Trabajo Fin de Máster

Método de descomposición de Benders. Rendimiento del algoritmo y aplicaciones

Daniel Alves Yáñez

Máster en Técnicas Estadísticas

Curso 2018-2019

Propuesta de Trabajo Fin de Máster

Título en galego: Método de descomposición de Benders. Rendemento do algoritmo e aplicacións

Título en español: Método de descomposición de Benders. Rendimiento del algoritmo y aplicaciones

English title: Benders decomposition method. Algorithm performance and applications.

Modalidad: Modalidad A

Autor: Daniel Alves Yáñez, Universidad de Santiago de Compostela

Directores: Julio González Díaz, Universidad de Santiago de Compostela; Jorge Rodríguez Veiga, ITMATI (Instituto Tecnológico de Matemática Industrial)

Breve resumen del trabajo:

El objetivo de este trabajo ha sido la implementación del algoritmo de Benders generalizado y su aplicación sobre una clase de problemas conocida como los problemas de pooling. Se parte del método de descomposición de Benders original y se presentan los aspectos teóricos necesarios para la generalización del método. Los problemas de pooling están catalogados como problemas difíciles de resolver y sirven para modelar plantas de procesamiento industrial. Las herramientas de optimización convencionales presentan dificultades para resolver este tipo de problemas. Por ello, utilizando el método de descomposición de Benders generalizado se trata de aprovecharse de la estructura de los mismos para conseguir buenos óptimos locales.

Recomendaciones: tener unos conocimientos mínimos sobre programación matemática.

Don Julio González Díaz, Profesor de la Universidad de Santiago de Compostela, don Jorge Rodríguez Veiga, Investigador de ITMATI (Instituto Tecnológico de Matemática Industrial), informan que el Trabajo Fin de Máster titulado

Método de descomposición de Benders. Rendimiento del algoritmo y aplicaciones

fue realizado bajo su dirección por don Daniel Alves Yáñez para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 4 de Septiembre de 2019.

El director:

GONZALEZ DIAZ JULIO - 33332679K

Firmado digitalmente por GONZALEZ DIAZ JULIO - 33332679K
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-33332679K, givenName=JULIO, sn=GONZALEZ DIAZ, cn=GONZALEZ DIAZ JULIO - 33332679K
Fecha: 2019.09.05 12:15:35 +02'00'

Don Julio González Díaz

El director:

RODRIGUEZ VEIGA JORGE - 33539881Q

Firmado digitalmente por RODRIGUEZ VEIGA JORGE - 33539881Q
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-33539881Q, givenName=JORGE, sn=RODRIGUEZ VEIGA, cn=RODRIGUEZ VEIGA JORGE - 33539881Q
Fecha: 2019.09.05 10:51:36 +02'00'

Don Jorge Rodríguez Veiga

El autor:

ALVES YAÑEZ DANIEL - 17463670T

Firmado digitalmente por ALVES YAÑEZ DANIEL - 17463670T
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-17463670T, givenName=DANIEL, sn=ALVES YAÑEZ, cn=ALVES YAÑEZ DANIEL - 17463670T
Fecha: 2019.09.05 10:10:10 +02'00'

Don Daniel Alves Yáñez

Agradecimientos

Antes de comenzar con los contenidos de este trabajo quiero agradecer a mi familia todo el apoyo y esfuerzo que han realizado para que esto fuera posible.

Del mismo modo, quisiera agradecer a mis directores Julio González Díaz y Jorge Rodríguez Veiga por toda la dedicación, paciencia y apoyo que me han brindado durante este último año.

También quisiera agradecer al ITMATI (Instituto Tecnológico de Matemática Industrial) por poner a mi disposición todo los medios que he necesitado para realizar esta memoria. En especial, quiero agradecer a todos mis compañeros de trabajo de ITMATI: Fito, Rabi, Jorge, Andrea, Javi, Gabriel Álvarez, Gabriel Capeáns, Damián y Diego por crear un ambiente de trabajo muy especial y único.

Por último, pero no menos importante, quiero darle las gracias a Daniel Miles Touya. Sin su ayuda y motivación, probablemente, no habría tomado el camino de la estadística, optimización y programación que tanto me apasiona.

Índice general

Resumen

Resumen en español

El propósito de este trabajo ha sido la implementación del método de descomposición de Benders generalizado para resolver una clase de problemas conocida como los problemas de pooling. Los problemas de pooling se utilizan para modelar plantas de procesamiento industrial en industrias como la petroquímica o la gasística. La dificultad de los mismos radica en que son problemas no lineales y no convexos. Con el método de descomposición de Benders generalizado nos aprovechamos de la estructura de este tipo de problemas para conseguir buenos óptimos locales. Se propone una nueva formulación para el subproblema en los modelos de pooling. Los resultados principales del estudio son que con la utilización de este método para resolver los problemas de pooling se consiguen resultados competentes con los de los *solvers* comerciales. Además, incluso en instancias en las que las herramientas comerciales de optimización no son capaces de conseguir soluciones factibles en un tiempo prefijado, el método de descomposición de Benders si consigue buenos óptimos locales. Por último, se han comparado nuestros resultados con los de la literatura y nuestra herramienta de optimización es capaz de encontrar mejores óptimos.

English abstract

The goal of this work has been the implementation of the generalized Benders decomposition method to solve a class of problems known as pooling problems. Pooling problems are used to model industrial processing plants in industries such as petrochemicals or gas. Their difficulty is that they are nonlinear and non-convex problems. With the generalized Benders decomposition method, we take advantage of the structure of these types of problems to achieve good local optimum. A new formulation is proposed for the subproblem in pooling models. The main results of the study are that with the use of this method to solve pooling problems, competent results are achieved with those of commercial solvers. In addition, even in instances where commercial optimization tools are not able to achieve feasible solutions in a predetermined time, Benders decomposition method does achieve local optimum. Finally, our results have been compared with those of the literature and our optimization tool is able to find better local optimal solutions.

Prefacio

Este trabajo está motivado por las labores de investigación llevadas a cabo por el autor, en colaboración con el ITMATI, sobre la optimización de plantas de proceso industrial. Por motivo de los acuerdos de confidencialidad firmados por ITMATI y el autor no se podrá exponer el código implementado.

En la actualidad los problemas de programación matemática de gran dimensión están muy presentes en la industria ya que permiten modelar de una manera realista multitud de procesos. Por ejemplo, en plantas de producción industrial se utilizan este tipo de modelos para la toma de decisiones tanto en escenarios presentes como futuros. En muchas ocasiones resolver de manera eficiente o simplemente obtener un buen óptimo local para los escenarios presentes con este tipo de problemas puede resultar una tarea difícil. Por ello, es de gran importancia la exploración de técnicas que permitan obtener soluciones óptimas para estos problemas es tiempos razonables.

Dentro del mundo de la programación matemática podemos clasificar los problemas en dos grandes campos: lineales y no lineales. Para los primeros, considerados fáciles, existen algoritmos como el método símplex o versiones mejoradas del mismo que consiguen resolver problemas con una gran dimensión muy eficientemente y en poco tiempo. Para los segundos, los algoritmos existentes ya no son tan eficientes como el símplex. En el caso más difícil, problemas no lineales y no convexos, los algoritmos existentes pueden tardar mucho tiempo incluso en dar con una solución local del mismo.

Debido a la dificultad para la consecución de soluciones en los problemas de planificación industrial autores como (Dantzig & Wolfe 1960) o (Benders 1962), a principios de los años sesenta, propusieron técnicas de descomposición con el objetivo de acelerar el proceso para la obtención de soluciones óptimas de dichos problemas. En el presente documento unicamente se empleará un método de descomposición basado en Benders ya que es más adecuado para resolución de los problemas que han aparecido en las colaboraciones con ITMATI.

La idea esencial del método de descomposición de Benders es acelerar el proceso de búsqueda de soluciones óptimas apoyándose en la estructura de determinados problemas. El enfoque original está basado en descomponer el problema original en dos problemas: el subproblema y el maestro. Con ellos lo que se consigue es tener dos problemas que son más sencillos de resolver, por separado, que el problema original. Supongamos que en ciertas restricciones del problema original están implicadas variables enteras o variables que conllevan la aparición de no linealidades en esas restricciones, lo que se trata de hacer con la descomposición de Benders es separar de alguna forma el problema original teniendo en cuenta estas variables. Una vez se tenga realizada la separación del problema original se resuelve de manera iterativa e independiente ambos problemas permitiendo una comunicación entre ellos basada en la información de los problemas duales. La ventaja principal de este algoritmo es que puede ser adaptado para multitud de problemas de programación matemática: lineales, lineales con variables enteras, no lineales, no lineales con variables enteras etc.

El objetivo de este trabajo es estudiar su rendimiento sobre una clase de problemas académicos conocida como los problemas de pooling. La elección de estos problemas está justificada ya que debido

a su naturaleza son los que más se asemejan a los problemas reales sobre los que el autor investiga en colaboración con ITMATI. Además, dentro del campo de la complejidad computacional, estos problemas se encuentran dentro de la clase NP-duros (Alfaki & Haugland 2013) con lo que nos permite obtener resultados realista sobre el comportamiento del algoritmo ya que son muy difíciles de resolver.

La organización de esta memoria es la siguiente: en el primer capítulo se presentan el método de descomposición de Benders en su formulación original explicando los conceptos teóricos y prácticos del método acompañados al final del capítulo por un ejemplo; en el segundo capítulo se presentan los problemas de pooling justificando su importancia a la hora de formular procesos de planificación industrial y se presentan dos formulaciones equivalentes de problema; en el tercer capítulo se presentan la extrapolación del método de Benders para problemas de optimización no lineal, aspectos específicos del modelado para los problemas de pooling al utilizar dicho algoritmo y se presentan dos heurísticas diseñadas para mejorar el rendimiento del mismo. En el capítulo cuarto se presentan los resultados computacionales obtenidos con los distintos métodos implementados y aplicados sobre los problemas de pooling. Por último, en el capítulo quinto se presentarán las conclusiones del trabajo.

En determinadas situaciones se tomará el abuso de notación $u^t = u$ siempre que se considere que esto puede facilitar la comprensión al lector.

Capítulo 1

Método de descomposición de Benders

A principios de los años sesenta Jacques F. Benders presenta en Benders (1962) un nuevo método para la resolución de problemas de programación matemática entera mixta (MIP). La dificultad de este tipo de problemas es que algunas de las variables que lo definen son de naturaleza entera. Este hecho provoca que se pierda la propiedad de la convexidad de la región factible dificultando la solución del mismo. Otras técnicas que se utilizan en la práctica para resolver este tipo de problemas son ramificación y acotación (Land & Doig 1960) o de planos de corte Kelley (1960) que no son tan eficientes. La ventaja principal del algoritmo de Benders, frente a los otros dos, es que este se basa la idea de identificar algún tipo de estructura que nos permita separar el problema en dos: el subproblema y el maestro. Esto permite resolver dos problemas que de forma separada son más sencillos de resolver, desde el punto de vista computacional, que el problema original.

En la literatura sobre Benders la aparición de variables que dificultan la resolución de los problemas de programación matemática son conocidas como variables complicantes. Es decir, son aquellas variables que en el caso de ser fijadas como parámetros provocarían que el proceso de búsqueda de la solución óptima fuese mucho más sencillo. Por ejemplo, son complicantes aquellas variables enteras implicadas en restricciones lineales o aquellas variables que provocan no linealidades en restricciones y en caso de ser fijadas como parámetros implicarían que dichas restricciones fuesen lineales. También se entiende como variables complicantes aquellas que permiten identificar algún tipo de estructura en el problema que nos permita separarlo en dos problemas diferentes. En definitiva cualquier variable que en caso de no estar involucrada en el problema haría que este tuviera unas mejores propiedades matemáticas desde el punto de vista de la optimización y que además permita identificar una estructura de separación se considerará una variable complicante.

Dado el siguiente problema MIP:

$$\begin{aligned} \text{mín} \quad & c^t x + f(y) \\ \text{s.a.} \quad & Ax + F(y) \geq b \\ & x \geq 0, y \in \mathrm{S} \subset \mathbb{Z}^p, \end{aligned} \tag{1.1}$$

donde c y x son vectores en $\mathbb{R}^m$, $f(y)$ es una función escalar en $\mathbb{R}$, A es una matriz definida en $\mathbb{R}^{n \times m}$, tanto $F(y)$ como b están definidas en $\mathbb{R}^n$ y S es un subconjunto de $\mathbb{Z}^p$. Nótese que no se especifica ningún supuesto sobre $f(y)$ y $F(y)$. Lo único que se exige en este método es que los problemas resultantes de la separación puedan ser resueltos con los métodos o *solvers* existentes.

Como se puede observar en (1.1) aparecen dos conjuntos de variables claramente diferenciados: por un lado, variables continuas x, por otro lado variables enteras y. Además en caso de ser eliminados o fijados los términos en los que están implicadas las variables y se tendrían problemas lineales. Este hecho lleva a la idea principal del método de descomposición de Benders que consiste en aprovecharse de la estructura del problema para separarlo en dos problemas: el subproblema que será lineal y el maestro que puede ser lineal, no lineal, discreto etc. En vez de resolver el problema completo que resultaría complejo, se resolverán, de forma iterativa, dos problemas de optimización por separado más sencillos desde el punto de vista computacional.

1.1. Deducción del subproblema y el maestro

Para simplificar la explicación de cómo se crean ambos problemas y cual es la relación entre ellos se introduce el siguiente problema de programación lineal entero mixto (MILP):

$$\begin{aligned} \text{mín} \quad & c^t x + d^t y \\ \text{s.a.} \quad & Ax + Dy \geq b \\ & x \geq 0, y \in \mathrm{S} \subset \mathbb{Z}^p, \end{aligned} \tag{1.2}$$

se trata de una modificación del problema (1.1) considerando que las funciones $f(y)$ y $F(y)$ son lineales y donde $\mathrm{S} \subset \mathbb{Z}^p$, lo que implica que las variables y han de tomar valores enteros.

Tal y como se ha comentado con anterioridad el método de descomposición de Benders es un método iterativo que se apoya en la resolución de dos problemas, el subproblema y el maestro, pero estos problemas no se resuelven de forma independiente sino que hay una comunicación entre ellos a través de la información de los problemas duales. En un sentido menos formal, el problema maestro hace propuestas para las variables complicantes, el subproblema resuelve fijando dichas variables y le transmite información al maestro mediante el dual.

A modo explicativo se va a considerar que la variable complicante del problema (1.2) es la y debido a su carácter entero. Por lo tanto, se puede definir el subproblema lineal como:

$$\begin{aligned} \text{mín} \quad & c^t x + d^t \bar{y} \\ \text{s.a.} \quad & Ax \geq b - D\bar{y} \\ & x \geq 0, \end{aligned} \tag{1.3}$$

el problema dual asociado a (1.3), ignorando el término constante $d^t \bar{y}$ es

$$\begin{aligned} \text{máx} \quad & u^t(b - D\bar{y}) \\ \text{s.a.} \quad & A^t u \leq c \\ & u \geq 0. \end{aligned} \tag{1.4}$$

La primera relación que se puede deducir del problema (1.3) y de su dual (1.4), ignorando el término constante $d^t \bar{y}$, es que si x y u son soluciones factibles del primal y del dual, respectivamente. Entonces, $Ax \geq b - D\bar{y}$, $x \geq 0$, $A^t u = uA \leq c$ y $u \geq 0$. Aplicando estas relaciones obtenemos que

$$c^t x \geq (uA)^t x = u^t(Ax) \geq u^t(b - D\bar{y}). \tag{1.5}$$

Esta relación se conoce como propiedad de *dualidad débil*.

Proposición 1.1. *(Dualidad débil). Dado un par de problemas primal (P) y dual (D), la función objetivo asociada a cualquier solución factible del problema de minimización es mayor o igual que la función objetivo asociada a cualquier solución factible del problema de maximización.*

De esta proposición y de las relaciones propuestas en (1.5) se deducen varias implicaciones. Las soluciones factibles del problema de minimización siempre serán cotas superiores para el problema de maximización y, equivalentemente, las soluciones factibles para los problemas de maximización siempre serán cotas inferiores del problema de minimización. Esta relación es de gran importancia ya que se cumple para cualquier par (P) y (D) independientemente de que sean lineales o no y será una de las piezas fundamentales del método de Benders. Por otra parte, la propiedad de dualidad débil permite obtener de forma inmediata los dos siguientes corolarios.

Corolario 1.1. *Si x y u son soluciones factibles del primal y del dual tales que $c^t x = u^t(b - D\bar{y})$, entonces x y u son soluciones óptimas de sus respectivos problemas.*

Corolario 1.2. *Dado un par problemas (P) y (D), si la función objetivo de uno de ellos es no acotada, entonces el otro no tiene soluciones factibles.*

La última propiedad que se necesitará para entender las relaciones entre el subproblema y el maestro es la de la *dualidad fuerte*.

Proposición 1.2. *(Dualidad fuerte). Dado un par de problemas primal (P) y dual (D), si uno de ellos tiene una solución óptima, entonces también el otro tiene solución óptima y los valores óptimos de la función objetivo coinciden. En particular, si x^* y u^* son soluciones óptimas de (P) y (D), respectivamente, tenemos que*

$$c^t x^* = u^{*t}(b - D\bar{y}). \tag{1.6}$$

Es importante destacar que, a diferencia de la propiedad de dualidad débil que siempre se cumple incluso en problemas no lineales, esta propiedad puede no cumplirse si los problemas no son lineales. Una condición suficiente para que se cumpla la propiedad de la dualidad fuerte es que tanto la función objetivo como la región factible del problema sean convexas.

A continuación se presentan tres subsecciones en las que con la ayuda de los problemas (1.3) y (1.4) juntos con las relaciones y propiedades matemáticas obtenidas a partir de los mismos, se tratará de explicar en detalle como se construye el problema maestro y las relaciones de este con el subproblema en forma de restricciones que se denominan cortes. Para realizar dichos cortes es necesario determinar los puntos extremos y direcciones extremas del conjunto factible del problema (1.4).

1.1.1. Cortes de optimalidad

Los cortes de optimalidad se derivan de los puntos extremos de (1.4). Una de las relaciones entre un par (P) y (D) es que si la función objetivo del primero es acotada entonces el segundo tiene soluciones factibles. Por lo tanto, para la construcción de estos cortes lo primero que se necesita es que el subproblema tenga una solución factible finita y, entonces, se tratará de buscar un punto extremo en su dual. Una solución óptima del dual (1.4) se corresponde con identificar un punto extremo u^* de la región factible del dual. Al ser este un problema de programación lineal implica que su región factible está definida por un poliedro convexo. Entonces, dicho punto se puede caracterizar como:

Proposición 1.3. *Dado un conjunto convexo C, un punto $u \in C$ es un punto extremo de C si no puede ser representando como una combinación convexa estricta de dos puntos distintos de C. Formalmente, el vector z es un punto extremo de C si la igualdad $z = \lambda x + (1 - \lambda)y$ con $\lambda \in (0, 1)$ implica que $z = x = y$. Donde z, x e y son vectores y λ es un escalar.*

A modo explicativo la proposición anterior junto con la propiedad de que la región factible del problema dual está definida por un poliedro convexo implica que los puntos extremos se encontrarán en las esquinas de dicho poliedro.

Supongamos que la función objetivo óptima del problema (1.3) es un escalar α y la solución óptima

del dual asociado (1.4) es un vector u^*. Por la tanto, la propiedad de la dualidad fuerte nos asegura que

$$\alpha - d^t\bar{y} = u^{*t}(b - D\bar{y}). \tag{1.7}$$

Además como la región factible del dual (1.4) no depende del valor fijado de las variables complicantes, u^* será factible para cualquiera valor fijo de dichas variables. Debido a que si el primal es factible y tiene óptimo finito siempre existirá un u^* que será factible en el dual, por la propiedad de la dualidad débil tenemos:

$$\alpha \geq d^t\bar{y} + u^{*t}(b - D\bar{y}). \tag{1.8}$$

Las consecuencias derivadas de la dualidad débil nos llevan a que la desigualdad (1.8) proporciona una cota inferior para el problema original. Como la desigualdad es válida para cualquier valor de y, se introducirá en el maestro como un corte de optimalidad.

1.1.2. Cortes de factibilidad

Lo siguiente que hay que considerar es que la solución del problema (1.3) sea infactible. En este caso, derivado de las relaciones entre el problema primal y el dual puede ocurrir, en general, que el dual asociado (1.4) esté no acotado en ese punto o sea infactible. Para entender en profundidad las implicaciones de la no acotación de este problema necesitamos introducir dos conceptos importantes en problemas de programación lineal: direcciones extremas y el teorema de Carathéodory.

Proposición 1.4. *Dado un conjunto convexo C, una dirección d de C es una dirección extrema si no puede ser representada como una combinación cónica estricta de dos direcciones distintas de C. Formalmente, el vector d es una dirección extrema de C si la igualdad $d = \lambda_1 c^1 + \lambda_2 c^2$ con $\lambda_1 > 0$ y $\lambda_2 > 0$, implica que d, c^1 y c^2 representan la misma dirección. Donde λ_1 y λ_2 con escalares y, por su parte c^1 y c^2 son vectores.*

Teorema 1. *(Teorema de Carathéodory). Sea $C \subseteq \mathbb{R}^n$ el poliedro no vacío definido por $\{x : Ax \leq b, x \geq 0\}$, entonces un punto $\bar{x}$ pertenece a C si y sólo si puede ser representado como una combinación convexa de los puntos extremos más una combinación cónica de sus direcciones extremas.*

Entonces, por el teorema anterior podemos reescribir el problema (1.4) en función de los puntos extremos y direcciones extremas del conjunto $\{u \in \mathbb{R}^n : A^t u \leq c\}$ de la manera siguiente:

$$\begin{aligned} \text{máx} \quad & \sum_{i=1}^{n} \lambda_i (b - D\bar{y})^t u^i + \sum_{j=1}^{m} \mu_j (b - D\bar{y})^t v^j \\ \text{s.a.} \quad & \sum_{i=1}^{n} \lambda_i = 1, \\ & \lambda_i \geq 0, \qquad i \in \{1, 2, ..., n\}, \\ & \mu_j \geq 0, \qquad j \in \{1, 2, ..., m\}. \end{aligned} \tag{1.9}$$

En caso de que el dual sea no acotado, en (1.9) existirá un vector v tal que:

$$(b - D\bar{y})^t v^j > 0. \tag{1.10}$$

Que v sea una dirección extrema dice que nos podemos mover indefinidamente en esa dirección mejorando siempre la función objetivo del problema dual. Por lo tanto, para evitar problemas de factibilidad en (1.3) y reconducir el algoritmo hacía soluciones factibles se debe identificar la dirección extrema con el objetivo de que no aparezca involucrada en las iteraciones siguientes. Para ello, se tiene que asegurar que $(b - D\bar{y})^t v^j$ sea menor o igual a cero en las futuras iteraciones. Geométricamente, tenemos que

encontrar valores para el vector v tal que $(b - D\bar{y})^t v^j$ forme un ángulo de menos de 90^o con alguna dirección extrema del conjunto.

