten menos nodos que na versión estándar. Isto pode deberse á aleatoriedade que existe na propia ramificación, xa que ramificar nun nodo por un punto distinto ou escoller unha nodo diferente que explorar, respecto á versión estándar, pode supoñer unha árbore final moi distinta. Por último, precisamente esa aleatoriedade



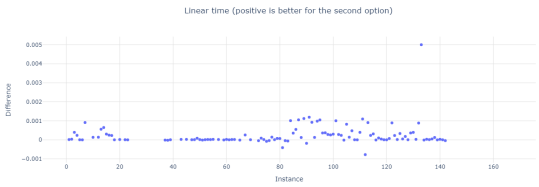
(a) Comparando o *gap*.



(b) Comparando o tempo total.



(c) Comparando o número de nodos explorados



(d) Comparando o tempo linear.

Figura 4.10: Gráficos das diferencias en *gap*, tempo, tempo linear e número de nodos explorados entre as configuracións estándar e limitando nun 50 %, respecto ó nodo raíz, as iteracións do símplex dual nas 10 primeiras iteracións de `RAPOSa`.

da que falabamos é a que provoca que nalgúns exemplo, como o da Figura 4.11, a cota inferior creza máis rapidamente (en número de iteracións, non só en tempo) fronte a versión estándar. Non obstante, o esperado é que precisemos algunhas iteracións máis para acadar o mesmo *gap* ca na versión estándar (e, polos resultados anteriores, tamén mais tempo de execución). Unha razón pola que estas modificacións poderían estar fallando é, como comentamos na Subsección 4.2.2, que o criterio de ramificación cos valores duais poida verse afectado. Repetimos o experimento das dúas primeiras configuracións fronte a estándar. Esta vez, establecendo o criterio de ramificación "sum". Os resultados, agora incluíndo tamén a libraría QPLib, recóllense no Cadro 4.4. En xeral, as conclusións son moi semellantes ás anteriores, polo que non comentaremos moito. No caso da librería QPLib, comprobamos que se trata de problemas de gran dificultade, pois a versión estándar só foi capaz de resolver 3 instancias co límite de tempo fixado en 10 minutos, mentres que as demais configuracións só unha. De novo, nos tres conxuntos de problemas as configuracións non estándar teñen efectos negativos sobre o rendemento do algoritmo baseado na RLT.

Fixemos tamén probas semellantes ás anteriores con outras configuracións que non foron probadas, cambiando o número de iteracións de `RAPOSa` nas que está activa a variación, así como modificando a porcentaxe de iteracións, respecto o nodo raíz, na que detemos o símplex dual. Tamén se probou a activar opcións que non o estaban na actual configuración estándar, como o *bound tightening*. En tódolos casos, as conclusión son semellantes e igualmente desfavorables polo que non pormenorizamos os resultados. Tamén, abrindo a posibilidade dun estudo con técnicas de aprendizaxe estatístico, se calcularon

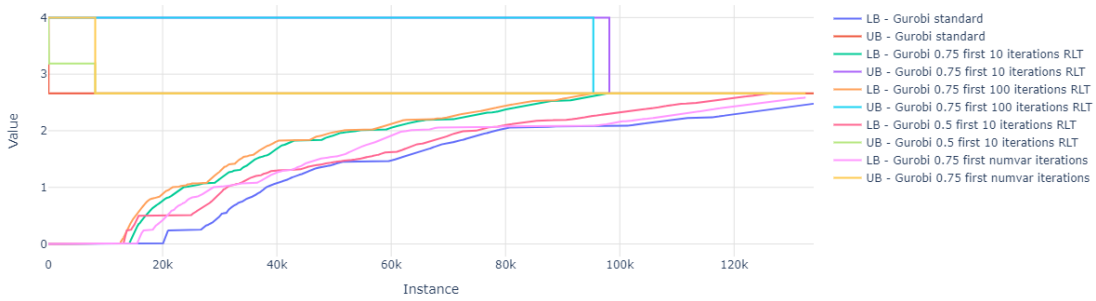


Figura 4.11: Evolución das cotas inferior e superior dun problema da libraría MINLPLib a medida que itera `RAPOSa` nas distintas configuracións.