Para toda dirección extrema, v^j con $j \in J$, se definirá el siguiente corte de factibilidad:

$$v^t(b - D\bar{y}) \leq 0. \tag{1.11}$$

Para la obtención de una dirección extrema de la región factible de (1.4), se podrá resolver el siguiente problema de optimización:

$$\begin{aligned} \text{máx} \quad & v^t(b - D\bar{y}) \\ \text{s.a.} \quad & A^t v \leq 0 \\ & e^t v \leq M \\ & v \geq 0, \end{aligned} \tag{1.12}$$

donde $e = (1, ..., 1) \in \mathbb{R}^n$ y M es un valor suficientemente grande tal que $u_i \in \{v \in \mathbb{R}^n : e^t v \leq M\}, \forall i = 1, ..., n$. Este problema siempre va a ser factible.

1.1.3. Problema maestro de Benders

Ahora ya se tienen todos los ingredientes necesarios para la construcción del problema maestro. Partiendo del problema original:

$$\begin{aligned} \text{mín} \quad & c^t x + d^t y \\ \text{s.a.} \quad & Ax + Dy \geq b \\ & x \geq 0, y \in \mathrm{S} \subset \mathbb{Z}^p. \end{aligned} \tag{1.13}$$

Por la dualidad débil, las relaciones entre el problema primal y dual así como por el teorema de Carathéodory sabemos que el valor óptimo $\bar{y}$ de y en el problema original se podrá obtener mediante el siguiente problema:

$$\begin{aligned} \text{mín} \quad & \max_{\bar{u} \in I} \quad \bar{u}(b - Dy) + d^t y \\ \text{s.a.} \quad & \bar{v}(b - Dy) \leq 0, \forall \bar{v} \in J \\ & y \in \mathrm{S} \subset \mathbb{Z}^p, \end{aligned} \tag{1.14}$$

o equivalentemente,

$$\begin{aligned} \text{mín} \quad & \alpha \\ \text{s.a.} \quad & \alpha \geq d^t y + u^i(b - Dy), \quad i \in \mathrm{I} \\ & v^j(b - Dy) \leq 0, \quad j \in \mathrm{J} \\ & y \in \mathrm{S} \subset \mathbb{Z}^p, \end{aligned} \tag{1.15}$$

donde I representa el conjunto de los puntos extremos y J el conjunto de direcciones extremas. Esta última será la formulación del problema maestro en el algoritmo de Benders encargado de proporcionar propuestas para los valores que deben tomar las variables complicantes. Además, por la dualidad débil se puede observar que el corte de optimalidad acota inferiormente el problema original con lo que las soluciones del problema maestro son una cota inferior del problema (1.2).

1.2. Algoritmo de Benders

Con lo visto hasta el momento se puede ver que el subproblema es un problema más restringido que el problema original lo que implicada que sus soluciones óptimas serán cotas superiores para dicho problema. Por su parte, el problema maestro se trata de una relajación del problema original con lo que sus soluciones óptimas se corresponden a cotas inferiores del mismo. Lo que hace el algoritmo en la práctica es de forma iterativa conseguir que las cotas de ambos problemas converjan al mismo punto, siendo este punto la solución óptima del problema original. Cabe destacar que esto último es cierto solo si estamos trabajando con problemas de programación matemática convexos. Si embargo, como veremos más adelante una generalización del método de Benders puede ser aplicado sobre problemas no convexos.

A continuación, en el Algoritmo 1 se muestra el pseudocódigo del método de descomposición de Benders.

Algoritmo 1: Método de descomposición de Benders.

Paso *0*: Se inicializa $\bar{y} \in \mathrm{S} \subset \mathbb{Z}^p$ y $\epsilon \geq 0$ se establecen las cotas superiores e inferiores a $UB = \infty$ y $LB = -\infty$. Además, $k = 0$. Se inicializan $I = J = \emptyset$.

Paso *1*: Resuelve el subproblema de Benders. Actualiza $k = k + 1$.

$$\begin{aligned} \text{mín} \quad & c^t x + d^t \bar{y} \\ \text{s.a.} \quad & Ax \geq b - D\bar{y} \\ & x \geq 0 \end{aligned} \tag{1.16}$$

si *El subproblema y su dual son infactibles* **entonces**
  Termina.
si no, si *El subproblema tiene solución óptima* **entonces**
  $UB = \text{mín}(UB, c^t \bar{x} + d^t \bar{y})$;
  se genera el corte de optimalidad de Benders $\alpha \geq d^t y + u^k(b - Dy)$ y se añade k al conjunto I del maestro;
si no, si *El subproblema es infactible y su dual es no acotado* **entonces**
  resuelve (1.11) y genera el corte de factibilidad de Benders $v^k(b - Dy) \leq 0$ y se añade k al conjunto J del maestro;

Paso *2*: Se resuelve el problema maestro;

$$\begin{aligned} \text{mín} \quad & \alpha \\ \text{s.a.} \quad & \alpha \geq d^t y + u^i(b - Dy), \quad i \in \mathrm{I} \\ & v^j(b - Dy) \leq 0, \quad j \in \mathrm{J} \\ & y \in \mathrm{S} \subset \mathbb{R}^p. \end{aligned} \tag{1.17}$$

obteniendo nuevos valores para $\bar{y}$ y una solución para $\alpha = \alpha^*$;
Se actualiza la cota inferior $LB = \text{máx}(LB, \alpha^*)$.
si $UB - LB \leq \epsilon$ **entonces**
  Termina;
  en otro caso se vuelve a **Paso** *1*.

Es importante destacar que en el caso de que en el problema original estén implicadas restricciones en las que únicamente estén involucradas variables complicantes estas restricciones se trasladarán al problema maestro. Esto se hace porque las variables complicantes en el subproblema original están fijas mientras que es el maestro el que propone soluciones para dichas variables.

También, es relevante el hecho de como encontrar una solución inicial para las variables complicantes en el algoritmo. En la práctica una sencilla forma de hacerlo es resolver en la inicialización del mismo el problema maestro fijando el valor de α para evitar que el problema sea no acotado. Otra forma es establecer cotas muy grandes, tanto inferiores como superiores para α y resolver el problema.

Un detalle importante de cara a la implementación del mismo es que aunque para la obtención de los puntos extremos y direcciones extremas teóricamente se tengan que resolver los problemas duales definidos a lo largo de este capítulo según sea necesario, en la práctica se puede hacer de otra forma que es más eficiente desde el punto de vista computacional. Los *solvers* como CBC (Forrest et al. 2018) o Gurobi (Gurobi Optimization 2019) devuelven la información del dual asociada al problema primal. Entonces, lo único que debemos tener en cuenta a la hora de construir los cortes es que determinar si el subproblema es factible o no. Si el subproblema es factible le pedimos la información dual al solver para construir el corte de optimalidad y si es infactible se resuelve el problema (1.11) para construir el corte de factibilidad.

Para finalizar esta sección, se comentarán algunos aspectos relacionados con la convergencia del algoritmo y, para ello, se debe tener en cuenta el proceso de generación de cortes tanto de optimalidad como de factibilidad. En cada iteración del algoritmo se introduce un corte y cada corte está asociado a un punto extremo o a una dirección extrema. Como la región factible del problema (1.4) no depende de las variables complicantes quiere decir que el espacio de las soluciones factibles de dicho problema es siempre el mismo. Entonces, si se generan todos los posibles cortes, el problema maestro y el problema original serán problemas equivalentes ya que dichos cortes definen todos los posibles puntos extremos y direcciones extremas del problema original. Como el conjunto de puntos extremos y direcciones extremas del problema (1.4) es finito, esto nos asegura que el algoritmo converge en una cantidad finita de iteraciones. Además, cabe decir que si un punto extremo se repite el algoritmo termina encontrado el óptimo del problema y las direcciones extremas nunca se repetirán entre dos iteraciones. En el peor de los casos habrá que generar todos los cortes posibles para que el algoritmo converja pero en la práctica esto no ocurre.

1.3. Ejemplo numérico del algoritmo

Supongamos que se quiere resolver el siguiente problema de programación lineal entero mixto:

$$\begin{aligned} \text{mín} \quad & x_1 + x_3 + y \\ \text{s.a.} \quad & x_1 - 6x_2 - 5x_3 + 2y \geq 1 \\ & -x_1 + x_2 + 2x_3 + 3y \geq 2 \\ & x_1, x_2, x_3 \geq 0 \\ & y \geq 0, y \in \mathbb{Z}. \end{aligned} \tag{1.18}$$

Se puede ver fácilmente que si la variable y no estuviera implicada en el modelo se tendría un problema de programación lineal. Además, considerar esta variable como la complicante nos permite separa el problema original en dos problemas diferentes con lo que se podrá aplicar el algoritmo de Benders para resolverlo. El subproblema tendrá la siguiente forma:

$$\begin{aligned} \text{mín} \quad & x_1 + x_3 + \bar{y} \\ \text{s.a.} \quad & x_1 - 6x_2 - 5x_3 + 2\bar{y} \geq 1 \\ & -x_1 + x_2 + 2x_3 + 3\bar{y} \geq 2 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \tag{1.19}$$

donde $\bar{y}$ es un valor fijo de y, con lo que el subproblema solo depende de las variables x_1, x_2 y x_3. El

problema maestro inicial estará definido por:

$$\begin{aligned} \text{mín} \quad & \alpha + y \\ \text{s.a.} \quad & y \geq 0, y \in \mathbb{Z}, \end{aligned} \tag{1.20}$$

donde $\alpha \in (-\infty, \infty)$.

- **Paso 0**. Se inicializa $\bar{y}^0 = 0$, ϵ es un valor muy pequeño, se establece $UB = \infty$, $LB = -\infty$. Además, $k = 0$ y $I = J = \emptyset$

- **Paso 1**. Se resuelve el subproblema y se actualiza $k = 1$.

$$\begin{aligned} \text{mín} \quad & x_1 + x_3 + 0 \\ \text{s.a.} \quad & x_1 - 6x_2 - 5x_3 + 0 \geq 1 \\ & -x_1 + x_2 + 2x_3 + 0 \geq 2 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \tag{1.21}$$

  obteniendo que la solución de este problema con $\bar{y} = 0$ es infactible. Como el subproblema es infactible resolvemos se resuelve (1.11). En concreto,

$$\begin{aligned} \text{mín} \quad & v_1 + 2v_2 \\ \text{s.a.} \quad & v_1 - v_2 \leq 0 \\ & -6v_1 + v_2 \leq 0 \\ & -5v_1 + 2v_2 \leq 0 \\ & v_1 + v_2 \leq M \\ & v_1, v_2 \geq 0, \end{aligned} \tag{1.22}$$

  cuya solución se alcanza para $v_1 = 1$ y $v_2 = 5/2$. Se genera el corte de factibilidad $12 - 19y \leq 0$.

- **Paso 2**. Se resuelve el problema maestro;

$$\begin{aligned} \text{mín} \quad & \alpha + y \\ \text{s.a.} \quad & 12 - 19y \leq 0 \\ & y \geq 0, y \in \mathbb{Z}, \end{aligned} \tag{1.23}$$

  obteniendo $y^1 = 1$ y $\alpha = -\infty$. Se actualiza $LB = \text{máx}(-\infty, -\infty) = -\infty$. Como $UB - LB > \epsilon$ se vuelve al **Paso 1**.

- **Paso 1**. Se resuelve el subproblema y se actualiza $k = 2$.

$$\begin{aligned} \text{mín} \quad & x_1 + x_3 + 1 \\ \text{s.a.} \quad & x_1 - 6x_2 - 5x_3 + 1 \geq 1 \\ & -x_1 + x_2 + 2x_3 + 1 \geq 2 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \tag{1.24}$$

  obteniendo que la solución de este problema con $\bar{y} = 1$ es factible. La solución se alcanza en $x_1 = 0$, $x_2 = 0$ y $x_3 = 0$. Por su parte, pidiéndole la información al *solver*, los duales asociados a la primera y segunda restricción toman los valores $u_1 = 0$ y $u_2 = 0$, respectivamente. Se actualiza $UB = \text{mín}(\infty, 1)$ Se genera el corte de optimalidad $\alpha \geq 0$.

- **Paso 2**. Se resuelve el problema Maestro;

$$\begin{aligned} \text{mín} \quad & \alpha + y \\ \text{s.a.} \quad & 12 - 19y \leq 0 \\ & \alpha \geq 0 \\ & y \geq 0, y \in \mathbb{Z}, \end{aligned} \tag{1.25}$$

obteniendo $y^2 = 1$ y $\alpha = 0$. Se actualiza $LB = max(-\infty, 1) = 1$. Como $UB = LB$ el algoritmo termina.

El valor óptimo de la variable complicante será $y^* = 1$. Resolviendo nuevamente el subproblema se obtienen los valores alcanzados por el resto de variables en el óptimo del problema original que son $x_1 = 0$, $x_3 = 0$ y $0 \leq x_2 \leq 1/6$. Por último, el valor de la función objetivo en el óptimo será 1, coincidiendo con $UB = LB = 1$.

Capítulo 2

Problemas de Pooling

Los problemas de pooling son un tipo de problemas de programación matemática que sirven para modelar diversas situaciones que se dan en plantas de procesamiento industrial y han adquirido una gran relevancia en los últimos cuarenta años (Alfaki & Haugland 2014). En concreto, industrias como la petroquímica o empresas dedicadas al procesamiento y distribución de gas utilizan dicho tipo de modelos para la optimización de sus plantas de procesamiento industrial. En este capítulo se explicará en que consisten estos modelos y se presentarán dos formulaciones equivalentes del mismo.

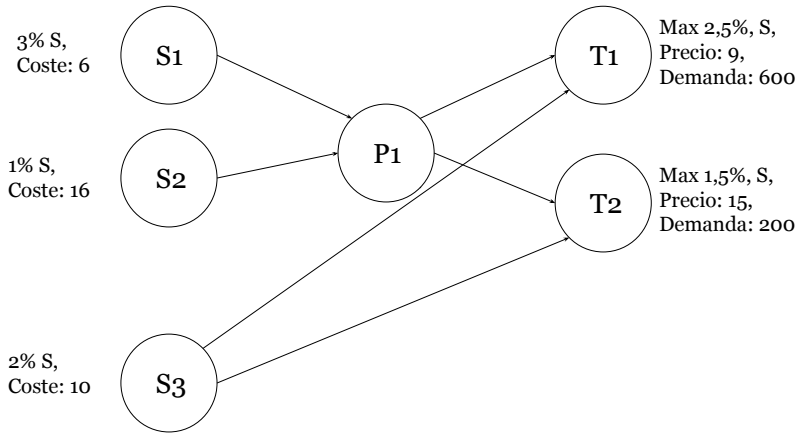
El alto valor añadido que generan estos modelos en las industrias comentadas ha provocado que se estudiaran ampliamente tanto desde el mundo académico como empresarial. La razón principal es que se tratan de modelos no lineales y no convexos y, es sabido, que tienen multitud de óptimos locales. Además, desde la óptica de la complejidad computacional estos modelos están catalogados como NP-duros demostrado en Alfaki & Haugland (2013) y en Dai et al. (2018). Por lo tanto, encontrar el óptimo global de este tipo de problemas es muy difícil o computacionalmente prohibitivo cuando el tamaño del problema es considerable. Es por ello, que las dos líneas de investigación principales en este campo sean, por un lado, el desarrollo de formulaciones equivalentes del mismo que ayuden a encontrar mejores soluciones con los *solvers* existentes en el mercado y, por otro lado, la búsqueda y desarrollo de nuevos métodos para resolverlos. Sobre la primera línea de investigación se puede consultar más información de la que se presentan en este capítulo en Alfaki (2012), Alfaki & Haugland (2013), Haverly (1978), Sahinidis & Tawarmalani (2005) y Floudas & Aggarwal (1990) entre otros. En cuanto a los métodos de resolución podemos clasificarlos en dos grandes campos: algoritmos cuyo objetivo es la optimalidad global de dichos problemas y los que tratan de encontrar mejores óptimos locales para los mismos. Dentro de los primeros destaca Alfaki & Haugland (2013) que propone un método para optimalidad global comparando los resultados con los de la herramienta de optimización global BARON (Tawarmalani & Sahinidis 2005). El problema de tratar de encontrar el óptimo global en los problemas de pooling es que si el número de variables y restricciones es elevado, entorno a los miles, el tiempo de ejecución de este tipo de algoritmos puede ser excesivo. Es por ello que muchos autores e incluso empresas del sector no tratan de buscar la optimalidad global sino que tratan de buscar métodos que consigan llegar a mejores soluciones locales del problema. El hecho de encontrar mejores soluciones locales puede implicar un aumento de los beneficios muy considerable en este tipo de industrias. Ejemplos de algoritmos para mejorar la obtención de óptimos locales en estos problemas se pueden consultar en Alfaki & Haugland (2014) donde se propone una heurística especialmente diseñada para los problemas de pooling que consigue buenos óptimos locales, en Dai et al. (2018) se utiliza programación lineal sucesiva (SLP) y una versión mejorada del algoritmo para resolverlos, en Floudas & Aggarwal (1990) se propone un algoritmo basado en Benders que será el utilizado en este trabajo para resolver los problemas de pooling.

Antes de presentar las dos formulaciones utilizadas para los problemas de pooling vamos explicar en que consisten este tipo de problemas. Supongamos que el modelo de pooling representa una refinería de petróleo. De una manera sencilla vamos a comentar el proceso que se lleva a cabo en ella. Primero se compra el crudo que será almacenado en depósitos inicialmente conocidos como fuentes. Este crudo es transportado a través de tuberías por la refinería hasta llegar a los pools o tanques. En estas unidades se mezclan los diferentes productos para conseguir productos intermedios o finales como la gasolina, el gasoil, el queroseno, etc. Dichos productos son nuevamente transportados por las tuberías correspondientes hasta llegar a las terminales en las que se pueden volver a mezclar diferentes productos intermedios para conseguir productos finales o simplemente se almacenan en ellas para atender la demanda de los consumidores de estos producto finales. Cabe destacar que el objetivo final es la maximización de los beneficios de la refinería minimizando los costes operacionales de la planta de procesado teniendo en cuenta las restricciones impuestas por la propia operación de la refinería y atendiendo la demanda de los consumidores.

En un sentido matemático, el proceso anterior, puede verse como una red de flujo que no es más que un grafo orientado con números asociados a los arcos o nodos y en el que por las aristas se transportan o viajan ciertas unidades de productos (Ver Anexo A). En concreto, el problema de pooling es una red de flujo con tres capas de nodos. La primera capa recibe el nombre de fuente y es por la que entran los productos iniciales (crudos), la segunda capa serán los pool o tanques en las que se mezcla productos con diferentes propiedades para transformarlos en nuevos productos y, la tercera capa, son las terminales, en las que se le exige que los nuevos productos cumplan ciertas calidades impuestas por el mercado. El proceso de mezcla de los productos solo se produce en los tanques y en las terminales siendo los causantes de no convexidades en el modelo. Es decir, el proceso llevado a cabo en los pools, matemáticamente, se traduce en restricciones bilineales provocando que se pierda la propiedad de la convexidad en la región factible del modelo. En caso de que los pool o tanques no estuvieran presentes en el modelo se tendrían modelos lineales para los que existen algoritmos muy eficientes.

Para entender mejor lo explicado previamente expondremos, a continuación, uno de los problemas más famosos en el campo de los problemas de pooling, representando en la Figura 2.1 y que fue definido en Haverly (1978).

Figura 2.1: Problema de Pooling de Haverly.



En el ejemplo de Haverly tenemos definida una red con flujo con tres capas de nodos. En la primera capa tenemos las fuentes que en este caso están formadas por tres tipos diferentes de crudos representados por S_1, S_2 y S_3. Cada uno de los crudos tiene asociado una concentración de sulfuro "S" diferentes 3 %, 1 % y 2 % respectivamente. A modo aclaratorio las concentraciones de propiedades como el sulfuro se conocen como calidades en la literatura sobre problemas de pooling. Cada uno de los crudos tiene un coste asociado como se puede ver en la Figura 2.1. Los crudos S_1 y S_2 son enviados al pool donde serán mezclados mientras que el crudo S_3 es enviado directamente a las terminales T_1 y T_2. Una vez los crudos se han mezclado en el nodo intermedio se envía a las terminales. En las terminales se impone que los nuevos productos resultantes de la mezcla en el pool y de la mezcla de los productos en las terminales cumpla con ciertas calidades. En este caso que la concentración de sulfuro no exceda al $2,5$ % en T_1 y $1,5$ % en T_2. Además, la terminal T_1 tiene tiene que atender una demanda de 600 y la terminal T_2 de 200. Como se puede ver los tres crudos iniciales son procesados para la obtención de dos productos finales distintos. Con lo que el problema del pooling consiste operar la hipotética refinería minimizando los costes de producción cumpliendo con las calidades exigidas por los consumidores y con la demanda de los mismos.

2.1. Formulaciones de los problemas de pooling

Antes de presentar los modelos se van a hacer una serie de aclaraciones. Como se ha comentado al principio de este trabajo, el objetivo principal del mismo ha sido la implementación de un algoritmo basado en Benders que permite obtener buenos óptimos locales en tiempos razonables de ejecución. Inicialmente se disponían de tres formulaciones equivalentes para los problemas de pooling PQ,TP y STP obtenidas de `http://www.ii.uib.no/~mohammeda/spooling/` que son las presentadas por el mismo autor en Alfaki & Haugland (2013). Cabe decir que las instancias están escritas en el lenguaje de modelado matemático GAMS y ha sido necesario traducirlas a Pyomo (Hart et al. 2017) ya que es el lenguaje de modelización basado en Python que se ha utilizado. En el presente documento solo se presentan las formulaciones PQ y TP ya que la STP no daba buenos resultados con el algoritmo implementado. Es sabido que las diferentes formulaciones equivalentes de los problemas de pooling tienen diferente tipo de comportamiento en el proceso de búsqueda de soluciones óptimas en función del algoritmo que se emplee para resolverlas. En cuanto a la notación, para que sea más cómodo para el lector en caso de que quiera profundizar más sobre el modelado de este tipo de problemas, se ha tomado la licencia de seguir la notación utilizada en Alfaki & Haugland (2013).