Librería		**Estándar**	**75 %, 10 iter.**	**75 %, 100 iter.**
DS	resoltos (132/180)	131	129	82
	gap (98)	0.0050	+18.05 %	+2195.61 %
	mellor *gap* (98)	91	55	1 %
	tempo (105)	11.2301	+41.39 %	+1652.33 %
	nodos (82)	124.7469	+80.39 %	+652.01 %
	tempo linear (82)	0.0101	-9.4 %	-0.13 %
	mellor tempo linear (82)	5	26	51
MINLPLib	resoltos (118/166)	118	116	114
	gap (49)	0.3926	+3.23 %	+43.78 %
	mellor *gap* (49)	36	30	19
	tempo (36)	13.8067	+9.95 %	+127.22 %
	nodos (114)	113.3527	+71.48 %	+352.13 %
	tempo linear (114)	0.0007	-21.67 %	-33.87 %
	mellor tempo linear (114)	8	12	94
QPLib	resoltos (3/63)	3	1	1
	gap (56)	0.3822	+7.23 %	+15.11 %
	mellor *gap* (56)	44	17	13
	tempo (3)	115.087744	+44.76 %	+47.15 %
	nodos (1)	143	+61.54 %	+262.94 %
	tempo linear (1)	0.0114	-8.98 %	-19.32 %
	mellor tempo linear (1)	0	0	1

Cadro 4.4: Comparación de varias configuracións de `RAPOSa`, todas elas empregando o criterio de ramificación "sum".

os resultados medios escollendo para cada problema a mellor configuración. Os resultado foron moi semellantes ós obtidos coa versión estándar. Polo tanto, podemos concluír que na maioría dos problemas o mellor resultado se obtivo con esta configuración.

Por último, posto que un número de iteracións fixo sobre as que activar a variación do algoritmo pode ser un parámetro moi ríxido, probamos a fixar unha porcentaxe sobre o

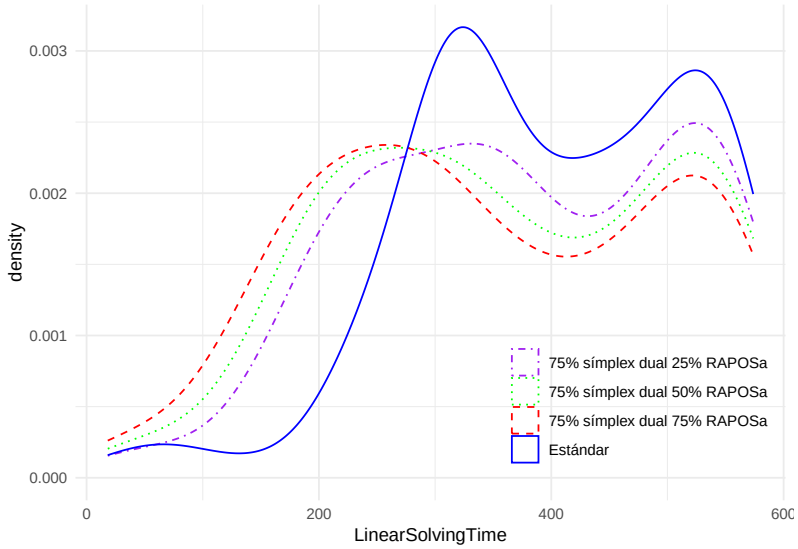


Figura 4.12: Estimación da densidade tipo núcleo do tempo linear de resolución.

número de iteracións que require cada problema na versión estándar. Aínda que isto non é aplicable na práctica, no caso de acadar uns resultados positivos, podería ser interesante de cara un futuro estudo empregando técnicas de aprendizaxe estadístico. Ademais, estudaremos estas configuracións en problemas que non se resolven en 10 minutos e que teñen máis de 50 iteracións. Así concluiremos definitivamente se, incluso nas situación con maior potencial, o comportamento é ou non desfavorable. Incluímos no Cadro 4.5 os resultados de 4 novas configuracións. Neste caso, estudamos tamén o comportamento da variación establecendo unha cantidade fixa (non unha porcentaxe). Os resultados, como podemos observar, son semellantes ós vistos ata agora. A configuración estándar é a que obtén un mellor *gap* en 88 dos 129 problemas e está moi cerca de selo en outras 32 instancias. Logo claramente segue sendo a configuración cun mellor rendemento.