A continuación se presenta la notación utilizada. x es un vector de números reales no negativos definidos en el conjunto S, es decir $x \in \mathbb{R}_+^{\mathrm{S}}$. $D = (N, A)$ es el grafo orientado que representa la estructura de red en los problemas de pooling donde N es el conjunto de nodos y A es el conjunto de arcos. S, I y T son tres conjuntos disjuntos con $N = S \cup I \cup T$ donde S representan el conjunto de fuentes, I el conjunto de pools o tanques y T el conjunto de terminales Para cada nodo $i \in N$ se tiene que $N_i^+ = \{j \in N : (i, j) \in A\}$ y $N_i^- = \{j \in N : (j, i) \in A\}$, entonces N_i^+ representan los arcos que salen del nodo i y N_i^- los que llegan a dicho nodo. Se asume que S y T son conjuntos no vacíos y que $A \subseteq (S \times I) \cup (S \times T) \cup (I \times T)$ implicando que $N_s^- = \emptyset, \forall s \in S$ y $N_t^+ = \emptyset, \forall t \in T$. Es decir, no puede existir ningún arco que suministre flujo a las fuentes y no existe ningún arco que salga de las terminales. $P \subseteq S \times I \times T$ es el conjunto de caminos con dos arcos en D. Los atributos de calidad están definidos en el conjunto K. Para cada $k \in K$ existe una constante para cada fuente y para cada terminal. q_s^k es el parámetro de calidad del atributo k en la fuente $s \in S$ y q_t^k es su análogo para cada terminal $t \in T$. Para cada $i \in N$ se define una constante, capacidad de flujo b_i que representa la cota máxima de flujo que puede pasar por los diferentes elementos de N. Para cada arco $(i, j) \in A$ se define la constante coste unitario c_{ij} que representa un coste si la materia prima está implicada en s y precio de venta de los productos finales en t.

Tal y como se ha comentado con anterioridad en el caso de que no existieran los pool o tanques en los problemas de pooling implicaría que dichos modelos fuesen lineales. En concreto, podrían representarse

como un problema de *flujo en redes con coste mínimo.*

Definición 2.2. *Dada una red con flujo, el problema de* flujo en redes con coste mínimo *consiste en determinar el flujo que ha de pasar por cada arco de tal manera que el coste asociado sea mínimo y que se cumplan las restricciones de conservación de flujo y las impuestas por las capacidades.*

En ese caso los flujos serían las variables de decisión del problema de optimización que consistiría en elegir los flujos, cumpliendo las restricciones impuestas que minimicen el coste total del flujo que pasa por la red. Siguiendo con la notación de Alfaki & Haugland (2013) se puede definir el politopo del flujo $F(D, b)$ como el conjunto de los $f \in \mathbb{R}_+^A$ tal que:

$$\sum_{j \in N_s^+} f_{sj} \leq b_s, \qquad s \in S, \tag{2.1}$$

$$\sum_{j \in N_i^-} f_{ji} \leq b_i, \qquad i \in N \setminus S, \tag{2.2}$$

$$\sum_{j \in N_s^+} f_{sj} - \sum_{j \in N_i^-} f_{ji} = 0, \quad i \in I. \tag{2.3}$$

La ecuación (2.1) indica que la suma de la cantidad de flujo que sale de cada una de las fuentes tiene que ser menor que b_s (máxima cantidad de flujo de cada fuente), la ecuación (2.2) refleja que la cantidad de flujo que sale de cualquier nodo distinto de las fuentes (terminales o pools) tiene que ser menor o igual que la cantidad de flujo máximo soportado por cada una esas unidades. La última ecuación se conoce como conservación de flujo e indica que la cantidad de flujo que entra en cada uno de los pool tiene que ser igual a la que sale de ellos.

2.2.1. Formulación PQ del problema

En la literatura las formulaciones Q de los problemas de pooling se caracterizan por utilizar parámetros para las calidades de los productos y materias primas en vez variables. Además, la formulación PQ se caracteriza principalmente por que se definen variables que indican proporciones de flujo de las fuentes a los tanques y unidades de flujo de los tanques a las terminales.

Antes de presentar las ecuaciones que definen el modelo de optimización se van a presentar una serie de consideraciones. Si un arco no está definido en A su flujo asociado será nulo o lo que es lo mismo $f_{ij} = 0$ si $i, j \in N$ y $(i, j) \notin A$. Es decir, se tienen dos nodos el i y el j definidos en la red pero no hay una conexión (arco) que los una. Se entenderá por variable de proporción y_i^s a la proporción de flujo que va al pool $i \in I$ y que se ha originado en la fuente $s \in S$. Al igual que antes $y_i^s = 0$ si no existe un arco que una la fuente s y el pool i. La proporción de flujo se calcula como $y_i^s = f_{si} / \sum_{t \in T} f_{it}$ donde f_{si} es el flujo que va de la fuente s al pool i y, $\sum_{t \in T} f_{it}$ representa la cantidad total de flujo que sale del pool i y llega a todas las terminales t de T. Para todos los caminos $(s, i, t) \in P$, se define x_{sit} como el flujo de todo el camino y nuevamente el flujo de ese camino será nulo si no existe ese camino $(x_{sit} = 0, \forall (s, i, t) \in (S, I, T) \setminus P)$. Llegados a este punto ya se puede presentar la formulación PQ del problema de pooling:

$$\min_{f,y,x} \quad \sum_{s \in S} \sum_{t \in T} \Big(c_{st} f_{st} + \sum_{i \in I} (c_{si} + c_{it}) x_{sit} \Big) \tag{2.4}$$

$$\text{s.a.} \quad \sum_{t \in T} \Big(f_{st} + \sum_{i \in I} x_{sit} \Big) \leq b_s, \qquad s \in S, \tag{2.5}$$

$$\sum_{s \in S} \sum_{t \in T} x_{sit} \leq b_i, \qquad i \in I, \tag{2.6}$$

$$\sum_{s\in S}\Big(f_{st}+\sum_{i\in I}x_{sit}\Big)\leq b_t,\qquad t\in T, \tag{2.7}$$

$$\sum_{s\in S}(q_s^k-q_t^k)\Big(f_{st}+\sum_{i\in I}x_{sit}\Big)\leq 0,\qquad t\in T,k\in K, \tag{2.8}$$

$$\sum_{s\in S}y_i^s=1,\qquad i\in I, \tag{2.9}$$

$$\sum_{s\in S}x_{sit}=f_{it},\qquad i\in I,t\in T, \tag{2.10}$$

$$\sum_{t\in T}x_{sit}\leq b_i y_i^s,\qquad s\in S,i\in I, \tag{2.11}$$

$$x_{sit}=y_i^s f_{it},\qquad (s,i,t)\in P, \tag{2.12}$$

$$f_{it}\geq 0,\qquad i\in S\cup I,t\in T, \tag{2.13}$$

$$0\leq y_i^s\leq 1,\qquad i\in I,s\in S. \tag{2.14}$$

Como se puede ver el problema de optimización consiste minimizar los costes donde, en economía, es análogo a maximizar beneficios teniendo en cuenta todas las restricciones impuestas por la estructura de red de los problemas de pooling. En la función objetivo aparece el parámetro c, representará un precio de compra (si es negativo) y de venta (si es positivo), con lo que se trata de minimizar los costes asociados a todos los flujos que van desde las fuentes a las terminales. La restricciones (2.5), (2.6) y (2.7) son restricciones de capacidad. La primera indica que el flujo total de las terminales tiene que ser menor o igual al suministrado por cada una de las fuentes; la segunda que el flujo del camino que pasa por el pool i tiene que ser menor o igual que la capacidad máxima soportada por el mismo y, la tercera; que el flujo que llega a cada una de las terminales tiene que ser menor o igual a la capacidad máxima de dicha terminal.

La inecuación (2.8) tiene que ver con las cotas para las calidades en las terminales, indicando que la calidad asociada a cada uno de los productos finales tiene que ser menor o igual a la que tenían las materias primas empleadas para obtenerlo. Es decir, pensando en el ejemplo de Haverly si cierto crudo tiene una concentración de sulfuro, por ejemplo, una vez el crudo ha sido procesado y mezclado en los pool con el fin de transformarlo en otro producto distinto, este último no puede tener una mayor concentración de sulfuro. La ecuación (2.9) representa la restricción lógica de que la suma de todas las proporciones de flujo que salen de la fuente s al pool i tiene que sumar uno. Las restricciones (2.10) y (2.11) son redundantes pero mejoran el comportamiento del modelo a la hora de encontrar soluciones óptimas para el mismo (Sahinidis & Tawarmalani 2005). Su interpretación, respectivamente, es que el flujo del arco (i,t) tiene que ser igual el flujo suministrado por la fuente para ese arco representado por el flujo de ese camino, mientras que el camino (i,s,t) no puede consumir más que una proporción y_i^s de la capacidad mínima del pool i. El término bilineal aparece en la restricción (2.12) indicando que la proporción de flujo que va de s a i por el flujo que va de los pool a las terminales tiene que ser igual al flujo total de ese camino. Las dos últimas ecuaciones indican que los flujos nunca pueden ser negativos y que la proporción de flujo que va de las fuentes a los pools tiene que estar comprendida entre 0 y 1 dado que es una proporción.

2.2.2. Formulación TP del problema

La formulación TP del problema de pooling es simétrica a la formulación PQ. La diferencia con la PQ es que ahora las variables que indican proporciones de flujo se definen sobre las terminales en vez de sobre las fuentes y las variables de flujos se definen sobre los arcos que van desde las fuentes a los pools.

Para todos los pools $i \in I$ se tiene que y_i^t es la proporción de flujo que va desde el pool i a las terminal $t \in T$. Entonces el cálculo de las variables que representan proporciones de flujo es $y_i^t = f_{it} / \sum_{s \in S} f_{si}$. Por lo tanto, la formulación TP del problema es la siguiente:

$$\min_{f,y,x} \quad \sum_{s \in S} \sum_{t \in T} \Big(c_{st} f_{st} + \sum_{i \in I} (c_{si} + c_{it}) x_{sit} \Big) \tag{2.15}$$

$$\text{s.a.} \quad \sum_{t \in T} \Big(f_{st} + \sum_{i \in I} x_{sit} \Big) \leq b_s, \qquad s \in S, \tag{2.16}$$

$$\sum_{s \in S} \sum_{t \in T} x_{sit} \leq b_i, \qquad i \in I, \tag{2.17}$$

$$\sum_{s \in S} \Big(f_{st} + \sum_{i \in I} x_{sit} \Big) \leq b_t, \qquad t \in T, \tag{2.18}$$

$$\sum_{s \in S} (q_s^k - q_t^k) \Big(f_{st} + \sum_{i \in I} x_{sit} \Big) \leq 0, \qquad t \in T, k \in K, \tag{2.19}$$

$$\sum_{t \in T} y_i^t = 1, \qquad i \in I, \tag{2.20}$$

$$\sum_{t \in T} x_{sit} = f_{it}, \qquad i \in I, s \in S, \tag{2.21}$$

$$\sum_{s \in S} x_{sit} \leq b_i y_i^t, \qquad i \in I, t \in T, \tag{2.22}$$

$$x_{sit} = y_i^t f_{si}, \qquad (s,i,t) \in P, \tag{2.23}$$

$$f_{si} \geq 0, \qquad s \in S, i \in U \cup T, \tag{2.24}$$

$$0 \leq y_i^t \leq 1, \qquad i \in I, t \in T. \tag{2.25}$$

Dado que el único cambio en este modelo son las proporciones de flujo implica que las variables que representan los flujos la función objetivo son las mismas. En este punto se omite una explicación tan detallada sobre las restricciones ya que su interpretación es análoga a la del modelo anterior.

Lo importante y el por qué de tener estas dos formulaciones es que pese a ser modelos equivalentes para los problemas de pooling computacionalmente presentan comportamientos diferentes. En este sentido, en los resultados empíricos obtenidos y presentados en el capítulo cuarto, con el algoritmo implementado, se verán reflejadas las diferencias de comportamiento entre ambos modelos. Este es un hecho importante a resaltar ya que, como se ha comentado previamente, en la práctica, este tipo de problemas no se resuelve a optimalidad global debido a que el tiempo computacional para instancias reales o con un número levado de variables y restricciones puede ser prohibitivo. En la práctica lo que buscan las empresas de las industrias donde se utilizan este tipo de modelos es tratar de encontrar buenas soluciones locales para dicho problema. Que el comportamiento de ambas formulaciones equivalentes sea diferente para los algoritmos sugiere que muchas empresa del sector pueden mejorar sus resultados económicos formulando su modelo particular de una o de otra forma.

Capítulo 3

Método de descomposición de Benders para los problemas de pooling

En el capítulo anterior sobre los aspectos generales de los problemas de pooling quedó de manifiesto que las operaciones realizadas en los tanques o pools provocan que el modelo sea no lineal y no convexo. En concreto, aparecen restricciones bilineales del tipo $x_{sit} = y_i^t f_{si}$. Para resolver este tipo de problemas existen técnicas como la programación lineal sucesiva (SLP) (Palacios-Gomez et al. 1982), algoritmos de punto interior (Wächter & Biegler 2005), métodos de penalización (Bazaraa et al. 2005), lagrangiano aumentado (Birgin et al. 2010), etc. El problema principal de estos métodos es que tienen que lidiar con el problema completo y dado que los problemas de pooling están catalogados como problemas difíciles de resolver, métodos que se aprovechen de la estructura del problema pueden ser de gran ayuda a la hora de conseguir buenos óptimos locales para el problema. Es por ello que en Floudas & Aggarwal (1990) se propone el método de Benders generalizado adaptado a los problemas de pooling y será el método principal de este trabajo. A modo aclaratorio, el método sirve para resolver cualquier tipo de problemas de optimización no lineales sobre los que se puedan detectar variables complicantes o cualquier tipo de estructura de separación que permita dividir el problema en dos: subproblema y maestro.

En general, tener problemas no lineales tiene una serie de implicaciones:

- no podemos hablar de puntos extremos,
- tampoco se puede hablar de direcciones extremas,
- la región factible puede no estar definida por un poliedro.

Además, si la región factible es no convexa como en el caso de los problemas de pooling, también se pierde la propiedad de que todo óptimo local es un óptimo global del problema, lo cual facilitaría el proceso de búsqueda de soluciones óptimas. Además, es sabido que los problemas de pooling presentan múltiples óptimos locales causados por la no convexidad de la región factible. Es por ello que en esta sección se presentará un algoritmo basado en Benders que en la práctica funciona realmente bien a la hora de buscar buenos óptimos locales para los problemas de pooling siendo este nuestro objetivo principal. Como las justificaciones teóricas presentadas en el primer capítulo ya no son del todo válidas bajo el supuesto de no linealidad, antes de presentar la extrapolación del algoritmo de Benders al mundo

no lineal se van a presentar una serie de resultados y proposiciones que serán de ayuda para el lector.

Dado el siguiente problema de optimización:

$$\begin{array}{lll} \text{mín} & f(x,y) & \\ \text{s.a.} & g_i(x,y) \leq 0 & i = 1,...,m \\ & h_j(x,y) = 0 & j = 1,...,l \\ & (x,y) \in S, & \end{array} \tag{3.1}$$

donde $S = X \times Y$, siendo X, Y los conjuntos de restricciones fáciles que involucran únicamente las variables x e y, respectivamente (en estos conjuntos estarían consideradas las restricciones de cota de las variables). Además, dado $(x,y) \in S$, denotamos por $R(x,y) = \{i : g_i(x,y) = 0\}$, el conjunto de las restricciones activas o saturadas en (x,y).

Las condiciones de optimalidad en problemas no lineales son diferentes a las condiciones de optimalidad bajo supuestos de linealidad. Ahora ya no podemos buscar el punto extremo de la región factible que consiga el mejor valor para la función objetivo ni tampoco (en caso de no ser acotada la región factible) aquella dirección extrema en la que moviéndose en su dirección la función objetivo no deja de mejorar. A continuación, se presenta un teorema que define las condiciones necesarias para que un punto (x,y) se considere óptimo así como bajo que supuestos se cumplen dichas condiciones.

Teorema 2. *(Condiciones necesarias de Karush-Kuhn-Tucker). Dado el problema de optimización (3.1). Supongamos que (x^*, y^*) es un punto factible tal que*

- *las funciones f y g_i con $i \in R(x^*, y^*)$ son diferenciables en (x^*, y^*),*
- *las funciones g_i con $i \notin R(x^*, y^*)$ son continuas en (x^*, y^*),*
- *las funciones $h_j(x,y)$ con $j \in \{1,...,l\}$ son continuamente diferenciables en (x^*, y^*).*
- *los vectores $\nabla g_i(x^*, y^*)$, con $i \in R(x^*, y^*)$, y $\nabla h_j(x^*, y^*)$ con $j \in \{1,...,l\}$ son linealmente independientes.*

Si (x^*, y^*) es un óptimo local, entonces existen escalares únicos u_i para todo $i \in R(x^*, y^*)$ y v_j para todo $j \in \{1,...,l\}$ tales que:

$$\begin{aligned} \nabla f(x^*, y^*) + \sum_{i \in R(x^*, y^*)} u_i \nabla g_i(x^*, y^*) + \sum_{j=1}^{l} v_j \nabla h_j(x^*, y^*) = 0 \\ u_i \geq 0 \qquad \forall i \in R(x^*, y^*). \end{aligned} \tag{3.2}$$

Además, si las funciones g_i con $i \notin R(x^*, y^*)$ son también diferenciables en (x^*, y^*), entonces las condiciones anteriores pueden ser escritas, de modo equivalente como:

$$\begin{aligned} \nabla f(x^*, y^*) + \sum_{i=1}^{m} u_i \nabla g_i(x^*, y^*) + \sum_{j=1}^{l} v_j \nabla h_j(x^*, y^*) = 0 \\ u_i g_i(x^*, y^*) = 0 \qquad \forall i \in \{1,...,m\} \\ u_i \geq 0 \qquad \forall i \in \{1,...,m\}. \end{aligned} \tag{3.3}$$

Los escalares u_j y v_j se conocen como *multiplicadores de Lagrange* y nos serán de gran ayuda para entender la relaciones existentes entre los problemas primal (P) y dual (D) en optimización no lineal. La condición de que (x^*, y^*) sea factible se conoce como *condición de factibilidad del primal*, las

condiciones $u_i g_i(x^*, y^*) = 0$ son las *condiciones de holguras complementarias* y las condiciones sobre los gradientes y sobre u_i se conocen como *condiciones de factibilidad del dual*. Si se quiere ver una demostración formal de este teorema se puede consultar en Bazaraa et al. (2005).

Si se recuerda del primer capítulo, el algoritmo de Benders se basa en ir resolviendo de forma iterativa el subproblema y el maestro, pasando información el primero al segundo mediante cortes que se construyen a partir de la teoría de la dualidad. Es por ello, que se va a presentar la función *lagrangiana* que es de gran relevancia para la construcción del problema dual en programación matemática no lineal. Dicha función, dado el problema (3.1) se define como:

$$L(x, y, u, v) = f(x, y) + \sum_{i=1}^{m} u_i g_i(x, y) + \sum_{j=1}^{l} v_j h_j(x, y). \tag{3.4}$$

Se puede ver que esta función guarda bastante relación con las condiciones KKT. Dado un punto (x^*, y^*) que cumple las condiciones KKT tendríamos que $u^* \geq 0$ y v^*. Entonces se puede rescribir la función lagrangiana restringida al primal como:

$$L^{RP}(x, y) = f(x, y) + \sum_{i=1}^{m} u_i^* g_i(x, y) + \sum_{j=1}^{l} v_j^* h_j(x, y). \tag{3.5}$$

En optimización no lineal una de las formulaciones más utilizadas para estudiar las relaciones entre los primales y los duales (teoría de la dualidad) es el dual lagrangiano que guarda una estrecha relación con la función lagrangiana (3.4). Asociadas a la función lagrangiana se tienen las funciones primal y dual, dadas por $L^P(x, y)$ y $L^D(u, v)$ y cuyas definiciones son:

$$L^P(x, y) = \sup_{(u,v), v \geq 0} L(x, y, u, v) \qquad y \qquad L^D(u, v) = \inf_{(x,y) \in S} L(x, y, u, v), \tag{3.6}$$

cuyos problemas primal (P) y dual (D) se definen respectivamente como:

$$\begin{aligned} \text{mín} \quad & L^P(x, y) \\ \text{s.a.} \quad & (x, y) \in S, \end{aligned} \tag{3.7}$$

y,

$$\begin{aligned} \text{máx} \quad & L^D(u, v) \\ \text{s.a.} \quad & u \geq 0. \end{aligned} \tag{3.8}$$

Se puede ver fácilmente que $L^P(x, y)$ toma valores en $\mathbb{R} \cup \{+\infty\}$ y $L^D(u, v)$ en $\mathbb{R} \cup \{-\infty\}$. Supongamos que un punto (x, y) es factible en (3.1), entonces $L^P(x, y) = f(x, y)$ ya que las restricciones asociadas a u_i toman valores no positivos y las restricciones asociadas a v_j toman el valor 0. Para minimizar $L^P(x, y)$ dado que (x, y) es factible lo mejor que se puede hacer es tomar $u_i = 0$, con lo que se obtiene $f(x, y)$. En el caso de que (x, y) no sea factible y se viole cierta restricción, podremos hacer tender $L^P(x, y)$ a infinito sin más que incrementar la variable dual asociada a la restricción o restricciones que se están violando. Estas relaciones implican que el problema (3.7) y el (3.1) sean problemas equivalentes.

Haciendo explicitas las expresiones matemáticas se puede definir el dual lagrangiano como:

$$\begin{aligned} \text{máx} \quad & L^D(u, v) = \inf_{(x,y) \in S} f(x, y) + \sum_{i=1}^{m} u_i g_i(x, y) + \sum_{j=1}^{l} v_j h_j(x, y) \\ \text{s.a.} \quad & u \geq 0. \end{aligned} \tag{3.9}$$

Se puede ver que el dual lagrangiano consiste en pasar penalizadas a la función objetivo todas las restricciones del problema primal excepto aquellas definidas en los conjuntos S.