	Estándar	**75 %, 10 %**	**75 %, 25 %**	**90 %, 25 %**	**1000, 25 %**
gap (129)	0.2351	+30.94 %	+45.72 %	+30.72 %	+13.72 %
case mellor *gap* (129)	120	50	38	41	81
mellor *gap* (129)	88	32	26	32	58

Cadro 4.5: Comparación de varias configuracións fixando como límite de iteracións para cada problema polinómico unha porcentaxe determinada respecto as empregadas na configuración estándar.

Aínda que xa comprobamos que a nosa variación ten un claro efecto negativo sobre o rendemento xeral do algoritmo, podemos facer un último análise gráfico para entender como varían os tempos lineais, e en particular, o tempo empregado na resolucións das relaxacións lineais. Para elo, importamos en R os resultado de executar en `RAPOSa` as configuracións empregando o 75 % das iteracións no símplex dual modificando a porcentaxe de iteracións, respecto a versión estándar, na que está activa a variación. Comezamos representando na Figura 4.12 unha estimación da densidade tipo núcleo para os tempos lineais de resolución. Neste gráfico, podemos ver que a densidade estimada para as tres modificacións é menor á da configuración estándar en tempos maiores a 300 segundos. É dicir, ao deter o algoritmo do símplex dual nunha certa porcentaxe das iteracións totais

de RAPOSa estamos conseguindo reducir o tempo empregado para a resolución das relaxacións lineais, como xa inferiamos do estudo computacional. É máis, observase que o efecto é claramente maior cantas máis iteracións esta activa a variación do algoritmo.

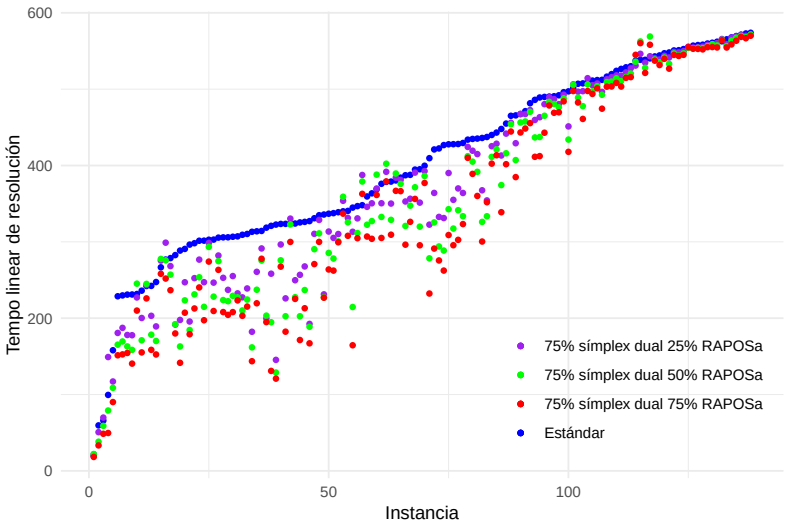


Figura 4.13: Representación do tempo linear de resolución para cada problema e configuración.

Na Figura 4.13, representamos para cada problema o tempo de resolución linear nas catro configuracións anteriores, ordenado os problemas segundo o tempo linear de resolución na versión estándar. Na Figura 4.13 podemos comprobar que a variacións feita no algoritmo non é capaces, en ningunha configuración, de reducir o tempo linear de resolución cando estes son moi grandes (maiores a 500 segundos). Podemos comprobar que estas instancias (aquelas con tempo linear de resolución maior a 500 segundos) non se tratan de problemas con relaxacións lineais complexas. Para elo, facemos unha gráfica semellante a anterior tendo en conta o tempo linear de resolución medio por nodo (Figura 4.13). Vemos que, de acordo coas ideas presentadas na Subsección 4.2.4, a variación sobre o algoritmo é especialmente efectiva en problemas lineais complexos, onde o símplex dual require máis dun segundo para acadar a solución óptima.

Para rematar, posto que estamos empregando o mesmo conxunto de problemas que na Subsección 4.2.4, podemos comprobar se a diminución do tempo linear de resolución se achega ao esperado (Figura 4.5), mediante un *box-plot*. En particular, para unha configuración onde se establece o límite de iteracións do símplex dual no 75 % respecto as empregadas no nodo raíz, durante tódalas iteracións de RAPOSa, se espera unha redución do 25 % no cómputo do tempo linear de resolución. Pois ben, a Figura 4.15 indícanos precisamente iso, xa que empregando a modificación nun 75 % das iteracións de RAPOSa (respecto á versión estándar) a redución do tempo linear de resolución é maior ó 25 % na maioría das instancias. Un dato moi importante, pois é o argumento definitivo para afirmar que o comportamento da modificación neste sentido é correcto, aínda que en termos xerais o rendemento é claramente inferior.

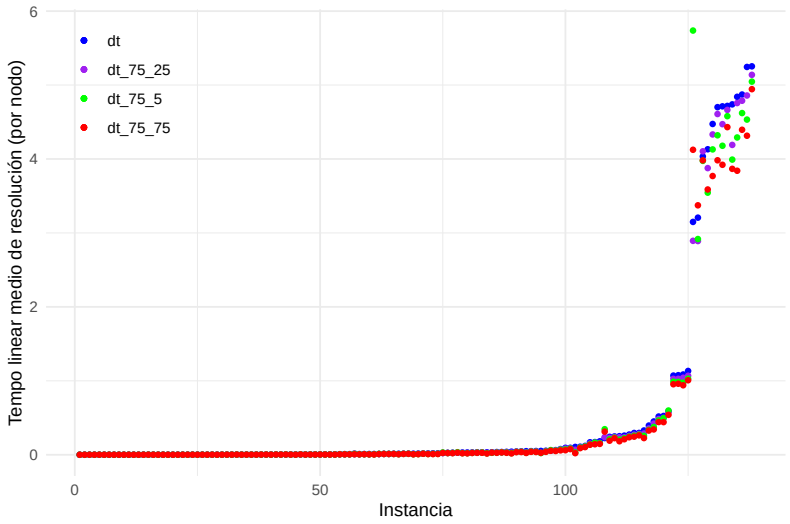


Figura 4.14: Representación do tempo linear de resolución medio por nodo explorado para cada problema e configuración.

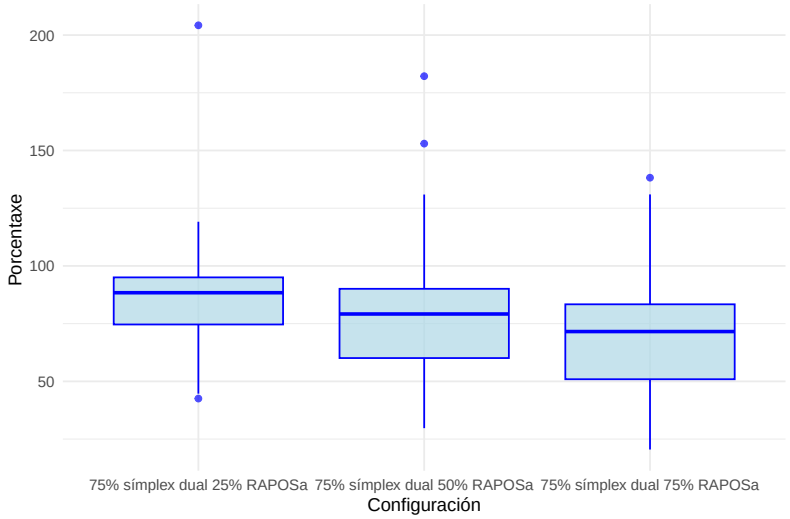


Figura 4.15: Porcentaxe do tempo empregado para a resolución das relaxacións lineais con cada configuración respecto á versión estándar.