Es fácil ver que la condición de factibilidad del dual de KKT en el punto (x^*, y^*) es equivalente a que $\nabla L^P(x^*, y^*) = 0$. La primera relación que se puede obtener de la función (3.5) es que para cualquier punto factible $L^{RP}(x, y) < f(x, y)$ ya que si el punto es factible $h_j(x, y) = 0$, $g_i(x, y) \leq 0$ y $u_i^* > 0$. De las condiciones KKT del primal junto con el lagrangiano restringido al primal si tenemos un punto (x^*, y^*) que sea un mínimo local para $L^{RP}(x, y)$, entonces existirá un entorno de (x^*, y^*) tal que, para todo (x, y) en el entorno, $f(x^*, y^*) = L^{RP}(x^*, y^*) \leq L^{RP}(x, y) \leq f(x, y)$. Esta última relación implica que en un entorno local el lagrangiano restringido al primal es una cota inferior de la función objetivo del primal.

Ahora que ya se tienen las formulaciones del problema primal y su dual en problemas de optimización no lineal se van a presentar dos de los teoremas más importantes en teoría de dualidad: el teorema de dualidad débil y el teorema de la dualidad fuerte.

Teorema 3. *(Teorema de dualidad débil). Dado un par de soluciones factibles (x, y) y (u, v) de los problemas P y D, respectivamente, entonces $f(x, y) = L^P(x, y) \geq L^D(u, v)$.*

Analíticamente se puede ver que

$$\begin{aligned} L^D(u, v) &= \text{ínf}_{(x^*, y^*) \in S} f(x^*, y^*) + \sum_{i \in R} u_i g_i(x^*, y^*) + \sum_{j=1}^{l} v_j h_j(x^*, y^*) \\ &\leq f(x, y) + \sum_{i \in R} u_i g_i(x, y) + \sum_{j=1}^{l} v_j h_j(x, y) \leq f(x, y), \end{aligned} \tag{3.10}$$

donde la primera desigualdad no es más que aplicar la definición de ínfimo y la segunda se obtiene de que dado que (x, y) y (u, v) son factibles, entonces $u \geq 0$, $g_i(x, y) \leq 0$ y $h_j(x, y) = 0$.

El teorema de la dualidad débil indica que la función objetivo de cualquier solución factible del primal siempre será superior a la función objetivo de cualquier solución factible del dual. Dado que el primal es de minimizar, para cualquier punto factible la función objetivo de este siempre será una cota superior del valor óptimo del problema mientras que en el dual cualquier solución factible será siempre una cota inferior para la solución óptima del problema original. Formalmente, del teorema de dualidad débil surgen las siguientes implicaciones:

Corolario 3.1. *$ínf_{(x,y)\in S}\{f(x, y) : g(x, y) \leq 0, h(x, y) = 0\} \geq sup\{L^D(u, v), u \geq 0\}$.*

Corolario 3.2. *Si $f(x^*, y^*) = L^D(u^*, v^*)$, con (x^*, y^*) factible en P y (u^*, v^*) factible en D, entonces (x^*, y^*) es óptimo de P y (u^*, v^*) es óptimo de D.*

Corolario 3.3. *Si $ínf_{(x,y)\in S}\{f(x, y) : g(x, y) \leq 0, h(x, y) = 0\} = -\infty$, entonces $L^D(u, v) = -\infty$, para todo $u \geq 0$.*

Corolario 3.4. *Si $sup\{L^D(u, v), u \geq 0\} = \infty$, entonces P no tiene soluciones factibles.*

Los corolarios que se acaban de presentar serán de gran ayuda para entender cómo se construye el problema maestro en el algoritmo de Benders generalizado y como se relaciona el mismo con el subproblema. Es importante destacar que el corolario 3.2 requiere supuestos de convexidad. Bajo dichos supuestos es una condición suficiente para la optimalidad global de una solución factible del primal. Es decir, si se ha encontrado una solución factible en P y encontramos una solución factible en D con el mismo valor para la función objetivo, entonces esta solución es el óptimo global del problema. Sin embargo, en general, no siempre se cumple e incluso puede que dicha solución no exista con lo que se estará ante soluciones del tipo $\text{ínf}_{(x,y)\in S}\{f(x, y) : g(x, y) \leq 0, h(x, y) = 0\} > \sup\{L^D(u, v), u \geq 0\}$, y

la diferencia entre estas dos cantidades se conoce como *duality gap*. En el caso de que el problema sea de optimización convexa podemos garantizar que se puede cerrar el duality gap a 0. Este último resultado está estrechamente relacionado con el teorema de dualidad fuerte que se presenta a continuación.

Teorema 4. *(Teorema de dualidad fuerte). Sea P un problema de optimización convexa en el que el vector 0 pertenece al interior de $h(X,Y)$ y se cumple la siguiente condición de regularidad: existe (x^*, y^*) tal que $g(x^*, y^*) < 0$ y $h(x^*, y^*) = 0$. Entonces no existe duality gap, es decir,*

$$ínf_{(x,y)\in S}\{f(x,y) : g(x,y) \leq 0, h(x,y) = 0\} = sup_{u\geq 0} L^D(u,v). \tag{3.11}$$

Además, si el anterior ínfimo es finito tenemos que

I $sup_{u\geq 0} L^D(u,v)$ se alcanza para un par (u^*, v^*) con $u^* \geq 0$ y

II si el ínfimo se alcanza en (x^*, y^*), entonces $u^* g(x^*, y^*) = 0$.

Si se quiere ver una demostración formal de este teorema se puede consultar en Bazaraa et al. (2005).

3.1. Método de descomposición de Benders generalizado

En Floudas & Aggarwal (1990) se propone el método de descomposición de Benders generalizado para problemas de optimización no lineales. Los autores enfocan el método desde la óptica de la resolución de los problemas de pooling, sin embargo se trata de un método que puede ser aplicado a otro tipo de problemas no lineales. La idea que está detrás de este algoritmo es esencialmente la explicada en el primer capítulo. Es decir, se basa en buscar aquellas variables complicantes que nos permitan identificar algún tipo de estructura de separación para poder separar el problema original en dos problemas diferentes más sencillos de resolver: subproblema y maestro. Una vez se tengan ambos problemas, con el maestro se conseguirán propuestas para los valores que deben tomar las variables complicantes y el subproblema le transmitirá información al maestro a través de los duales para indicarle como deberían cambiar el valor las variables complicantes en la siguiente iteración con el objetivo de llegar a una solución óptima. Es importante destacar que, aunque se puedan resolver todos los problemas intermedios del algoritmo a sus respectivos óptimos globales, el algoritmo no garantiza que el óptimo global del problema original sea encontrado.

Para explicar cómo se construye el subproblema y el maestro en el contexto de la optimización no lineal nos basaremos en la teoría de dualidad presentada previamente junto con los fundamentos teóricos propuestos en Geoffrion (1972) y Floudas (1995). Supongamos que tenemos el siguiente problema de optimización:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{s.a.} \quad & g_i(x,y) \leq 0 \qquad i = 1,...,m \\ & h_j(x,y) = 0 \qquad j = 1,...,l, \\ & x \in X, \\ & y \in Y, \end{aligned} \tag{3.12}$$

donde $X \times Y \subseteq S$. y es el vector de variables complicantes. Es decir, aquellas variables en que en caso de ser fijadas como parámetros harían que el problema fuese más fácil de resolver. En concreto, en lo problemas de pooling tenemos que si se fija la variable complicante proporción de flujo pasamos de un problema no convexo a, una vez descompuesto, dos problemas convexos que si son fáciles de resolver con los *solvers* y métodos existentes.

Los supuestos fundamentales sobre los cuales se puede aplicar el algoritmo, expuestos en Geoffrion (1972) son:

- Para un valor fijo de y el problema original puede ser separado en diferentes problemas de optimización definidos sobre los vectores de variable x.
- Fijado un valor de y los problemas resultantes pueden ser resueltos de manera eficiente con las técnicas existentes.
- El problema (3.12) no es convexo en x y y conjuntamente, pero fijado el valor de y el problema es convexo en x.

La construcción del subproblema es análoga a la explicada en el primer capítulo. Una vez se ha identificado la variable complicante esta se fija en el problema original creando así el subproblema. Es decir,

$$\begin{aligned} \min_{x} \quad & f(x, \bar{y}) \\ \text{s.a.} \quad & g_i(x, \bar{y}) \leq 0 \qquad i = 1, ..., m \\ & h_j(x, \bar{y}) = 0 \qquad j = 1, ..., l, \\ & x \in X. \end{aligned} \tag{3.13}$$

En ocasiones durante este capítulo se utilizara la notación vectorial $g(x, y)$ y $h(x, y)$ para designar $g_1(x, y), ..., g_m(x, y)$ y $h_1(x, y), ..., h_l(x, y)$ respectivamente.

3.1.1. Deducción del problema maestro

Una vez se tiene el subproblema hay que buscar la forma de construir el problema maestro para problemas de optimización no lineales. La construcción de este problema se basa en tres premisas propuestas en Geoffrion (1972):

1. la proyección del problema (3.12) sobre el espacio generado por las variables complicantes y,
2. la representación dual de V que esencialmente es la región factible del problema para valores fijos de y. Este será presentado a continuación y,
3. la representación dual del problema (3.12) sobre el espacio generado por y.

La proyección del problema (3.12) sobre el espacio generado por las variables complicantes y es la siguiente. El problema original puede ser escrito de forma equivalente como:

$$\begin{aligned} \min_{y} \quad & \text{ínf}_x \quad f(x, y) \\ & \text{s.a.} \quad g_i(x, y) \leq 0 \qquad i = 1, ..., m \\ & \qquad h_j(x, y) = 0 \qquad j = 1, ..., l \\ & \qquad x \in X, \\ & \qquad y \in Y. \end{aligned} \tag{3.14}$$

Parametrizando el problema anterior en y tendremos una formulación equivalente para el subproblema (3.13) que llamaremos $v(y)$ y será:

$$\begin{aligned} v(y) = \text{ínf}_x \quad & f(x, y) \\ \text{s.a.} \quad & g_i(x, y) \leq 0 \qquad i = 1, ..., m \\ & h_j(x, y) = 0 \qquad j = 1, ..., l \\ & x \in X, \\ & y \in Y \cap V, \end{aligned} \tag{3.15}$$

donde V se define como,

$$V = \{y : h(x,y) = 0, g(x,y) \leq 0 \quad \text{para algún} \quad x \in X\}. \tag{3.16}$$

Con lo que la solución de este problema consiste en encontrar el mínimo en x dado un valor fijo de y perteneciente a la región factible del problema original. La proyección del subproblema y del problema original sobre el espacio generado por y tiene una serie de implicaciones presentadas en el siguiente teorema (Geoffrion 1972).

Teorema 5. *(Proyección). El problema (3.12) es infactible o tiene una solución óptima no acotada si lo mismo ocurre para el problema (3.14). Si (x^*, y^*) es una solución óptima en (3.12), entonces y^* tiene que ser una solución óptima para (3.14). Si y^* es una solución óptima en (3.14) y x^* es el ínfimo en (3.15) cuando se ha fijado el valor de $y = y^*$, entonces (x^*, y^*) es una solución óptima para el problema original (3.12).*

Supongamos ahora que se ha resuelto el subproblema y se obtiene que para un valor fijo de y la solución del problema es infactible. Al igual que en el primer capítulo habrá que encontrar alguna forma de reconducir el algoritmo hacia soluciones factibles del subproblema. Para ello, es necesaria la representación dual de V. Supongamos que X es un conjunto convexo no vacío y que g_i son convexas y h_j son lineales en X para valores fijos de $y \in Y$. Además, supongamos que existe un conjunto cerrado tal que para cada $y \in Y$ se tiene,

$$Z_y = \{z \in \mathbb{R}^m : h(x,y) = 0, g(x,y) \leq z \quad \text{para algún} \quad x \in X\}, \tag{3.17}$$

es decir, tiene que existir valores $\bar{y} \in Y$ que estén en V tal que exista un punto z para que el conjunto anterior sea acotado y no vacío. Para que esto ocurra el siguiente sistema tiene que tener solución:

$$\begin{aligned} 0 &\geq L_*(x, y, \bar{u}, \bar{v}), \quad \forall \bar{u}, \bar{v} \in \Theta, \\ \text{donde} \quad \Theta &= \Big\{\bar{u} \in \mathbb{R}^m, \bar{v} \in \mathbb{R}^l : \bar{u} \geq 0, \sum_{i=1}^{l} \bar{u}_i = 1\Big\}, \\ \text{y donde} \quad L_*(x, y, \bar{u}, \bar{v}) &= \bar{u} g(x,y) + \bar{v} h(x,y). \end{aligned} \tag{3.18}$$

Nótese que $L_*(x, y, \bar{u}, \bar{v})$ es la representación dual de V. La demostración de que este sistema siempre tiene solución cuando el problema de partida es factible y que la solución trivial $\bar{v} = 0$ es una solución para dicho sistema se puede consultar en Geoffrion (1972). Además, la condición aquí presentada es fundamental para la construcción de los cortes de factibilidad ya que se trata de encontrar dado un valor fijo para y un punto factible del problema dado ese valor.

En la práctica, supongamos que se ha resuelto el subproblema para un valor $\bar{y}$ y resulta que la solución de este problema es infactible. Siguiendo a Floudas & Aggarwal (1990) una forma de encontrar una solución que cumpla las condiciones de (3.18) es resolver el siguiente problema:

$$\begin{aligned} \min_{x,\alpha} \quad & \alpha \\ \text{s.a.} \quad & g_i(x, \bar{y}) - \alpha \leq 0 \qquad i = 1, ..., m \\ & h_j(x, \bar{y}) - \alpha \leq 0 \qquad j = 1, ..., l, \\ & -h_j(x, \bar{y}) - \alpha \leq 0 \qquad j = 1, ..., l, \\ & \alpha \geq 0 \\ & x \in X. \end{aligned} \tag{3.19}$$

Este problema se conoce con el nombre de subproblema infactible. Otras formulaciones del mismo se pueden consultar en Floudas (1995). Resolver (3.19) consiste en encontrar aquel punto x^* tal que para

un valor fijo de y se minimice la máxima violación de todas las restricciones del problema. La solución óptima del mismo nos da el punto "más factible" dado el valor de las variables complicantes. El corte de factibilidad asociado a este problema viene dado por la siguiente formulación del lagrangiano:

$$L_*(x^*, \bar{y}, u^*, v^*) = u^* g(x^*, \bar{y}) + v^* h(x^*, \bar{y}), \tag{3.20}$$

donde $v^* = v_1^* - v_2^*$ que son los multiplicadores asociados a las restricciones obtenidas mediante el desdoblamiento de las restricciones de igualdad y u^* son los multiplicadores de Lagrange asociadas a las restricciones de desigualdad del problema (3.19). Si nos fijamos esta función es equivalente a la representación dual del conjunto V presentando en (3.18). Siendo menos formales lo que se quiere precisamente es evitar el corolario 3.4 de la dualidad débil que indica que si un problema de optimización no lineal es infactible su dual toma el valor infinito. Es decir, si el subproblema es infactible se busca un valor dado $\bar{y}$ que sea lo más factible posible en el subproblema. Además, si se tiene en cuenta la interpretación económica de los multiplicadores de Lagrange estos indican cuanto mejoraría la función objetivo por cada unidad que se relaje su restricción asociada. Entonces, teniendo en mente el problema (3.19) donde los valores de y están fijos los multiplicadores nos dicen como mejoraría la factibilidad del subproblema si se pudiera relajar la restricción asociada y, como las variables complicantes se resuelven en el maestro estas restricciones sí se podrán relajar para otros valores de y teniendo en cuenta la información proporcionada por los multiplicadores de Lagrange. Por lo tanto, teniendo presente (3.17), (3.18) y (3.19) es sencillo deducir que los cortes de factibilidad que serán introducidos en el problema maestro serán:

$$L_*(x^*, y, u^*, v^*) \leq 0, \tag{3.21}$$

donde, en el problema maestro, y deja de ser una constante y el resto de variables serán parámetros. Además que (3.21) sea menor que cero evita que si el subproblema es infactible el dual tome el valor infinito y, por otro lado, el resultado presentado en (3.18) nos asegura que existe una solución y^* tal que se cumpla la condición (3.21). En caso de que dicha condición no se cumpla indicará que el problema de partida no es factible.

El último ingrediente que falta para la construcción del problema maestro son los cortes de optimalidad. Al igual que en el primer capítulo estos cortes se derivan a partir del subproblema cuando este tiene soluciones factibles. Para ello, en programación no lineal es necesario presentar la representación dual del problema (3.12) sobre el espacio generado por y. Si recordamos de la dualidad en programación no lineal podemos escribir $L^D(u, v)$ del problema (3.12) de manera equivalente como:

$$\sup_{v,u\geq 0} \quad \inf_{x\in X} L(x, y, u, v), \quad \forall y \in Y \cap V, \tag{3.22}$$

que como veremos a continuación $L(x, y, u, v)$, definida en (3.4), será la ecuación necesaria para la construcción de los cortes de optimalidad.

Para la derivación del problema maestro recordemos que por dualidad fuerte se sabe que en el óptimo $f(x, y) = L^D(u, v)$ pero como puede existir duality gap nos conformaremos con la condición de la dualidad débil que indica $f(x, y) \geq L^D(u, v)$ ya que nuestros problemas son de minimizar. En el primer capítulo se había comentado que el problema maestro era una relajación del problema original y que si todos los cortes posibles generados por el algoritmo eran considerados en el maestro, este último sería equivalente al problema de partida. Por ello, teniendo en cuenta las relaciones anteriores podemos sustituir (3.22) y (3.18) en (3.14) obteniendo el siguiente problema equivalente al (3.14):

$$\begin{aligned} \min_{y} \quad & \sup_{v,u\geq 0} \quad \inf_{x\in X} L(x, y, u, v) \\ & \text{s.a.} \quad \inf_{x\in X} L_*(x, y, \bar{u}, \bar{v}) \leq 0, \quad \forall (\bar{u}, \bar{v}) \in \Theta, \end{aligned} \tag{3.23}$$

añadiendo una variable escalar auxiliar μ y utilizando la definición del supremo como mínima cota

superior se obtiene:

$$\begin{aligned} \min_{y\in Y,\mu} \quad & \mu \\ \text{s.a.} \quad & \inf_{x\in X} L(x,y,\bar{u},\bar{v}) \leq \mu, \quad \forall \bar{v}, \forall \bar{u} \geq 0 \\ & \inf_{x\in X} L_*(x,y,\bar{u},\bar{v}) \leq 0, \quad \forall (\bar{u},\bar{v}) \in \Theta. \end{aligned} \tag{3.24}$$

Este último problema es justamente la formulación del problema maestro donde el primer conjunto de restricciones representa los cortes de optimalidad y el segundo conjunto los cortes de factibilidad. Además, por la dualidad débil de puede ver que el corte de optimalidad acota inferiormente el problema original con lo que las soluciones del problema maestro son una cota inferior del problema (3.14).

Lo último que se quiere resaltar en esta subsección son las intuiciones para la interpretación geométrica del problema maestro. Cuando se presentó el algoritmo de Benders en el mundo lineal se había comentado que este caracterizaba los puntos extremos y direcciones extremas del problema original. En este caso, como el algoritmo está pensado para problemas de optimización no lineales ya no se puede interpretar de esta forma. Ahora lo que se consigue con el problema maestro es aproximar la región factible del problema original mediante los lagrangianos de las restricciones del subproblema. Supongamos que los lagrangianos son lineales en y entonces el maestro irá construyendo la región factible del problema (3.15) mediante hiperplanos. Si se quiere buscar una explicación más formal de la interpretación geométrica del problema maestro de puede consultar en Floudas (1995).

3.2. Algoritmo de Benders generalizado

Al igual que ocurría en el método de Benders presentado en el primer capítulo el subproblema es un problema más restringido que el problema original con lo que este será una cota superior para el mismo. Por su parte, el problema maestro, es una relajación del problema original con lo que acotará inferiormente a dicho problema. Cabe decir que la cota inferior aumenta de forma monótona a medida que se van incorporando cortes de optimalidad, por lo que irá cerrando el duality gap de forma iterativa. En caso de que el problema original sea un problema de optimización convexa, en el óptimo, las cota de ambos problemas coincidirían. Sin embargo, como el método de Benders generalizado está pensado para un campo más amplio de problemas de optimización. Si el problema de partida es no convexo, como los problemas de pooling, puede existir duality gap con lo que las cotas de ambos problemas nunca convergerán al mismo punto porque dicho punto puede no existir. La forma de remediarlo, con el fin de que el algoritmo termine, es monitorizar las cotas dadas por el problema maestro y si estas no cambian en un número determinado de iteraciones el algoritmo termina. A continuación, en el Algoritmo 2 se muestra el pseudocódigo del método de descomposición de Benders generalizado.

Es importante recordar que si en el problema que se quiera resolver tenemos restricciones en las que solo aparecen involucradas variables complicantes estas restricciones deben ser trasladadas al problema maestro. El resto de consideraciones para su implementación en la práctica son análogas a las comentadas en el primer capítulo.

Lo último que se quiere destacar es que el algoritmo de Benders generalizado converge en una cantidad finita de paso. La demostración de la convergencia del algoritmo se puede consultar en Geoffrion (1972).

3.3. Aspectos específicos del modelado

En el proceso de investigación llevado a cabo para la realización de este documento se han probado distintas formulaciones para el subproblema y el subproblema infactible. Es importante destacar que lo presentado en esta subsección presenta un buen comportamiento para los problemas de pooling pero,

Algoritmo 2: Método de descomposición de Benders generalizado

Paso *0*: Se inicializa $y^1 \in Y$ y $\epsilon \geq 0$ se establecen las cotas superiores e inferiores a $UB = \infty$ y $LB = -\infty$. Además, $k = 0$. Se inicializa $K^{feas} = K^{infeas} = 0$, $LB_{times} = 20$ y $LB_{count} = 0$.