4.3. Conclusións

Neste capítulo propoñemos unha posible variación sobre o algoritmo de ramificación e acoutamento, que consiste en resolver durante a ramificación espacial unha certa cantidade de nodos sen chegar a optimalidade, co fin de axilizar o crecemento da cota inferior do valor óptimo e reducir o tempo de resolución. Presentamos unha sólida base teórica na que fundamentamos esta modificación e adaptamos coidadosamente o algoritmo RLT para asegurar a converxencia do mesmo. Tamén demostramos analítica e graficamente o forte potencial da nosa proposta e comprobamos, con algunhas probas preliminares, que funciona como agardabamos. Por un lado, demostramos que na maioría das instancias o tempo que se adica para resolver as relaxacións lineais é unha porcentaxe alta do tempo total de resolución do problema polinómico. Por outro, nas probas preliminares, conseguimos reducir o tempo que emprega o símplex dual para a resolución do nodo raíz obtendo unha cota inferior moi próxima ó valor óptimo. Con este fortes argumentos decidimos implementar, da man do equipo de desenvolvemento de `RAPOSa`, esta variación no algoritmo. Despois de facer probas cunha gran variedade de configuracións distintas, especialmente deseñadas para estudar diferentes situacións, presentamos os resultados. En tódolos casos, logramos reducir o tempo linear e, en particular, o tempo linear dedicado a resolución das relaxacións lineais. É máis, a redución do tempo linear de resolución respecto a versión estándar é precisamente a esperada. Non obstante, a lixeira aleatoriedade na árbore de ramificación, adulterada polas adaptacións necesarias que fixemos sobre o algoritmo baseado na RLT, xunto cun aumento do número de nodos, algo superior ó esperado, fai que o tempo necesario para a resolución dun problema polinómico, en xeral, aumente. En aquelas instancias que non se resolveron en 10 minutos, as configuracións coa modificación activa obteñen un *gap* inferior á versión estándar só en contadas ocasións. Polo tanto, malia reducir o tempo linear podemos concluír que esta variación, en xeral, non é capaz de mellorar o rendemento do algoritmo baseado na RLT. Ademais, tamén descartamos calquera estudo futuro no ámbito da aprendizaxe estatística, pois non existe unha marxe de mellora suficiente.

Bibliografía

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. (2010). *Linear programming and network flows*. John Wiley & Sons, 4th edition.

Bussieck, M. R., Drud, A. S., and Meeraus, A. (2003). MINLPLib – a collection of test models for mixed-integer nonlinear programming. *INFORMS J. Comput.*, 15(1):114–119.

Chang, Y.-J. and Wah, B. (1994). Polynomial programming using groebner bases. In *Proceedings Eighteenth Annual International Computer Software and Applications Conference*, pages 236–241.

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

Dalkiran, E. and Sherali, H. D. (2013). Theoretical filtering of RLT bound-factor constraints for solving polynomial programming problems to global optimality. *J. Glob. Optim.*, 57(4):1147–1172.

Dalkiran, E. and Sherali, H. D. (2016). RLT-POS: reformulation-linearization technique-based optimization software for solving polynomial programming problems. *Math. Program. Comput.*, 8(3):337–375.

Dantzig, G. B. (1963). Linear programming and extensions. *A RAND Corporation Research Study. Princeton University Press* .

Dantzig, G. B. and Thapa, M. N.-D. (2003). *Linear programming. 2: Theory and extensions*. Springer Ser. Oper. Res. New York, NY: Springer.

Dua, V. (2015). Mixed integer polynomial programming. *Computers & Chemical Engineering*, 72:387–394. A Tribute to Ignacio E. Grossmann.

Fourer, R., Gay, D. M., and Kernighan, B. W. (1990). A modeling language for mathematical programming. *Manage. Sci.*, 36(5):519–554.