Paso *1*: Resuelve el subproblema de Benders. Actualiza $k = k + 1$.

$$\begin{aligned} \min_{x} \quad & f(x, y^k) \\ \text{s.a.} \quad & g_i(x, y^k) \leq 0 \qquad i = 1, ..., m \\ & h_j(x, y^k) = 0 \qquad j = 1, ..., l, \\ & x \in X. \end{aligned} \tag{3.25}$$

si *El subproblema tiene solución óptima* **entonces**
| $UB = \min(UB, f(x, y^k))$;
| se genera el corte de optimalidad de Benders $L(x^k, y, v^k, u^k) \leq \mu$ y se actualiza $K^{feas} = K^{feas} + 1$;

si no, si *El subproblema es infactible se resuelve el subproblema infactible* **entonces**

$$\begin{aligned} \min_{x,\alpha} \quad & \alpha \\ \text{s.a.} \quad & g(x, y^k) - \alpha \leq 0 \\ & h(x, y^k) - \alpha \leq 0 \\ & -h(x, y^k) - \alpha \leq 0 \\ & \alpha \geq 0 \\ & x \in X. \end{aligned} \tag{3.26}$$

Se genera el corte de factibilidad de Benders $L_*(x^k, y, u^k, v^k) \leq 0$ y se actualiza $K^{infeas} = K^{infeas} + 1$;

Paso *2*: Se resuelve el problema Maestro;

$$\begin{aligned} \min_{y \in Y, \mu} \quad & \mu \\ \text{s.a.} \quad & L(x^k, y, u^k, v^k) \leq \mu, \quad k = 1, ..., K^{feas} \\ & L_*(x^k, y, u^k, v^k) \leq 0, \quad k = 1, ..., K^{infeas}. \end{aligned} \tag{3.27}$$

obteniendo los valores y^*, μ^*;
Se actualiza la cota inferior $LB = \max(LB, \mu^*)$.

si *el maestro es infactible* **entonces**
| Termina.
si $UB - LB \leq \epsilon$ **entonces**
| Termina.
si *Si* $LB_{times} = LB_{count}$ **entonces**
| Termina.
en otro caso
| $LB_{count} = LB_{count} + 1$;
vuelve a **Paso** *1*.

en general, no tiene porque ser así para todo tipo de problemas. En caso de que se quiera implementar este algoritmo para la resolución de otro tipo de problemas se tendrán que estudiar sus características concretas y en base a ello formularlos de manera adecuada.

El primer problema que nos encontramos y que también se encontró en Floudas & Aggarwal (1990), utilizando una formulación diferente a la aquí presentada para los problemas de pooling, es que aun consiguiendo una solución óptima para un valor fijo de las variables complicantes puede ocurrir que con dicha solución no se pueda generar ningún corte de optimalidad. Si pensamos detenidamente, los cortes del problema maestro se generan a partir de restricciones del subproblema que tengan implicadas variables complicantes. Si el multiplicador asociado a dicha restricción toma el valor cero el corte queda anulado provocando que el problema maestro deje de ser una guía para futuros valores de las variables complicantes. En un punto donde se cumplan las condiciones KKT se pueden dividir las restricciones activas en dicho punto en dos conjuntos. En concreto,

- Restricciones fuertemente activas. $R^+(x^*, y^*) = \{i : u_i^* > 0\}$.
- Restricciones débilmente activas. $R^0(x^*, y^*) = \{i : u_i^* = 0\}$.

A modo aclaratorio para cualquier punto factible (x, y) se considera que una restricción del tipo $g(x, y) \leq 0$ esta activa en ese punto si $g(x, y) = 0$ e inactiva si $g(x, y) < 0$. Por su parte, las restricciones del tipo $h(x, y) = 0$ siempre están activas para cualquier punto factible del problema.

En Floudas & Aggarwal (1990) para solventar dicho problema introducen nuevas restricciones en el problema y nuevas variables binarias para asegurar que en una solución concreta del subproblema al menos una de las restricciones que tengan implicadas variables complicantes esté fuertemente activa en dicho punto. En este documento se hace una propuesta diferente para conseguir que los multiplicadores asociados dichas restricciones sean distintos de cero sin incrementar la dificultad computacional del problema que se deriva de la introducción de variables binarias.

Lo primero que se ha hecho es reformular el subproblema con todas sus restricciones de menor o igual. La forma de hacerlo es la siguiente:

- Restricciones del tipo $g(x, y) \geq 0$ puede ser escritas como $-g(x, y) \leq 0$.
- Restricciones del tipo $h(x, y) = 0$ se pueden desdoblar en dos restricciones del tipo $-h(x, y) \leq 0$ y $h(x, y) \leq 0$.

Una vez se ha hecho esta transformación para todas las restricciones del subproblema se introduce una variable de holgura en las restricciones en las que aparezcan implicadas variables complicantes y esta variable se penaliza en la función objetivo multiplicándola por un valor M muy grande. Para entenderlo, como tenemos todas las restricciones de menor o igual y teniendo en mente las condiciones KKT del subproblema pueden ocurrir dos cosas:

- La primera es que la restricción no este activa para un valor fijo de las variable complicante con lo que para cumplir las condiciones KKT necesariamente su multiplicador asociado toma el valor cero.
- La segunda es que la restricción este activa en ese punto con lo que su multiplicador tomará un valor positivo.

Al añadir la variable de *slack* (holgura), x^{slack}, penalizada en la función objetivo del problema justamente obligamos a que al menos una de las restricciones que tenga variables complicantes implicadas esté activa en ese punto. Lo que se acaba de explicar se entenderá mejor con un ejemplo y para ello consideremos la formulación PQ de los problemas de pooling con las transformaciones aquí comentadas. Nótese que las variables complicantes serán las variables correspondientes a las proporciones de

flujo y por tanto aparecerán fijas como $\bar{y}$.

$$\min_{f,x} \quad \sum_{s\in S}\sum_{t\in T}\Big(c_{st}f_{st} + \sum_{i\in I}(c_{si}+c_{it})x_{sit}\Big) + Mx^{slack} \tag{3.28}$$

$$\text{s.a.} \quad \sum_{t\in T}\Big(f_{st} + \sum_{i\in I}x_{sit}\Big) \le b_s, \qquad s\in S, \tag{3.29}$$

$$\sum_{s\in S}\sum_{t\in T}x_{sit} \le b_i, \qquad i\in I, \tag{3.30}$$

$$\sum_{s\in S}\Big(f_{st} + \sum_{i\in I}x_{sit}\Big) \le b_t, \qquad t\in T, \tag{3.31}$$

$$\sum_{s\in S}(q_s^k - q_t^k)\Big(f_{st} + \sum_{i\in I}x_{sit}\Big) \le 0, \qquad t\in T, k\in K, \tag{3.32}$$

$$\sum_{s\in S} -x_{sit} \le -f_{it}, \qquad i\in I, t\in T, \tag{3.33}$$

$$\sum_{s\in S} x_{sit} \le f_{it}, \qquad i\in I, t\in T, \tag{3.34}$$

$$\sum_{t\in T} x_{sit} \le b_i\bar{y}_i^s + x^{slack}, \qquad s\in S, i\in I, \tag{3.35}$$

$$-x_{sit} \le -\bar{y}_i^s f_{it} + x^{slack}, \qquad (s,i,t)\in P, \tag{3.36}$$

$$x_{sit} \le \bar{y}_i^s f_{it} + x^{slack}, \qquad (s,i,t)\in P, \tag{3.37}$$

$$-f_{it} \le 0, \qquad i\in S\cup I, t\in T, \tag{3.38}$$

$$x^{slack} \ge 0. \tag{3.39}$$

Si nos fijamos detenidamente el hecho de introducir una única variable de *slack* penalizada en la función objetivo provoca que o bien una de las restricciones asociadas a la variable de *slack* está activa en un punto dado o de lo contrario esta variable auxiliar tomará un valor positivo provocando que la función objetivo empeore. Como el problema es de minimizar los *solvers* empleados para resolver el subproblema darán una mayor importancia a reducir el valor de la función objetivo tratando que el valor de la variable auxiliar sea lo más pequeño posible. Esta será la formulación empleada para resolver el problema PQ con el método de Benders generalizado. La formulación para el problema TP es análoga.

La formulación del subproblema aquí presentada es la que proporcionó un mejor comportamiento en las pruebas realizadas. Además de esta se han considerado otras dos formulaciones. La primera consistía en mantener la formulación original del problema metiendo una variable de *slack* por cada restricción de menor o igual en las que aparecen implicadas variables complicantes y dos variables de *slack* en las restricciones análogas pero de igualdad. La segunda era similar a la formulación propuesta pero en vez de tener una única variable de *slack* se tenía una por restricción con variables complicantes implicadas. Finalmente, después de haber realizado diversas pruebas nos hemos decantado por la formulación aquí propuesta ya que era la que conseguía mejores resultados para los problemas de pooling.

En cuanto a la formulación para el subproblema infactible, en la literatura se han propuesto diversas formas de hacerlo. Ejemplos de ello se pueden consultar en Floudas (1995) o en Grothey et al. (2000). Para este trabajo se han probado tres formulaciones: la primera es la propuesta en Floudas & Aggarwal (1990) que es justamente el problema (3.19); la segunda se corresponde a una modificación de esta pero metiendo una variable de *slack* en cada restricción y en función objetivo tendríamos que minimizar un vector de variables en vez de una única variable y; la tercera consiste en lo mismo pero manteniendo la formulación original de las restricciones del problema lo que implica que en las restricciones de igualdad se tengan que introducir dos variables distintas de *slack*. Los resultados que se presentarán en este documento han sido ejecutados con la primera formulación porque ha sido la que mejores resultados nos ha aportado.

3.4. Mejoras o modificaciones del algoritmo

El objetivo principal de todo este trabajo ha sido la implementación del algoritmo de Benders generalizado para conseguir buenos óptimos locales para los problemas de pooling ya que como se ha comentado muchas industrias como las petroquímicas o gasísticas utilizan este tipo de modelos para la optimización de sus plantas de producción. Si pensamos desde el punto de vista de estas industrias querrán, además de tener buenos óptimos locales para el problema, conseguir dichas soluciones en el menor tiempo posible. Se podría pensar en por qué estas empresas no tratan de resolver sus problemas de optimización a optimalidad global y la respuesta es que con los *solvers* existentes para optimalidad global los tiempos de resolución pueden ser prohibitivos o incluso no llegar a conseguir dicha solución debido a la complejidad de los mismos. En esta sección se presentarán dos pequeñas modificaciones que pueden acelerar y mejorar el proceso de búsqueda de soluciones óptimas: cortes dinámicos y estabilización cuadrática del algoritmo.

3.4.1. Cortes dinámicos

En el algoritmo de Benders la mayor parte del esfuerzo computacional se produce a la hora de resolver el problema maestro. En general, el subproblema es lineal o convexo para valores fijos de las variables complicantes mientras que el maestro puede ser lineal, no lineal, no lineal con variables enteras, etc. Es decir, a priori el problema maestro presenta mayor dificultad desde el punto de vista de la optimización. Si a ello le sumamos que en cada iteración del algoritmo se añade una nueva restricción al maestro y si el número de iteraciones para encontrar la solución óptima es elevado puede causar que la resolución de dicho problema no sea una tarea sencilla. Justamente motivado por el tamaño que puede alcanzar el problema maestro y su dificultad en Soumis et al. (2012) se propone un algoritmo para realizar una actualización dinámica de los cortes introducidos en el problema maestro.

La idea detrás de este método de actualización de cortes es fácil de entender. Supongamos que estamos en una iteración k del algoritmo y resolvemos el problema maestro con k cortes. Entonces, puede ocurrir que en la solución óptima del problema maestro en esa iteración no todas las restricciones estén activas en ese punto implicando que la solución del problema maestro se podría alcanzar igualmente si dichas restricciones no estuvieran presentes en el problema. La idea del método consiste en eliminar de alguna forma las restricciones que no se están utilizando para la consecución de las soluciones en el problema maestro. Si lo pensamos detenidamente el problema maestro va construyendo la región factible del problema original mediante los cortes con lo que al llegar a determinado punto de la región puede ocurrir que muchos cortes ya no sean necesarios para caracterizarlo.

El hecho de eliminar los cortes provoca que el problema maestro sea más pequeño con lo que será más fácil de resolver. Sin embargo, lo que no se puede hacer es en cada iteración eliminar todos los cortes que estén inactivos porque el algoritmo se puede ciclar. Supongamos que estamos en una iteración k del algoritmo, resolvemos el maestro, obtenemos una nueva solución para las variables complicantes y eliminamos las restricciones que no han estado activas en ese punto. Ahora vamos a la

siguiente iteración $k+1$ repetimos el proceso y llegamos a otro punto para las variables complicantes. Comenzamos la iteración $k+2$ y el subproblema nos indica que los cortes que se deben introducir en el problemas maestro son justo los que se han eliminado en la iteración k, resolvemos el maestro y nos devuelve la solución obtenida en k. Este proceso podría repetirse indefinidamente y es lo que se conoce como ciclado del algoritmo.

El método propuesto en Soumis et al. (2012) esta diseñado para evitar que el algoritmo no se cicle o en caso de que se detecte un ciclo el algoritmo pueda salir de él. Lo que hace el método en la práctica es lo siguiente:

- Se espera un cantidad M de iteraciones antes de eliminar ningún corte.
- Una vez se ha superado M se comprueban los cortes que han estado inactivos por más de K iteraciones consecutivas. En caso de que se supere dicha cantidad se eliminan dichos cortes.
- si se han eliminado cortes en 3 iteraciones consecutivas se actualiza el factor M como $M = M \times 1{,}5$

Con la primera condición se espera a que el problema maestro tenga el número de cortes suficientes para que las soluciones que reporte sean una buena guía para el valor de las variables complicantes. Con la segunda condición se espera a tener un indicador razonable de que el corte ya no está aportando información al problema maestro. La tercera condición es la utilizada para evitar incurrir en ciclos, además con se ha propuesto una condición más que no esta recogida en Soumis et al. (2012) para facilitar que el algoritmo no se cicle. Si un corte que ha sido eliminado por el proceso dinámico se vuelve a introducir en el problema maestro en iteraciones posteriores este ya no se elimina nunca más. En la práctica, se lleva un registro de los cortes y los multiplicadores asociados a cada corte. Supongamos que estamos en una iteración k del algoritmo en la que tenemos k cortes en el problema maestro, resolvemos el subproblema y obtenemos un nuevo corte. Entonces, lo que se hace es ver si los coeficientes de este nuevo corte son iguales, con una tolerancia, a los coeficientes de los cortes que han sido eliminados pero que se saturarían para la solución del maestro en la iteración k. Si resulta que los cortes son iguales con esa cierta tolerancia este el corte que se introduce en la iteración k y ya no se elimina nunca más. A continuación, en el Algoritmo 3 se presenta el pseudocódigo del método que es una leve modificación del presentado en Soumis et al. (2012).

Algoritmo 3: Algoritmo de cortes dinámicos

Paso *0*: Se inicializa $M = 10$, $c^k_{inac} = 0$ y $K = 3$.

Paso *1*: Resuelve el problema maestro obteniendo μ^k.
si $\mu^k > \mu^{k-1}$ **entonces**
  Actualiza el contador de inactividad para cada corte $c^k_{inac} = c^k_{inac} + 1$;
  si *el número de cortes de Benders* $> M$ **entonces**
    si *el número de veces consecutivas que se han eliminado cortes* < 3 **entonces**
      Desactiva todos los cortes que han estado inactivos K o más veces;
    en otro caso
      $M = M \times 1{,}5$
en otro caso
  No trates de eliminar ningún corte;
  Ejecuta el algoritmo de Benders 2;
Resuelve el subproblema o subproblema infactible;
Añade el corte correspondiente al mestro;
si *el corte ya ha sido desactivado y está saturado en* k **entonces**
  Activa el corte en el maestro para siempre;
Vuelve a **Paso** *1*.

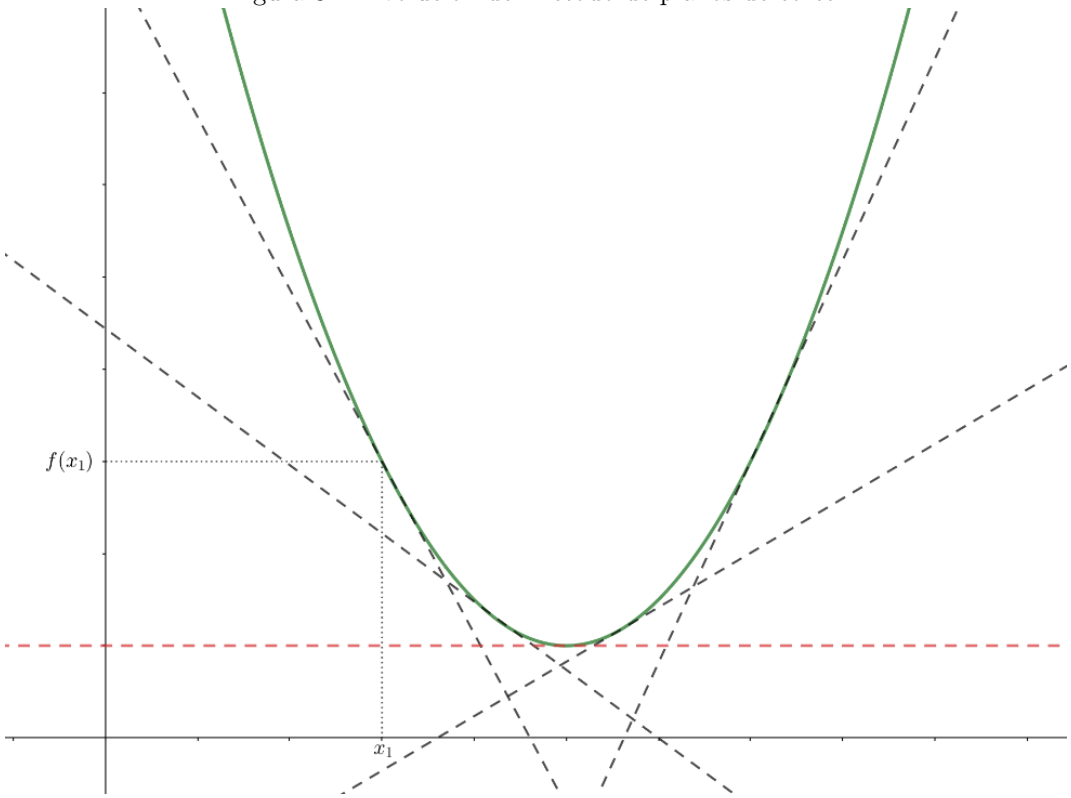
Los parámetros que aquí se establecen son los utilizados en la implementación práctica de este método para resolver los problemas de pooling. Se han probado distintas configuraciones y esta fue la que mejor funcionaba para todas las instancias ejecutadas. A la hora de elegir unos u otros valores dependerá de las propias características del problema. Por la experiencia práctica si el valor de M y de K son muy cercanos el algoritmo elimina una mayor cantidad de cortes pero puede incurrir en ciclos teniendo que ir aumentando el valor de M futuras iteraciones hasta salir del ciclo. En cambio si esta diferencia es muy grande es mucho más difícil que el algoritmo se cicle pero el número de cortes eliminados será menor. En definitiva, no hay una regla exacta para la elección de dichos parámetros, dependerá del problema concreto que se trate de resolver. Si se quiere ver una demostración de que el algoritmo de Benders con la actualización dinámica de cortes converge se puede ver en Soumis et al. (2012).

Lo último que se quiere destacar en esta subsección es que el proceso dinámico de eliminación de cortes puede provocar que la cota inferior definida por la función objetivo del problema maestro deje de ser monotonamente creciente.

3.4.2. Estabilización cuadrática del algoritmo

En esta subsección se va a presentar una pequeña heurística que se ha implementado con el objetivo de acelerar el proceso de búsqueda de soluciones óptimas en el algoritmo de Benders. Esta heurística se basa en los *bundle methods* que se pueden consultar en Lemaréchal et al. (1995). Los *bundle methods* fueron originalmente pensados para estabilizar el método de planos de corte (Kelley 1960) en problemas convexos. La idea del método de planos de corte consiste en ir aproximando la solución óptima del problema mediante hiperplanos, véase la Figura 3.1.

Figura 3.1: Evolución del método de planos de corte.



En este gráfico se muestran posibles hiperplanos que genera el método de planos de corte y la línea discontinua de color rojo representa la solución óptima del problema de optimización. La desventaja de este método es que cuando las soluciones del proceso iterativo se encuentran muy cerca de la solución óptima del problema se produce un fenómeno de oscilación entorno a dicho óptimo provocando que la convergencia del algoritmo sea lenta en la práctica.

Los *bundle methods* surgen precisamente para estabilizar el algoritmo. La idea subyacente al método es que cada vez que se produzca una mejora "importante", en términos de función objetivo, en la siguiente iteración se obliga a permanecer cerca del punto que la ha causado. En esencia el método obliga a mantenerse cerca de la mejor solución encontrada durante el proceso iterativo actualizándose esa mejor solución a medida que el algoritmo avanza.

La idea subyacente del problema maestro en el algoritmo de Benders consiste en construir la región factible del subproblema a través de la información proporcionada por los cortes de factibilidad y de optimalidad. Intuitivamente los primeros nos dicen hacía donde nos debemos mover para conseguir soluciones factibles y los segundos en que direcciones la función objetivo del subproblema mejoraría. Por ello, parece razonable pensar que una forma de acelerar el proceso de búsqueda de óptimos locales sería indicarle al problema maestro que no se aleje demasiado de los puntos factibles encontrados pero permitiendo movimiento en el caso de que se encuentre un nuevo punto factible que mejore el valor de la función objetivo.

A modo explicativo se presenta la forma más sencilla para los *bundle methods* apoyándonos en la formulación del problema maestro. Esta ha sido la utilizada en este trabajo y tiene la siguiente forma:

$$\begin{aligned} \min_{y\in Y,\mu} \quad & \mu + \frac{1}{2}R||y-y^b||^2, \\ \text{s.a.} \quad & \text{ínf}_{x\in X} L(x,y,\bar{u},\bar{v}) \leq \mu, \quad \forall \bar{v}, \forall \bar{u} \geq 0 \\ & \text{ínf}_{x\in X} L_*(x,y,\bar{u},\bar{v}) \leq 0, \quad \forall (\bar{u},\bar{v}) \in \Theta, \end{aligned} \tag{3.40}$$

donde y^b se conoce como centro de estabilización y R es un parámetro de penalización. Entonces, basándonos en estos métodos, la heurística que se ha propuesto consiste en cada vez que se introduzca un corte de optimalidad se comprueba si la función objetivo del problema maestro ha mejorado respecto a la iteración anterior. Si ha mejorado se actualiza el centro de estabilización al punto causante de la mejora, en caso contrario se deja el punto anterior. De cara a la implementación es importante tener en cuenta que solo se actualiza el centro de estabilización cuando se introducen cortes de optimalidad porque son los que realmente mueven la función objetivo del problema maestro. Por otro lado, si el algoritmo parte de puntos infactibles estarán introduciéndose cortes de factibilidad en el problema maestro hasta que se encuentre un punto factible, en ese caso se toma $R = 0$ hasta que se genere un corte de optimalidad y la función objetivo del maestro mejore respecto a la iteración anterior. Para resolver los problemas de pooling se ha fijado $R = 2$ ya que así conseguimos buenas soluciones. Una gran parte de la investigación realizada sobre estos métodos se basa precisamente es determinar qué se considera por una mejora importante en función objetivo y cuál es la mejor forma de actualizar el parámetro de penalización para que el algoritmo consiga mejores soluciones y termine lo más rápido posible. Más información sobre estos métodos aplicados al método de Benders se puede consultar en Zaourar & Malick (2014) y Rubiales et al. (2013).

Por último, es importante destacar que la implementación de esta heurística ha sido lo más básica posible en el sentido de que se considera una mejora "importante" en función objetivo cada vez que el valor de la función objetivo del problema maestro decrece y se ha propuesto un valor fijo para el parámetro de penalización. Es importante recordar que nuestro objetivo con esta heurística es acelerar el proceso de búsqueda de buenos óptimos locales no tener un método que busque el mejor óptimo local posible, pues para ello tenemos el algoritmo de Benders generalizado. Por otro lado, la introducción de esta expresión penalizada en función objetivo puede provocar, al igual que con la eliminación dinámica

de cortes, que la cota inferior definida por el maestro deje de ser monotonamente creciente.

Capítulo 4

Resultados Computacionales

El objetivo principal de este trabajo ha sido la implementación de una herramienta de optimización que nos permita obtener buenos óptimos locales para los problemas de pooling. Recapitulando en lo expuesto hasta el momento se tienen dos modelos PQ y TP y cuatro variantes del método de Benders generalizado para resolverlos:

- Algoritmo de Benders generalizado (B).
- Algoritmo de Benders generalizado con cortes dinámicos (BD).
- Algoritmo de Benders generalizado con estabilización cuadrática (BB).
- Algoritmo de Benders generalizado con estabilización cuadrática y cortes dinámicos (BBD)

En la práctica, cuando se trabaja con problemas complejos desde la óptica de la optimización matemática, es decir, no convexos y que tienen multitud de óptimos locales, se suelen emplear distintas técnicas en tándem para obtener mejores soluciones si cabe. Estas técnicas consisten en resolver con una herramienta de optimización el problema en cuestión e inicializar con la solución obtenida otro algoritmo para así conseguir un mejor óptimo local. Los tándem realizados en este trabajo consisten en resolver primero el problema con los métodos implementados y las soluciones obtenidas utilizarlas como punto de partida para la herramienta comercial de optimización no lineal Knitro (Byrd et al. 2006). Además, para determinar la calidad de las soluciones obtenidas con las distintas técnicas implementadas, compararemos nuestros resultados con los de la literatura sobre los problemas de pooling, en concreto, los presentados en Alfaki & Haugland (2014) y con los resultantes de utilizar Knitro y la herramienta comercial de optimalidad global BARON (Tawarmalani & Sahinidis 2005). También se ha intentado hacer la comparativa con una herramienta de optimización no lineal de software libre IPOPT (Wächter & Biegler 2005) pero finalmente se ha descartado porque no era capaz de obtener soluciones para la gran mayoría de las instancias. A modo aclaratorio todos los *solvers* empleados se lanzaron con sus configuraciones por defecto.

La implementación de los métodos se ha realizado íntegramente en el lenguaje de programación Python (Van Rossum & Drake 2009). Para el modelado de los problemas se ha utilizado la librería Pyomo (Hart et al. 2017) de Python que es un lenguaje de modelado matemático. A la hora de resolver el subproblema y el maestro, dado que dichos problemas resultantes de la descomposición son lineales, se han utilizado los *solvers* Gurobi (Gurobi Optimization 2019) y CBC (Forrest et al. 2018). El primero es una herramienta comercial diseñada para resolver problemas de optimización lineal y CBC es una herramienta de *software* libre diseñada especialmente para resolver problemas con variables enteras. Sin embargo, este último tiene implementado un potente solver lineal llamado CLP que es el que

realmente resuelve los problemas cuando son lineales y no presentan variables enteras. La versión de Gurobi utilizada ha sido la 8.0.0 y de CBC la 2.10.2. Por lo tanto, podemos tener una herramienta totalmente gratuita si los problemas intermedios se resuelven con *solvers* de *software* libre que como veremos más adelante proporcionan resultados competentes con sus análogos comerciales. La máquina empleada para la ejecución de los resultados tiene 64 gigas de memoria RAM y un procesador Intel(R) Core(TM) i7-6700K con 4.00GHz.

A continuación se presentarán las instancias utilizadas para los experimentos empíricos que han sido obtenidas de `http://www.ii.uib.no/~mohammeda/spooling/`. Estas instancias han sido ampliamente utilizadas en la literatura para testar distintos algoritmos de optimización tanto global como local. Ejemplos de ello se pueden consultar en Alfaki & Haugland (2013), Alfaki & Haugland (2014), Haverly (1978), Alfaki (2012) o Dai et al. (2018) entre otros. Es importante destacar que las instancias publicadas en el enlace están escritas en GAMS con lo que ha sido necesario traducirlas a Pyomo.

En la Tabla 4.1 se presenta información acerca de las instancias consideradas pequeñas en este documento. S representa el número de fuentes, I el número de pools o tanques, T es la cantidad de terminales, K es el número de propiedades y A representa la cantidad de arcos de la red. La columna nº vars hace referencia al número de variables implicadas en el modelo, la columna bilineal es el número de restricciones en las que aparecen implicados términos bilineales y la última columna representa el número de restricciones lineales implicadas en el modelo.

	S	I	T	K	A	nº vars	bilineal	lineal
Adhya1	5	2	4	4	13	33	20	42
Adhya2	5	2	4	6	13	33	20	50
Adhya3	8	3	4	6	20	52	32	62
Adhya4	8	2	5	4	18	58	40	55
Bental4	4	1	2	1	7	13	6	15
Bental5	5	3	5	2	32	92	60	53
Haverly1	3	1	2	1	6	10	4	13
Haverly2	3	1	2	1	6	10	4	13
Haverly3	3	1	2	1	6	10	4	13

Tabla 4.1: Tamaño de las instancias pequeñas.

Por su parte, en la Tabla 4.2 tenemos representado el tamaño de las instancias consideradas grandes. Como se puede ver las instancias aquí presentadas tiene un tamaño considerable, superando las 1000 variables y restricciones en la mayoría de los casos y llegando, las de mayor tamaño, a superar las 10000 variables y restricciones. Además, vemos como el número de restricciones con términos bilineales también es elevado y, si se recuerda, estos son los causantes de las no convexidades en los modelos de pooling. Debido al tamaño considerable y a la complejidad de las instancias, los resultados obtenidos en las pruebas empíricas pueden ser un indicador bastante realista del comportamiento que tendrían los algoritmos implementados sobre problemas reales de características similares.

	S	I	T	K	A	nº vars	bilineal	lineal
A0	20	10	15	24	171	500	329	531
A1	20	10	15	24	179	540	361	541
A2	20	10	15	24	192	756	538	562
A3	20	10	15	24	218	756	538	562
A4	20	10	15	24	248	979	731	592
A5	20	10	15	24	277	1245	968	611
A6	20	10	15	24	281	1364	1083	624
A7	20	10	15	24	325	1825	1500	661
A8	20	10	15	24	365	1895	1530	667
A9	20	10	15	24	407	2399	1992	701
B0	35	17	21	34	384	1537	1153	1093
B1	35	17	21	34	515	2354	1839	1180
B2	35	17	21	34	646	3739	3093	1273
B3	35	17	21	34	790	5327	4537	1385
B4	35	17	21	34	943	7534	6591	1494
B5	35	17	21	34	1044	8991	7947	1566
C0	60	15	50	40	811	3637	2826	2356
C1	60	15	50	40	1070	5840	4770	2533
C2	60	15	50	40	1278	7998	6720	2674
C3	60	15	50	40	1451	10567	9116	2828

Tabla 4.2: Tamaño de las instancias grandes.

4.1. Rendimiento de los algoritmos y comportamiento de los modelos

En esta sección se realizará un análisis de los resultados empíricos para las instancias presentadas previamente, formuladas con los modelos PQ y TP. Estas serán ejecutadas con: los algoritmos implementados, los tándem, Knitro y BARON. Para todas las pruebas realizadas se ha puesto un límite de tiempo máximo de 1200 segundos y solo se presentarán resultados referentes a soluciones factibles. A modo aclaratorio para los tándem se dejan estos 1200 segundos de tiempo máximo en los métodos implementados y en la segunda pasada con Knitro. Con ello se tratará de dar respuesta a las siguientes

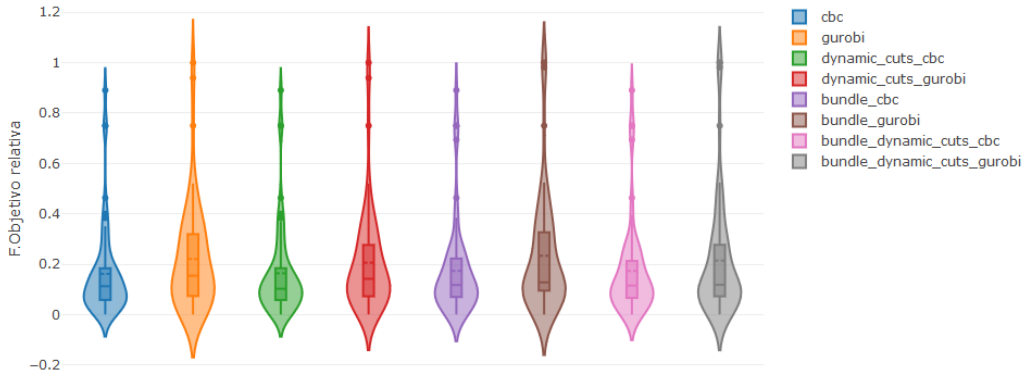
preguntas:

- ¿Qué método tiene un mejor rendimiento y consigue mejores resultados?
- ¿Existen diferencias de comportamiento de los algoritmos en función del modelo utilizado?
- En caso de querer tener la mejor solución posible, ¿se pueden conseguir mejoras utilizando los tándem?
- ¿Se mejoran los resultados obtenidos de Knitro, BARON o los expuestos en la literatura en Alfaki & Haugland (2014)?

Se comienza el análisis de los resultados haciendo una comparativa de los métodos (sin tándem) para ver cual presentan un mejor comportamiento para todas las instancias y modelos utilizados. La Figura 4.1 se conoce como *violin*. Estas representaciones gráficas están compuestas por un gráficos de cajas acompañados por las curvas de densidad empírica de los datos en cuestión. Las cajas contienen información acerca de la media y mediana (en el interior), el límite superior de la caja representa el cuartil que deja el 75 % (Q_3) de los datos por debajo de ese límite y el inferior se corresponde con el cuartil que acota el 25 % (Q_1) de los datos inferiormente, con lo que en el interior de la caja se encuentra el 50 % de los mismo. Las líneas que salen de las cajas se conocen como "bigotes" y se extienden hasta 1,5 veces el rango intercuartílico ($Q_3 - Q_1$). La ventaja de estos gráficos frente a los de cajas convencionales es que con las curvas de densidad empíricas se puede ver de forma más clara donde hay una mayor concentración de información.

Una vez explicado en que consiste el *violin* de forma general se va a proceder con la interpretación de los resultados aquí expuestos. La Figura 4.1 representa la distribución de los valores para la función objetivo relativa obtenida a partir de los distintos métodos para todas las instancias. Dado que los valores en función objetivo son muy diferentes (ver Apéndice B) se han rescalado sus valores donde el cero representa la mejor función objetivo y el uno la peor. En cuanto a la leyenda "cbc" y "gurobi" indican que se ha utilizado como *solver* CBC o Gurobi para resolver los problemas de optimización intermedios. Si estos aparecen solos quiere decir que el algoritmo empleado es el B. Por su parte, "dynamic", "bundle" y "bundle_dynamic" se refieren a las ejecuciones realizadas con los algoritmos BD, BB y BBD respectivamente con los *solvers* indicados.

Figura 4.1: Distribución de la función objetivo relativa para todos los métodos (sin tándem).

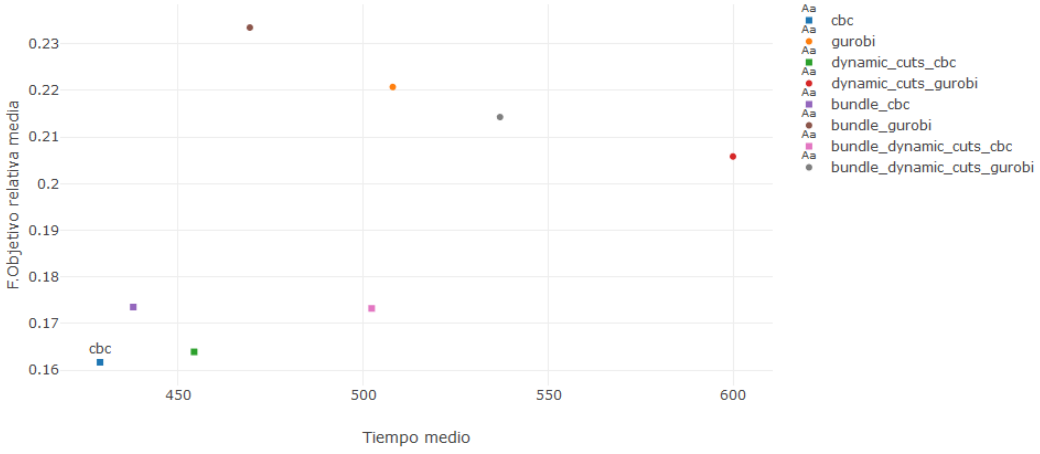


Con lo explicado hasta el momento se considerará que los métodos que tienen un mejor comportamiento en términos de optimalidad (mejor función objetivo) son aquellos que presentan una mayor concentración de datos cerca del cero. En la práctica nos interesan herramientas de optimización lo más robustas posibles en el sentido de que sean capaces de resolver cualquier problema dentro de una

clase de problemas, siendo en nuestro caso los problemas de pooling. En vista a los resultados para la función objetivo relativa obtenida por cada método, podemos decir que el mejor comportamiento se alcanza con los algoritmos: B con CBC, BD con CBC, BB con CBC y BBD con Gurobi.

El objetivo no solo es conseguir buenos óptimos locales para los problemas de pooling, sino también llegar a ellos en el menor tiempo posible. En la Figura 4.2 se presenta el rendimiento medio alcanzado con los métodos (sin tándem). Se compara la función objetivo relativa media obtenida por el método con el tiempo medio de resolución para todas las instancias y problemas. Se entiende que una herramienta de optimización tiene mejor rendimiento que otra si es capaz de conseguir mejores soluciones en menor tiempo. Entonces, si nos fijamos en la gráfica, los puntos que se encuentran abajo a la izquierda serán aquellos con mejor rendimiento mientras que los de arriba a la derecha serán los peores. Como se puede observar, en media, los mejores resultados se consiguen utilizando: B con CBC, BD con CBC y BB con CBC. La primera lección que podemos extraer con la representación gráfica anterior y con esta es que los métodos que presentan mayor robustez, en términos de función objetivo, son también los que tienen un mejor rendimiento computacional en media. En base a ello, respondiendo a la primera pregunta, podemos decir que los métodos con mejor rendimiento y que aportan unos mejores resultados en media son B con CBC, BB con CBC y BD con CBC.

Figura 4.2: Rendimiento en media para todos los métodos (sin tándem).

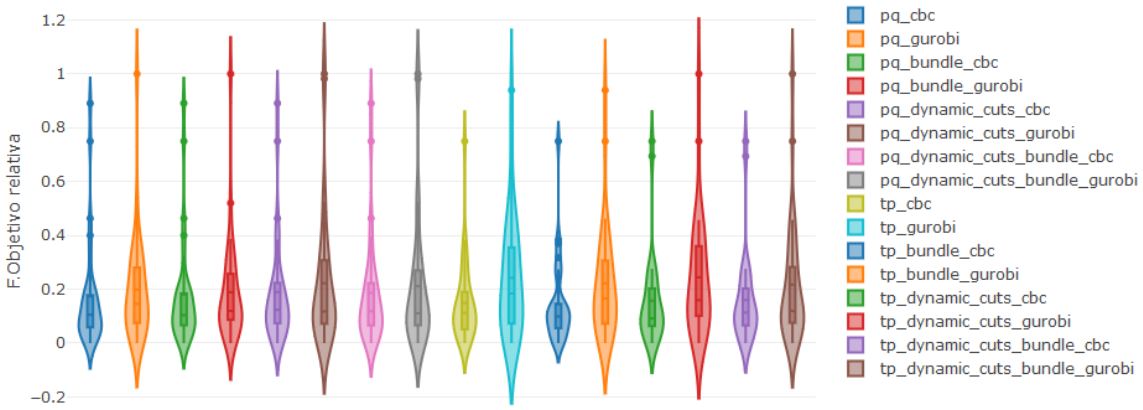


Tal y como se ha comentado en apartados anteriores, no solo el algoritmo utilizado para resolver los problemas de pooling es crítico a la hora de obtener buenos óptimos locales, sino también el modelado empleado para los mismos puede influir en la calidad de las soluciones obtenidas. Por ello, se realizará un análisis similar al anterior desagregando los resultados en función del modelo utilizado.

En la Figura 4.3 se presenta mediante el *violin* la distribución de los valores de la función objetivo relativa desagregando según el modelo utilizado. La interpretación de los resultados es análoga a la presentada previamente, la diferencia es que "pq" indica que el algoritmo y *solver* ha sido utilizado sobre el modelo PQ y "tp" lo mismo pero para el modelo TP. Nuevamente, los mejores resultados en términos de función objetivo relativa se han obtenido utilizando los métodos B, BD y BB, todos ellos con CBC. En cuanto al modelo empleado, no podemos ver una clara diferencia de un modelo respecto al otro. Parece que con el modelo TP los métodos comentados son más robustos ya que hay una mayor concentración de soluciones cercanas al cero. En concreto con el que mejores soluciones se obtienen, para las pruebas realizadas, es el modelo TP utilizando el algoritmo BB con CBC.

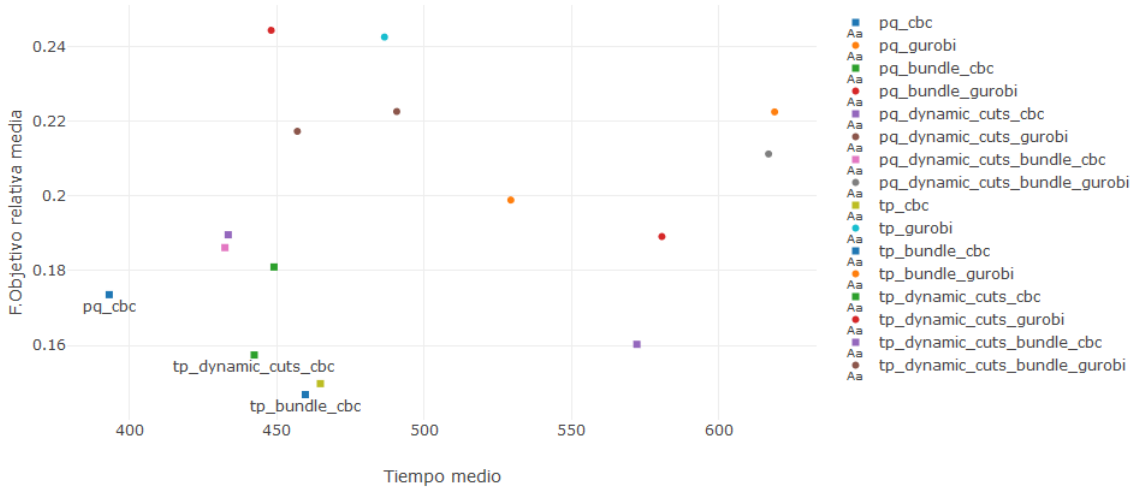
Teniendo en cuenta que la utilización de un modelo u otro influye a la hora de obtener mejores óptimos locales vamos a comprobar si también afecta al rendimiento medio de los algoritmos empleados.

Figura 4.3: Distribución de la función objetivo relativa para todos los métodos desagregando por modelo (sin tándem).



En la Figura 4.4 se puede ver que los mejores rendimientos medios en términos de función objetivo se alcanzan con el modelo TP utilizando B, BD y BB con CBC. Mientras que las soluciones más rápidas se alcanzan con el modelo PQ el algoritmo B con CBC. Pensando en una implementación para un proyecto de optimización de plantas de procesamiento industrial la elección de la formulación del modelo o del método elegido dependerá de si estamos dispuestos a esperar más tiempo para obtener unos mejores resultados o, por la contra, se prefiere sacrificar mejores óptimos locales pero consiguiendo resultados en menor tiempo. Independientemente del método o formulación elegida los mejores rendimientos medios se consiguen resolviendo los problemas intermedios con la herramienta de optimización gratuita CBC.