Furini, F., Traversi, E., Belotti, P., Frangioni, A., Gleixner, A., Gould, N., Liberti, L., Lodi, A., Misener, R., Mittelmann, H., Sahinidis, N. V., Vigerske, S., and Wiegele, A. (2019). QPLIB: a library of quadratic programming instances. *Math. Program. Comput.*, 11(2):237–265.

González-Díaz, J. (2018). Programación lineal y entera. Universidade de Santiago de Comspotela.

González-Díaz, J., González-Rodríguez, B., Leal, M., and Puerto, J. (2021). Global optimization for bilevel portfolio design: Economic insights from the dow jones index. *Omega*, 102:102353.

González-Rodríguez, B. (2017). Introducción a la programación no lineal. Traballo fin de grao, Universidade de Santiago de Compostela.

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

González-Rodríguez, B., Ossorio-Castillo, J., González-Díaz, J., González-Rueda, Á. M., Penas, D. R., and Rodríguez-Martínez, D. (2023). Computational advances in polynomial optimization: RAPOSa, a freely available global solver. *J. Glob. Optim.*, 85(3):541–568.

González-Rodríguez, B. and Naoum-Sawaya, J. (2024). Degree reduction techniques for polynomial optimization problems.

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

Gómez-Casares, I., González-Rodríguez, B., González-Díaz, J., and Rodríguez-Fernández, P. (2024). Impact of domain reduction techniques in polynomial optimization: A computational study.

Hägglöf, K., Lindberg, P. O., and Svensson, L. (1995). Computing global minima to polynomial optimization problems using Gröbner bases. *J. Glob. Optim.*, 7(2):115–125.

Karaca, O., Darivianakis, G., Beuchat, P., Georghiou, A., and Lygeros, J. (2017). The repop toolbox: Tackling polynomial optimization using relative entropy relaxations. *IFAC-PapersOnLine*, 50(1):11652–11657. 20th IFAC World Congress.

Karia, T., Adjiman, C. S., and Chachuat, B. (2022). Assessment of a two-step approach for global optimization of mixed-integer polynomial programs using quadratic reformulation. *Computers & Chemical Engineering*, 165:107909.

Lasserre, J. B. (2001). Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817.

Lasserre, J. B., Toh, K.-C., and Yang, S. (2017). A bounded degree SOS hierarchy for polynomial optimization. *EURO J. Comput. Optim.*, 5(1-2):87–117.

Laurent, M. (2009). Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, New York.

Nie, J. (2013). An exact Jacobian SDP relaxation for polynomial optimization. *Math. Program.*, 137(1-2 (A)):225–255.

Nie, J. (2014). Optimality conditions and finite convergence of Lasserre's hierarchy. *Math. Program.*, 146(1-2 (A)):97–121.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity.* Dover Publications, Inc., Mineola, New York.

Puranik, Y. and Sahinidis, N. V. (2017). Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, 22(3):338–376.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Rockafellar, R. T. (2007). *Fundamentals of optimization*. University of Washington, Seattle. Lecture Notes.

Salazar-González, J. J. (2001). *Programación Matemática*. Díaz de Santos.

Sherali, H. D., Dalkiran, E., and Desai, J. (2012). Enhancing RLT-based relaxations for polynomial programming problems via a new class of v-semidefinite cuts. *Comput. Optim. Appl.*, 52(2):483–506.

Sherali, H. D. and Fraticelli, B. M. P. (2002). Enhancing RLT relaxations via a new class of semidefinite cuts. *J. Glob. Optim.*, 22(1-4):233–261.

Sherali, H. D. and Tuncbilek, C. H. (1992). A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *J. Glob. Optim.*, 2(1):101–112.

Sherali, H. D. and Tuncbilek, C. H. (1997). New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters*, 21(1):1–9.

Teles, J. P., Castro, P. M., and Matos, H. A. (2013). Multi-parametric disaggregation technique for global optimization of polynomial programming problems. *J. Glob. Optim.*, 55(2):227–251.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1 (A)):25–57.