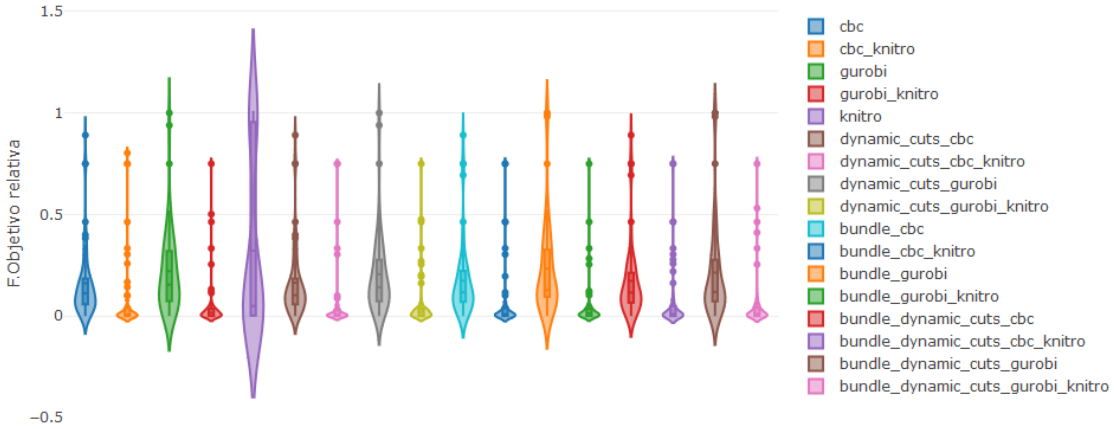
Figura 4.4: Rendimiento en media para todos los métodos desagregando por modelo (sin tándem).



Al trabajar con problemas no convexos que presentan multitud de óptimos locales nos puede interesar utilizar técnicas de optimización en tándem para tratar de explorar mejores puntos dentro de la región factible del modelo. En la Figura 4.5 se presenta la distribución de los resultados obtenidos utilizando los distintos métodos implementados, los tándem con Knitro y las soluciones obtenidas solo

utilizando Knitro. Se puede ver que aunque Knitro llega a los mejores resultados en términos de función objetivo relativa en algunos casos consigue las peores soluciones en 1200 segundos ya que no es capaz de dar con soluciones factibles para algunas instancias en este tiempo (Ver Apéndice B). Knitro es capaz de resolver las instancias pequeñas pero le cuesta resolver las instancias grandes. Por su parte, los métodos implementados si son capaces de conseguir soluciones factibles para todas las instancias con el límite de tiempo de 1200 segundos. Comparando nuestros mejores métodos (B, BB y BD con CBC) con Knitro podemos decir que los métodos basados en Benders presentan una mayor robustez para resolver los problemas de pooling ya que estos siempre consiguen un óptimo local con el límite máximo de tiempo de 1200 segundos. Sin embargo, cuando se utilizan los tándem con Knitro vemos como los resultados se concentran mucho más entorno al cero indicando que los mejores valores para la función objetivo en términos relativos se han obtenido con estos métodos. En concreto, los que mejor comportamiento presentan son los tándem utilizando los algoritmos B, BD y BB todos ellos con CBC. Con lo que si nuestro objetivo principal es conseguir los mejores óptimos se deben emplear los métodos en tándem.

Figura 4.5: Distribución de la función objetivo relativa para todos los métodos con tándem

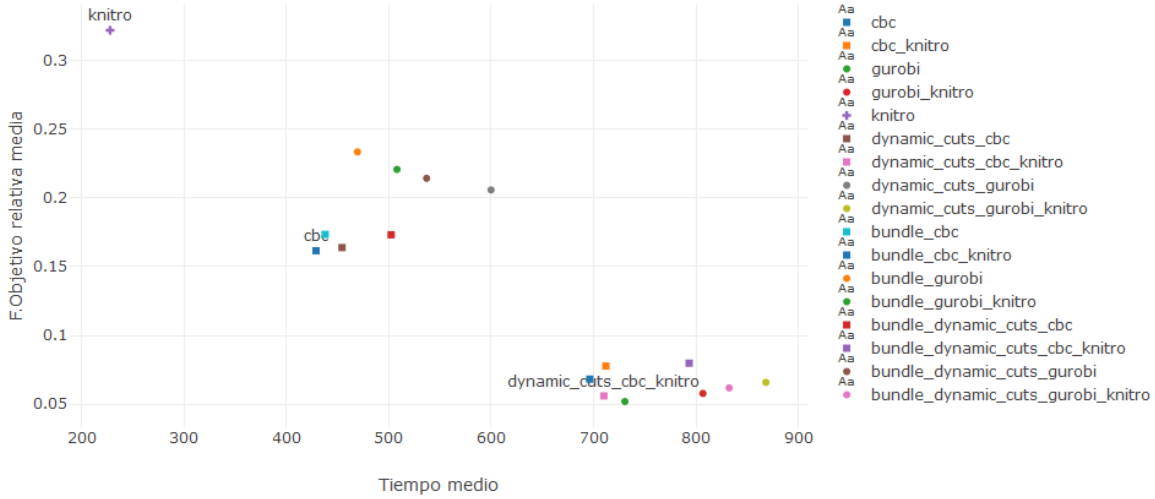


En cuanto al rendimiento comparando función objetivo relativa media y tiempos de ejecución medios para los métodos implementados, los tándem y Knitro se puede ver que este último es el que consigue unos tiempos de ejecución menores en media. Cabe decir que este es un *solver* comercial programado en el lenguaje de programación C (Kernighan & Ritchie 1978) y es sabido que se trata de un lenguaje con mejor rendimiento computacional que Python. Por su parte, los mejores valores medios para la función objetivo relativa se alcanzan siempre con los tándem, siendo los mejores los obtenidos con los métodos BBD, BD y BB con CBC como solver lineal.

- Si se quieren soluciones rápidas se puede utilizar Knitro, corriendo el riesgo puede no conseguir soluciones factibles en tiempos razonables para los problemas de pooling.
- Si se quiere asegurar la obtención de puntos factibles, consiguiendo un poco más de función objetivo en términos relativos y consumiendo un poco más de tiempo se puede utilizar el algoritmo B con CBC.
- Si lo deseado es conseguir los mejores óptimos locales, entonces se deben emplear los tándem con Knitro. En concreto, los algoritmos BB y BD con CBC o BB con Gurobi.

Lo último que se presentará en este capítulo es una comparativa de nuestros mejores métodos con los métodos heurísticos HRS y HRS_ALT presentados en Alfaki & Haugland (2014), Knitro y BARON para todas las instancias grandes. La idea del método HRS consiste en partir del problema de flujo en

Figura 4.6: Rendimiento en media para todos los métodos con tándem

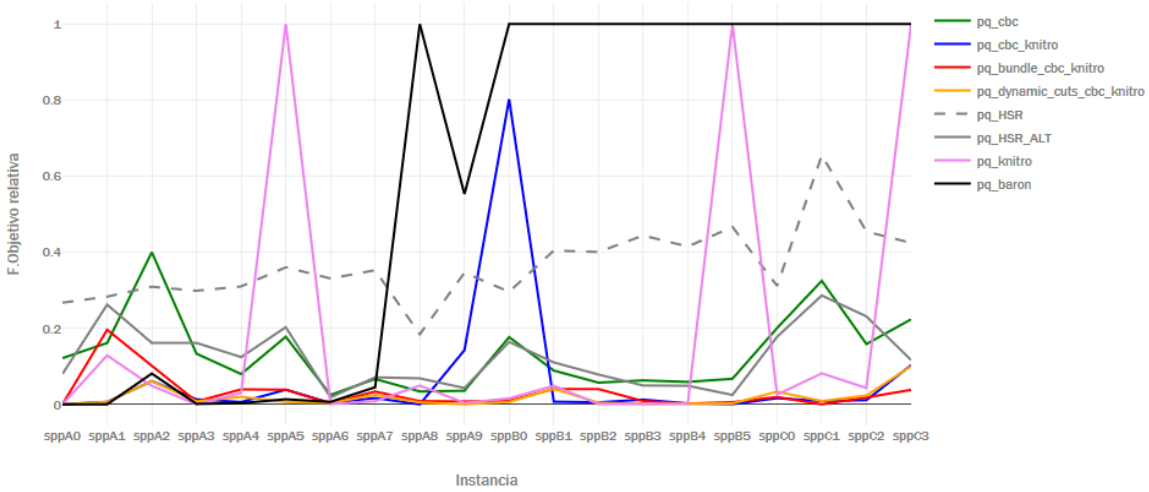


redes a coste mínimo resultante de no incluir los pools en la red e ir construyendo la red introduciendo los pools de forma iterativa resolviendo los respectivos problemas de optimización. Por su parte, el método HRS_ALT es un tándem que consiste en resolver el problema de pooling con HRS e inicializar el método ALT con la solución obtenida con el primero. El método ALT (Audet et al. 2004) consiste en hacer una partición del conjunto de variables de tal forma que si fijamos las variables del primer conjunto y las del segundo no, el problema resultante es lineal y viceversa. Este método va fijando de forma iterativa las variables del primer conjunto, resuelve el problema lineal, desfija las primeras y fija las del segundo resuelve el problema lineal... El proceso continúa hasta que se encuentre un óptimo local. Este se identifica cuando las soluciones del problema no cambian de una iteración a otra. Es importante destacar que los resultados obtenidos con nuestros métodos y las heurísticas de la literatura se pueden comprar ya que estas últimas siempre convergen antes del límite de tiempo prefijado por los autores, 1200 segundos. Aunque las máquinas en las que se han ejecutado son diferentes, como la nuestra presenta una potencia superior, en caso de ser ejecutados en nuestra máquina los resultados serían los mismo pero el tiempo de ejecución, posiblemente, fuese menor.

En la Figura 4.7 tenemos representados los valores alcanzados en términos de función objetivo relativa con nuestros mejores métodos, HSR, HRS_ALT, Knitro y BARON para todas las instancias grandes sobre el modelo PQ. Aquí solo se muestran resultados referentes a este modelo ya que es el único coincidente entre nuestros resultados y los expuestos en Alfaki & Haugland (2014). Se puede observar que, los tándem implementados alcanzan mejores soluciones que las obtenidas con HSR y HRS_ALT para todas las instancias. Si comparamos los resultados de aplicar unicamente el algoritmo B con CBC sobre las instancias grandes vemos como este mejora en todos los casos a HSR excepto en la instancia sppA2 e incluso es competente con el tándem HRS_ALT en térnimos de función objetivo relativa. Este hecho es relevante ya que los métodos HSR y HRS_ALT están expresamente diseñados para los problemas de pooling mientras que los métodos implementados son mucho más generales. Además, es importante destacar que B con CBC es directamente comparable con HSR ya que no son métodos en tándem mientras que HRS_ALT sí lo es. Si se hace una comparativa directamente con los *solvers* comerciales se observa como hay una mayor robustez, a la hora de resolver los problemas de pooling, con los métodos implementados que con los solver comerciales. Además, vemos que para algunas instancias estos *solvers* comerciales no son capaces de encontrar una solución factible del problema en menos de 1200 segundos, mientras que con nuestros métodos siempre se encuentran soluciones factibles en el límite de tiempo. Finalmente, si nos tuviéramos que decantar por la elección

de uno de los métodos para resolver los problemas de pooling elegiríamos los tándem BB o BD con CBC utilizando Knitro en la segunda pasada.

Figura 4.7: Comparativa con la literatura, Knitro y BARON de los resultados obtenidos con los mejores métodos sobre el modelo PQ.



Capítulo 5

Conclusiones

El objetivo principal de este trabajo fue la implementación de cuatro variantes del método de descomposición de Benders generalizado que permiten encontrar buenos óptimos locales para los problemas de pooling en un tiempo de ejecución razonable. Ha quedado de manifiesto que resolver este tipo de problemas, cuando el tamaño de los mismos es elevado, es una tarea muy difícil. Se ha visto como el *solver* comercial Knitro, para algunas de las instancias grandes, no es capaz de conseguir ni una solución factible en menos de 1200 segundo. El comportamiento aún es peor en el caso del *solver* global BARON ya que en 11 de las 20 instancias grandes con el modelo PQ no es capaz de encontrar una solución factible en el límite de tiempo permitido. En cambio, las técnicas de descomposición implementadas consiguen, para todos los problemas e instancias soluciones factibles antes de este límite de tiempo. Con lo que queda de manifiesto que la utilización de los métodos implementados pueden ser de gran ayuda para empresas de las industrias petroquímicas y gasísticas. Además, ha resultado que incluso técnicas diseñadas específicamente para encontrar óptimos locales en los problemas de pooling como HSR y HRS_ALT consiguen peores óptimos locales que los conseguidos con las mejores técnicas implementadas, siendo estas mucho más generales.

De los métodos implementados se puede concluir que los que mejor comportamiento tienen, en general, tanto en términos de optimalidad como en tiempos de ejecución son B, BD, y BB todos ellos con CBC. Con lo que se tiene una herramienta totalmente gratuita y competente para resolver la familia de problemas consideradas en este trabajo. Si se utilizan, estas técnicas en tándem los resultados son competentes e incluso superando a los obtenidos con herramientas comerciales.

En cuanto a la formulación matemática del problema de optimización hay diferencias de comportamiento entre la formulación PQ y TP, pero los resultados obtenidos también dependen del método utilizado para resolverlos. No se puede concluir que uno se comporte mejor que el otro de manera clara en términos medios. El modelo TP con los algoritmos B, BB con CBC o BBD con Gurobi es el que consigue un mejor rendimiento medio en términos de función objetivo relativa. Sin embargo, si lo que se quieren son resultados más rápidos se podría utilizar el modelo PQ y el algoritmo B con CBC.

Lo siguiente a destacar es que se puede tener una herramienta para resolver problemas de optimización con características similares a los de pooling y competente con los *solvers* comerciales de forma gratuita. Es decir, si los solver utilizados tanto para resolver los problemas intermedios como los tándem son gratuitos el coste monetario de la herramienta será nulo.

Lo novedoso de este trabajo ha sido la formulación del subproblema para los problemas de pooling presentada en la subsección 3.3. Se ha conseguido remediar el problema de que los cortes de Benders se anulen cuando las restricciones están débilmente activas en un punto sin aumentar la complejidad

computacional del modelo. La forma de hacerlo ha sido formular el subproblema con todas sus restricciones como menor o igual y utilizar una única variable de holgura penalizada en función objetivo sobre aquellas restricciones en las que aparecen implicadas variables complicantes.

A modo aclaratorio, los cortes dinámicos fueron pensados originalmente para ser utilizados en el método de Benders sobre problemas de programación estocástica (Soumis et al. 2012). Cuando se trabaja con problemas estocásticos con el algoritmo de Benders, se puede tener multitud de subproblemas y un único problema maestro que es el encargado de coordinar todos los subproblemas. Lo que ocurre es que en vez de añadir un corte en cada iteración del algoritmo se añaden tantos cortes como subproblemas se tengan. Esto provoca que el crecimiento del problema maestro sea muy rápido. En los problemas de pooling (deterministas) se comprobó que en muchas instancias el número de iteraciones para que el algoritmo convergiera era elevado, se decidió implementar esta heurística para facilitar la resolución del problema maestro. En los resultados empíricos resultó que la heurística sobre el método de Benders generalizado aplicado sobre los problemas de pooling funciona realmente bien. Incluso permite explorar puntos de la región factible a los que el método de Benders generalizado convencional no es capaz de llegar, en algunos casos.

Los *bundle methods* utilizados en el algoritmo de Benders, fueron originalmente pensados para ser aplicados sobre problema convexos. Basándonos en el hecho de que los problemas de pooling presentan multitud de óptimos locales se han adaptado con el objetivo de acelerar el procesos de búsqueda de óptimos locales. La idea es que si queremos obtener óptimos locales de forma rápida una forma sencilla de hacerlo es mantenerse cerca del mejor óptimo encontrado en el proceso iterativo, permitiendo movimiento cada vez que se encuentra un nuevo óptimo local mejor. En los resultados empíricos no resultó ser el método más rápido pero sí que es uno de los que mejores óptimos locales consigue.

De cara a un posible trabajo futuro, la herramienta de optimización implementada puede ser mejorada mediante la construcción de un multiarranque de manera sencilla. Durante el proceso iterativo de los algoritmos implementados se recorren multitud de óptimos locales del problema original para finalmente devolver el mejor. Es fácil ver que en dicho proceso iterativo se pueden guardar, si se quiere, otros óptimos locales. Con el algoritmo de Benders, o las modificaciones implementadas, se podría, por ejemplo, guardar un número prefijado de óptimos locales que se hayan encontrado durante el proceso iterativo y así inicializar nuevamente el algoritmo con las soluciones encontradas. Además, si este último paso lo realizamos en paralelo el tiempo computacional no será muy diferente al consumido la primera vez que se empleó el método.

Apéndice A

Nociones básicas de teoría de grafos y redes con flujo

A.1. Nociones básicas de teoría de grafos

Un *grafo* D es un par (N, A) consistente en un conjunto N de elementos llamados *nodos* o *vértices* y un conjutno A cuyos elementos representan *arcos* o *aristas*. Según cómo sean los elementos de A se pueden distinguir dos tipos de grafos:

- **Grafos orientados:** Un grafo *orientado* o *dirigido* es aquel en el que $A \subset N \times N$, es decir, los arcos son pares ordenados; el arco (i, j) empieza en el nodo i y termina en el nodo j.
- **Grafos no orientados:** En un *grafo* no *orientado* A está compuesto por subconjuntos de N de dos elementos. En este caso, como $\{i, j\}$ y $\{j, i\}$ son el mismo conjunto, representarán el mismo arco.

Gráficamente, el carácter orientado o no orientado de un grafo se representará mediante presencia o ausencia de flechas. Se usará n y m para referisse al número total de nodos y aristas respectivamente.

Un grafo es *completo* si todas las aristas posibles están presentes. En el caso de un grafo orientado el máximo número de aristas en $n(n-1)$ porque no se permiten lazos en este documento y, en el caso de un grafo no orientado, se tendrá la mitad, $n(n-1)/2$. Un grafo es *plano* si puede dibujarse en un plano sin que sus aristas se corten.

Un *subgrafo* de D es un grafo $D' = (N', A')$ que tiene todos sus vértices y aristas en D, es decir, $N' \subseteq N$ y $A' \subseteq A$. Un subgrafo de *expansión* de D es un subgrafo $D' = (N', A')$ tal que $N' = N$. Si la arista (i, j) está presente en un grafo D, decimos que los nodos i y j son *adyacentes* y también que son *incidentes* con las aristas (i, j) y que la arista (i, j) es *incidente* con los nodos i y j. El grado de un nodo cualquiera de un grafo D es el número de aristas incidentes con él, es decir, es el número de aristas que entras y salen de ese nodo. Sea D un grafo *no orientado* y sea $(a_1, a_2, ..., a_r)$ y una secuencia de aristas de D. Si existen vértices $v_0, v_1, ..., v_r$ tales que, para $l \in \{1, 2, ..., r\}$, $a_l = (v_{l-1}, v_l)$, decimos que la secuencia es una cadena. Para referirnos a una cadena se usará indistintamente la secuencia de aristas o la secuencia de nodos que la forma. Se tienen los siguientes tipos de cadenas:

- **Cadena cerrada:** Una cadena en la que $v_0 = v_r$
- **Camino:** Una cadena en la que todos los vértices sin distintos.

- **Circuito o ciclo:** Una cadena cerrada en la que no hay más nodos coincidentes que el primero y el último.

Un grafo se dice que es *conexo* si para cada par de vértices existe una cadena no orientada que los une. Se dice que es *fuertemente conexo* si para cada par de vértices existe una cadena orientada que los une. Un grafo se dice que es un árbol si es conexo y no contiene ciclos (no orientados). Un árbol de *expansión* de D es un árbol que es un subgrafo de expansión de D. Dicho árbol será un subgrafo conexo minimal en el sentido de que no habrá otro subgrafo conexo con menos aristas.

A.2. Redes con flujo

Una red es un grafo con uno o más números asociados con cada arco o nodo. Estos números pueden representar distancias, costes, fiabilidades u otros parámetros de interés. Llamaremos flujo al envío de elementos u objetos de un lugar a otro dentro de una red. Por ejemplo, el transporte de crudos o derivados de los mismos a través de tuberías en una refinería, el traslado de personas desde su domicilio a su lugar de trabajo, el trasporte de gas a través de una red de gas... Los modelos correspondientes a estas situaciones los llamaremos *modelos de redes con flujo*.Por ejemplo, podríamos estar interesados en minimizar el coste de envío de productos a través de las tuberías en una refinería de petróleo.

Los objetos que "viajan" o fluyen por la red de llaman *unidades de flujo* o simplemente *unidades*. Las unidades de flujo pueden ser bienes, personas, información o casi cualquier cosa. La red con flujo es un grafo orientado. De forma general llamaremos f_k al flujo que pasa por el arco k. A cada arco k se le asignarán tres parámetros:

- **Cota inferior,** $l_k \geq 0$**:** Cantidad mínima de flujo que debe pasar por el arco k.
- **Capacidad o cota superior,** $u_k \geq 0$**:** Cantidad máxima de flujo que el arco k puede soportar.
- **Coste o beneficio,** c_k : Si es negativo denota el coste de unidad de flujo que pasa por el arco k; si es positivo denota beneficios.

El arco k también se puede denotar como (i, j) (siendo i y j los nodos incidentes en el arco k); en este caso, también se denotará los parámetros asociados como l_{ij}, u_{ij} y c_{ij}. Entonces una red con flujo R vendrá caracterizada por una tupla $((N, A), (l, u, c))$, donde (N, A) es el grafo subyacente a la red y (l, u, c) serán las capacidades o costes de los arcos.

Apéndice B

Tablas de resultados

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	HSR	HSR+ALT	Global
Adhya1	-462.5	-462.5	-462.5	-462.5	-549.8	-549.8	-549.8	-549.8	-340.9	-541.5	-541.5	-549.8
Adhya2	-60.0	-60.0	-60.0	-60.0	-549.8	-549.8	-549.8	-549.8	–	-541.4	-541.4	-549.8
Adhya3	-490.5	-491.5	-502.8	-491.4	-561.0	-561.0	-559.6	-561.0	–	-552.9	-552.9	-561.1
Adhya4	-470.8	-470.8	-470.8	-470.8	-470.8	-470.8	-470.8	-470.8	-877.6	-877.6	-877.6	-877.6
Bental4	-416.7	-416.7	-416.7	-416.7	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0
Bental5	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0
Haverly1	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-400.0	-400.0	-400.0	-400.0
Haverly2	-600.0	-600.0	-600.0	-600.0	-600.0	-600.0	-600.0	-600.0	-400.0	-600.0	-600.0	-600.0
Haverly3	-700.0	-700.0	-700.0	-700.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0

Tabla B.1: Resultados de la herramienta para el modelo PQ utilizando CBC y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	HSR	HSR+ALT	Global
Adhya1	-1.9e-06	-1.9e-06	-1.9e-06	-1.9e-06	-549.8	-549.8	-549.8	-549.8	-340.9	-541.5	-541.5	-549.8
Adhya2	-462.5	-10.1	-462.5	-10.1	-549.8	-549.8	-549.8	-549.8	–	-541.4	-541.4	-549.8
Adhya3	-512.4	-503.3	-512.4	-503.3	-561.045	-559.6	-561.0	-559.6	–	-552.85	-552.85	-561.05
Adhya4	-421.1	-421.1	-421.1	-421.1	-470.8	-470.8	-470.8	-470.8	-877.6	-877.6	-877.6	-877.6
Bental4	-435.7	-400.0	-435.7	-416.7	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0
Bental5	-3090.6	-3247.1	-3090.6	-3247.1	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0
Haverly1	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0	-400.0
Haverly2	-600.0	-600.0	-600.0	-600.0	-600.0	-600.0	-600.0	-600.0	-400.0	-600.0	-600.0	-600.0
Haverly3	-700.0	-700.0	-700.0	-700.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0

Tabla B.2: Resultados de la herramienta para el modelo PQ utilizando Gurobi y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	HSR	HSR+ALT	Global
Adhya1	-509.8	-509.8	-509.8	-509.8	-549.8	-549.8	-549.8	-549.8	-22.7	-541.5	-541.5	-549.8
Adhya2	-509.8	-509.8	-509.8	-509.8	-549.8	-549.8	-549.8	-549.8	-21.8	-541.4	-541.4	-549.8
Adhya3	-509.8	-509.8	-509.8	-509.8	–	–	–	–	-27.4	-552.9	-552.9	-561.1
Adhya4	-541.0	-268.3	-541.0	-268.3	-877.6	-470.8	-877.6	-470.8	-122.4	-877.6	-877.6	-877.6
Bental4	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	4.7	-450.0	-450.0	-450.0
Bental5	-3201.3	-3500.0	-3450.1	-3500.0	–	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0
Haverly1	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-1.2	-400.0	-400.0	-400.0
Haverly2	-600.0	-600.0	-600.0	-600.0	-400.0	-400.0	-400.0	-400.0	0.8	-600.0	-600.0	-600.0
Haverly3	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0

Tabla B.3: Resultados de la herramienta para el modelo TP utilizando CBC y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	HSR	HSR+ALT	Global
Adhya1	-509.8	-509.8	-509.8	-509.8	-549.8	-549.8	-549.8	-549.8	-22.7	-541.5	-541.5	-549.8
Adhya2	-33.3	-4.2e-08	-33.3	-4.2e-08	-549.8	-549.8	-549.8	-549.8	-21.8	-541.4	-541.4	-549.8
Adhya3	-509.8	-509.8	-533.3	-509.8	-552.9	-552.9	-559.6	-552.9	-27.4	-552.9	-552.9	-561.1
Adhya4	-755.1	-505.6	-755.1	-505.6	-877.6	-877.6	-877.6	-877.6	-122.4	-877.7	-877.7	-877.7
Bental4	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	-450.0	4.7	-450.0	-450.0	-450.0
Bental5	-3468.4	-3039.6	-3256.5	-3022.7	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0	-3500.0
Haverly1	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-1.2	-400.0	-400.0	-400.0
Haverly2	-600.0	-600.0	-600.0	-600.0	-400.0	-400.0	-400.0	-400.0	0.9	-600.0	-600.0	-600.0
Haverly3	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0	-750.0

Tabla B.4: Resultados de la herramienta para el modelo TP utilizando Gurobi y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	BARON	HSR	HSR+ALT
sppA0	-31459.7	-27884	-28203.4	-27841.3	-35812.3	-35812.3	-35812.3	-35812.3	-35812.3	-35812.3	-26238.2	-32947.2
sppA1	-24554.7	-22733.1	-24554.7	-21769.9	-29084.9	-23535.8	-29084.9	-29084.9	-25509.9	-29276.6	-20981.7	-21601.2
sppA2	-13823	-14181	-13823	-13923.8	-21627.7	-20704.2	-21627.7	-16647.1	-21922.5	-21176.4	-15916.8	-19317.9
sppA3	-33519.1	-35016.2	-31579.7	-35537.9	-38159.5	-38373.9	-38373.9	-28681.2	-38669.6	-38622.9	-27117.4	-32423.1
sppA4	-37968.1	-36405.9	-36988.7	-37540.7	-41005.3	-39629.4	-40431.9	-40436.4	-40008.8	-41127	-28464.6	-36130.4
sppA5	-22746.7	-23472	-23272.2	-24050.2	-26598.4	-26598.4	-27488.2	-25034.8	–	-27293.2	-17694.6	-22052.9
sppA6	-41068.4	-41251.2	-41108.2	-41158.5	-41961.5	-42002.7	-41956.2	-42111.8	-42075.4	-41839.4	-28171.8	-41341.0
sppA7	-41328.1	-39270.2	-39910.7	-39270.2	-43549	-42794.6	-43183.2	-42794.6	-43922.8	-42292.2	-28667.9	-41135.8
sppA8	-29485	-29067.8	-29880.1	-29067.8	-30518.9	-30258.8	-30363.5	-30258.8	-29011.6	-	-24893.8	-28431.9
sppA9	-21127.7	-18850.4	-19749.3	-17744.9	-18787	-21742.8	-21908.3	-21832.6	-21818.6	-9803.3	-14324.1	-20969.0
sppB0	-34388.9	-31084.6	-34846.1	-31084.6	-8243.39	-41365.4	-41516.1	-41365.4	-41121.4	-	-29410.1	-34917.1
sppB1	-56780.3	-54630.6	-56929.5	-57134.5	-61877.2	-59799.8	-59756.7	-61619.7	-59265	-	-37111.8	-55442.8
sppB2	-50738.6	-49743.7	-50598.9	-48475.6	-53527	-51653.3	-53574.5	-53165.8	-53765.4	-	-32250.1	-49594.5
sppB3	-69224.7	-67661.6	-68538.9	-70323.9	-72959.9	-73228.4	-73819	-73438.8	-73723.1	-	-41081.4	-70211.6
sppB4	-55920.3	-56724.6	-56372.6	-56494.5	-59292.3	-59313.6	-59335.4	-59346.3	-59320.8	-	-34765.8	-56520.4
sppB5	-56369.4	-55926.3	-54929.2	-56439.5	-60394.9	-60088.8	-60284.9	-56378.4	–	-	-32241.2	-58942.9
sppC0	-67494.2	-65533	-68847.3	-67444.8	-83066.6	-82802.8	-81609.6	-83049.1	-82276.3	-	-58124.0	-69431.5
sppC1	-68103.7	-62517	-68103.7	-69093	-99986.1	-100864	-99986.1	-96278.6	-92649.2	-	-34775.8	-71995.2
sppC2	-102597	-100317	-100284	-100317	-120584	-119728	-119094	-119272	-116695	-	-66532.9	-93738.5
sppC3	-93706.2	-98584.5	-93706.2	-97891.5	-108252	-116089	-108587	-114387	–	-	-69465.0	-106645.6

Tabla B.5: Resultados de la herramienta para el modelo PQ utilizando CBC y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	BARON	HSR	HSR+ALT
sppA0	-29520.6	-17042	-31304.1	-17042	-34882.8	-32154	-34576.9	-32154	-35812.3	-35812.3	-26238.2	-32947.2
sppA1	-19046.9	-19429.9	-22060.9	-18420.5	-25403.8	-28736.5	-26732.8	-26806.3	-25509.9	-29276.6	-20981.7	-21601.2
sppA2	-15368.2	-15106.4	-16944.7	-17320.3	-21922.5	-21832.9	-22325.2	-22836.5	-21922.5	-21176.4	-15916.8	-19317.9
sppA3	-34006.6	-29399.4	-33964.6	-35169.7	-36974.4	-38494.3	-37245.7	-38646.1	-38669.6	-38622.9	-27117.4	-32423.1
sppA4	-35424.3	-36359.4	-36403.2	-35873	-41257.1	-39378.4	-39534.8	-38885.1	-40008.8	-41127	-28464.6	-36130.4
sppA5	-22310.1	-24590.3	-23428	-24079.1	-26598.4	-26900.8	-27032.1	-26741.6	–	-27293.2	-17694.6	-22052.9
sppA6	-40739.4	-40802.7	-40953.2	-41733.7	-42092.3	-39976	-41961.5	-41961.5	-42075.4	-41839.4	-28171.8	-41341.0
sppA7	-39415.7	-40086	-39495.1	-39980.2	-43627.9	-43560.7	-43155.5	-43620.3	-43922.8	-42292.2	-28667.9	-41135.8
sppA8	-29653.6	-29009.2	-27755.1	-29009.2	-30012.9	-30167.1	-15999.3	-30167.1	-29011.6	-	-24893.8	-28431.9
sppA9	-19267.5	-19367.2	-19995.2	-21324.1	-21570.2	-21451.8	-21381.1	-21720.2	-21818.6	-9803.3	-14324.1	-20969.0
sppB0	-30223.2	-36666.6	-31026.7	-34345.9	-37021.7	-40496.9	-41207.7	-41774.8	-41121.4	-	-29410.1	-34917.1
sppB1	-56410	-56206.6	-56087.9	-55310.7	-60606.5	-59740.1	-59612.2	-62314.7	-59265	-	-37111.8	-55442.8
sppB2	-40914.6	-47787.6	-43166	-44343.9	-52843.6	-50994.6	-51936.3	-52899.2	-53765.4	-	-32250.1	-49594.5
sppB3	-62792.3	-64714.6	-65381	-66307	-73669.6	-73558.5	-71946.4	-73875.6	-73723.1	-	-41081.4	-70211.6
sppB4	-54981.1	-55293.5	-55910	-56473.9	-59256.3	-59422.5	-59278.4	-34977.5	-59320.8	-	-34765.8	-56520.4
sppB5	-46576.5	-51278.4	-47071.3	-53803.6	-60164.5	-60412.6	-58380.6	-59383.5	–	-	-32241.2	-58942.9
sppC0	-54304	-59803.1	-55832.6	-61643.7	-79151.2	-83201.9	-81499.5	-80014.3	-82276.3	-	-58124.0	-69431.5
sppC1	-68554.1	-68926.9	-61769.1	-72783.3	-97580.4	-99101.9	-99105.8	-96523.1	-92649.2	-	-34775.8	-71995.2
sppC2	-84440.9	-93635.8	-81135.5	-90358.8	-118770	-119751	-121939	-117563	-116695	-	-66532.9	-93738.5
sppC3	-86819.1	-83558.3	-85942.6	-87453.1	-116834	-116878	-120686	-117616	–	-	-69465.0	-106645.6

Tabla B.6: Resultados de la herramienta para el modelo PQ utilizando Gurobi y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	BARON
sppA0	-30896.8	-33537.8	-32174.6	-31405.3	-35018.1	-35727.1	-35727.1	-34784.3	-35727.1	-35812.3
sppA1	-25109.1	-25109.1	-25109.1	-25109.1	-28900.4	-29084.4	-28900.4	-29084.4	-26732.8	-29276.6
sppA2	-16730	-16730	-16730	-16730	-23042	-23042	-23042	-23042	-16697.9	-22325.2
sppA3	-33834.9	-35342.4	-33827.8	-34630.8	-37414.9	-37978.5	-37245.7	-38646.1	-38494.3	-33320.1
sppA4	-35283.5	-36417.5	-35283.5	-36417.5	-39378.4	-38230.5	-39378.4	-38230.5	-40806.5	-29758
sppA5	-22407.1	-23449.6	-23913.3	-23927.9	-26621.2	-27033.4	-26492.9	-23171.8	–	-23656.2
sppA6	-40416.9	-39818	-38076.2	-39931.2	-42077.1	-42112	-42087.8	-42111.8	-40527	-39840.1
sppA7	-36879	-37602	-40073.7	-38682.2	-42999	-43620.3	-43496.2	-44285.1	-34141.9	-32182.8
sppA8	-29931.6	-29701.3	-29947.4	-29820.2	-30302.1	-30209.1	-30389	-23809.8	-27339.1	-19411.7
sppA9	-20554.6	-20275.9	-19125	-20468	-16244.6	-21804.4	-21821.8	-21911	–	-13711.5
sppB0	-35807.4	-33345	-35807.4	-33653.6	-41003.7	-41132.2	-41003.7	-40069.7	–	-29715.9
sppB1	-56033.5	-56689.2	-58813.8	-55450.1	-59873.5	-58447.4	-61834.8	-59647.5	-56116	-18821
sppB2	-42041	-42896.4	-40585.9	-42896.4	-52017.1	-53305.7	-52053	-53305.7	-53812.3	-
sppB3	-70052.7	-69793.6	-70524.9	-69793.6	-73866.7	-73005.3	-73627.3	-73005.3	-68790	-
sppB4	-53271.5	-50760	-53873.8	-49284.5	-59430.6	-59249.4	-59363.6	-59397.5	-56923.1	-28163
sppB5	-57458.5	-55263.9	-57070.3	-54946.7	-60433.1	-60257.6	-60339.8	-59574.7	-58225.6	-
sppC0	-58966	-65246.2	-57636.1	-63382	-84224.7	-84436.7	-78932.9	-83374.9	-68175.1	-
sppC1	-65490.6	-73006.1	-63542.1	-73006.1	-90843.9	-97093.8	-99567	-97093.8	-87002.6	-
sppC2	-90818	-89479.8	-88716	-89479.8	-101544	-121158	-111230	-121157	–	-
sppC3	-106048	-89768.6	-106048	-89768.6	-120074	-106629	-119763	-106684	–	-

Tabla B.7: Resultados de la herramienta para el modelo TP utilizando CBC y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

	B	BB	BD	BBD	BK	BBK	BDK	BBDK	K	BARON
sppA0	-33922	-31815.5	-30643.9	-31971	-34723.1	-35812.3	-34883.9	-35727.1	-35727.1	-35812.3
sppA1	-22511.8	-22511.8	-22511.8	-22511.8	-29207.4	-28133.9	-29207.4	-28133.9	-26732.8	-29276.6
sppA2	-11267.9	-12504.1	-16950.5	-12797.6	-20265.3	-16507.4	-18578.2	-21803.6	-16697.9	-22325.2
sppA3	-31621.9	-34356.4	-31621.9	-35929.4	-37821.2	-38646.1	-37821.2	-38368.4	-38494.3	-33320.1
sppA4	-28137.7	-29883.2	-33326.8	-35052.3	-40937.9	-40694.3	-39193.5	-39044.7	-40806.5	-29758
sppA5	-17065.8	-24334.2	-24422.1	-24334.2	-27488.2	-27663.4	-26940.6	-27663.4	–	-23656.2
sppA6	-39069.7	-35603.9	-39069.7	-39365.2	-42111.8	-42112	-42111.8	-42111.9	-40527	-39840.1
sppA7	-39741.9	-38143.8	-39703.4	-39861.7	-43081.6	-43416.2	-43184.4	-43564.4	-34141.9	-32182.8
sppA8	-29212.2	-29432.7	-29330.4	-29459.7	-30015.6	-30233.5	-25589.5	-30332.5	-27339.1	-19411.7
sppA9	-19362.2	-19677.8	-20379.2	-19912.1	-21911	-21808.4	-21580.3	-21355.8	–	-13711.5
sppB0	-26942.4	-22640.8	-26942.4	-22640.8	-41031.6	-41226.7	-41031.6	-41226.7	–	-29715.9
sppB1	-50733.6	-56362.1	-50733.6	-56231.1	-58259.9	-59962.4	-58259.9	-61210.6	-56116	-18821
sppB2	-35700.7	-34444.4	-35700.7	-38642.2	-53346.6	-53334.1	-53346.6	-51725.5	-53812.3	-
sppB3	-65064.5	-61726.4	-62803.1	-68291.7	-73678.5	-64773.6	-73723.2	-73708.3	-68790	-
sppB4	-51906.7	-52927.9	-51688.6	-52415.5	-56711.7	-59240	-47409.9	-59107.4	-56923.1	-28163
sppB5	-47091.2	-48064.6	-47325.4	-53283.4	-58682.6	-60263.4	-60059	-60213.8	-58225.6	-
sppC0	-45414.1	-60598.1	-45414.1	-60598.1	-81776.4	-82359.3	-81776.4	-82359.3	-68175.1	-
sppC1	-55969.2	-66085.6	-69962.2	-62961.7	-98788.2	-98083.9	-99208.6	-93443.4	-87002.6	-
sppC2	-85860.4	-73833.3	-74348.8	-92612.5	-116426	-107873	-119255	-115315	–	-
sppC3	-70837.2	-71554.9	-72009.2	-81260.6	-60162.6	-118732	-88075.1	-56588	–	-

Tabla B.8: Resultados de la herramienta para el modelo TP utilizando Gurobi y comparando con resultados de la literatura y Knitro. El símbolo - indica que no se ha obtenido una solución factible en menos de 1200 segundo.

Bibliografía

Alfaki, M. (2012), Models and solution methods for the pooling problem.

Alfaki, M. & Haugland, D. (2013), 'Strong formulations for the pooling problem', *Journal of Global Optimization* **56**(3), 897–916.
URL: *https://doi.org/10.1007/s10898-012-9875-6*

Alfaki, M. & Haugland, D. (2014), 'A cost minimization heuristic for the pooling problem', *Annals of Operations Research* **222**(1), 73–87.
URL: *https://doi.org/10.1007/s10479-013-1433-1*

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S. & Mladenović, N. (2004), 'Pooling problem: Alternate formulations and solution methods', *Management Science* **50**, 761–776.

Bazaraa, M. S., Sherali, H. D. & Shetty, C. M. (2005), *Nonlinear Programming*, John Wiley & Sons, Inc.
URL: *https://doi.org/10.1002/0471787779*

Benders, J. F. (1962), 'Partitioning procedures for solving mixed-variables programming problems', *Numerische Mathematik* **4**(1), 238–252.
URL: *https://doi.org/10.1007/BF01386316*

Birgin, E., A. Floudas, C. & MartÃnez, J. M. (2010), 'Global minimization using an augmented lagrangian method with variable lower-level constraints', *Mathematical Programming* **125**, 139–162.

Byrd, R. H., Nocedal, J. & Waltz, R. A. (2006), *Knitro: An Integrated Package for Nonlinear Optimization*, Springer US, Boston, MA, pp. 35–59.
URL: *https://doi.org/10.1007/0-387-30065-1_4*

Dai, Y.-H., Diao, R. & Fu, K. (2018), 'Complexity analysis and algorithm design of pooling problem', *Journal of the Operations Research Society of China* **6**(2), 249–266.
URL: *https://doi.org/10.1007/s40305-018-0193-7*

Dantzig, G. B. & Wolfe, P. (1960), 'Decomposition principle for linear programs', *Operations Research* **8**(1), 101–111.
URL: *https://doi.org/10.1287/opre.8.1.101*

Floudas, C. A. (1995), *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications (Topics in Chemical Engineering)*, Topics in Chemical Engineering, Oxford University Press, USA.
URL: *http://gen.lib.rus.ec/book/index.php?md5=96B60AE93265BB493867408D3325FBC2*

Floudas, C. A. & Aggarwal, A. (1990), 'A decomposition strategy for global optimum search in the pooling problem', *ORSA Journal on Computing* **2**(3), 225–235.
URL: *https://doi.org/10.1287/ijoc.2.3.225*

Forrest, J., Ralphs, T., Vigerske, S., LouHafer, Kristjansson, B., Jpfasano, EdwinStraver, Lubin, M., Haroldo Gambini Santos, Rlougee & Saltzman, M. (2018), 'Coin-or/cbc: Version 2.9.9'.
URL: *https://zenodo.org/record/1317566*

Geoffrion, A. (1972), 'Generalized benders decomposition', *Journal of Optimization Theory and Applications* **10**, 237–260.

Grothey, A., Leyffer, S. & Mckinnon, K. I. M. (2000), 'A note on feasibility in benders decomposition'.

Gurobi Optimization, L. (2019), 'Gurobi optimizer reference manual'.
URL: *http://www.gurobi.com*

Hart, W. E., Laird, C. D., Watson, J.-P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L. & Siirola, J. D. (2017), *Pyomo–optimization modeling in python*, Vol. 67, second edn, Springer Science & Business Media.

Haverly, C. A. (1978), 'Studies of the behavior of recursion for the pooling problem', *SIGMAP Bull.* (25), 19–28.
URL: *http://doi.acm.org/10.1145/1111237.1111238*

Kelley, J. E. (1960), 'The cutting-plane method for solving convex programs', *Journal of the Society for Industrial and Applied Mathematics* **8**(4), 703–712.
URL: *http://www.jstor.org/stable/2099058*

Kernighan, B. W. & Ritchie, D. M. (1978), *The C Programming Language*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Land, A. H. & Doig, A. G. (1960), 'An automatic method of solving discrete programming problems', *Econometrica* **28**(3), 497.
URL: *https://doi.org/10.2307/1910129*

Lemaréchal, C., Nemirovski, A. & Nesterov, Y. (1995), 'New variants of bundle methods', *Math. Program.* **69**, 111–147.

Palacios-Gomez, F., Lasdon, L. & Engquist, M. (1982), 'Nonlinear optimization by successive linear programming', *Manage. Sci.* **28**(10), 1106–1120.
URL: *http://dx.doi.org/10.1287/mnsc.28.10.1106*

Rubiales, A. J., Lotito, P. A. & Parente, L. A. (2013), 'Stabilization of the generalized benders decomposition applied to short-term hydrothermal coordination problem', *IEEE Latin America Transactions* **11**(5), 1212–1224.

Sahinidis, N. & Tawarmalani, M. (2005), 'Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints', *Journal of Global Optimization* **32**, 259–280.

Soumis, F., Pacqueau, R. & Hoang, L. N. (2012), A fast and accurate algorithm for stochastic integer programming, applied to stochastic shift scheduling.

Tawarmalani, M. & Sahinidis, N. V. (2005), 'A polyhedral branch-and-cut approach to global optimization', *Mathematical Programming* **103**, 225–249.

Van Rossum, G. & Drake, F. L. (2009), *Python 3 Reference Manual*, CreateSpace, Paramount, CA.

Wächter, A. & Biegler, L. T. (2005), 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming', *Mathematical Programming* **106**(1), 25–57.
URL: *https://doi.org/10.1007/s10107-004-0559-y*

Zaourar, S. & Malick, J. (2014), 'Quadratic stabilization of benders decomposition'.