Trabajo Fin de Máster

Calibrado y configuración de una herramienta de optimización matemática

Andrea Fernández López

Máster en Técnicas Estadísticas

Curso 2018-2019

Propuesta de Trabajo Fin de Máster

Título en galego: Calibrado e configuración dunha ferramenta de optimización matemática.

Título en español: Calibrado y configuración de una herramienta de optimización matemática.

English title: Calibration and configuration of a mathematical optimization tool.

Modalidad: Modalidad B.

Autor/a: Andrea Fernández López, Universidade de Santiago de Compostela.

Director/a: Julio González Díaz, Universidade de Santiago de Compostela; Ángel Manuel González Rueda, Universidade da Coruña.

Tutor/a: Diego Rodríguez Martínez, ITMATI (Instituto Tecnológico de Matemática Industrial)

Breve resumen del trabajo:

El TFM se encuadrará en la colaboración de ITMATI con la empresa del sector gasista REGANOSA. En el marco de esta colaboración se ha desarrollado GANESO™, un software con distintos módulos, uno de los cuales se encarga de la optimización de la red de gas en estado estacionario. Los objetivos serán la familiarización con los problemas de optimización de redes de gas y con el módulo de optimización de la red de gas en estado estacionario de GANESO™, e intentar calibrar los algoritmos de programación subyacentes para identificar las mejores configuraciones de ejecución como función de la estructura concreta de los problemas a resolver. Esta tarea se llevará a cabo tanto con instancias de prueba de los problemas como con instancias de problemas reales de las redes gallega y española.

Recomendaciones:

Haber cursado las asignaturas de Programación Lineal y Entera y Programación Matemática.

Otras observaciones:

Agradecimientos

En primer lugar me gustaría agradecer a toda mi familia el apoyo y ánimos que me aportaron durante estos dos años, en especial a mis padres, mis abuelos y a mi tía Mucha.

A mis directores de TFM, Julio González Díaz y Ángel Manuel González Rueda, por el trabajo y la ayuda que me aportaron durante este proyecto.

A todo el equipo de ITMATI, por aportar día a día su apoyo y esa pizca de humor creando un ambiente de trabajo completamente agradable.

A mis amigas Alba, Laura y Raquel, por creer en mí durante estos años, apoyarme siempre que lo necesitaba y animarme a emprender este camino.

Finalmente, a las amigas y amigos que he hecho durante esta etapa de mi vida, Áurea, Paula, Nuno, Bea, Brais, Paloma, Borja y Pablo, por compartirla conmigo y apoyarme en todo momento.

Índice general

Resumen

Resumen en español

El transporte de flujo de gas por los conductos de una red produce una pérdida de presión. El problema de esta reside en que el gas debe llegar a los puntos de consumo con una presión específica, por lo que para garantizar esto se utilizan las denominadas estaciones de compresión, que consumen una proporción del gas para aumentar la presión del mismo. De esta forma, el objetivo es minimizar el gas consumido en las mismas. Sin embargo, para modelizar este problema se deben considerar distintos factores, tanto físicos del gas como de los elementos que pertenecen a la red.

En este trabajo se presentará el modelado matemático general de una red de gas y se introducirán las restricciones necesarias para el estudio de la propagación de calidad en la misma, lo que nos garantizará que a cada punto de suministro de la red llegue la energía necesaria. Para ello introduciremos una nueva variable en el modelo: el poder calorífico. Una vez presentado este nuevo modelo, se calibrará el software GANESO™ (desarrollado en el marco de colaboración entre ITMATI y REGANOSA) para la resolución de los mismos, utilizando para ello instancias de prueba y de problemas reales tanto de la red gallega como de la red española. Una vez realizado este procedimiento, se presentarán los resultados obtenidos con las distintas configuraciones elegidas.

English abstract

When the gas is flowing through the pipes of the gas transport network, a pressure loss is produced. The issue involved in this loss is that the gas must arrive at the consumption points of the network with a specific pressure, so in order to solve this compressor stations are needed, which use some proportion of the gas to operate. Therefore, our objetive will be to minimize the gas consumption in the compressor stations. However, we need to keep in mind other factors such as physic gas properties or factors related to the elements of the network.

Throughout this project, we will introduce the mathematical modeling of a general gas network and we will add the neccessary constraints related to the gas quality propagation, which ensure that the energy that arrives at each consumption point is the correct one. In order to do so, we will introduce a new variable to our problem, the calorific value. Secondly, we will try to calibrate the software GANESO™ (created through the colaboration between ITMATI and REGANOSA) in order to solve the new models, and we will use examples and real problems' instances, from both the Galician and the Spanish network. Lastly, we will present and discuss the results obtained with the different configurations we have chosen before.

Prefacio

Este trabajo se encuadra en la colaboración de ITMATI con REGANOSA, empresa del sector gasista. Por ello, es necesario llevar a cabo una breve explicación de los elementos que componen una red de gas, así como las ecuaciones físicas necesarias para el modelado de la misma.

Se define una red de transmisión de gas como un conjunto de elementos, como estaciones de compresión o válvulas de control, conectados a través de conductos. Más adelante explicaremos cada uno de estos elementos.

Debemos tener en cuenta que cuando el gas fluye a través de un conducto de la red, este pierde presión, debido principalmente al rozamiento que se produce con las paredes del mismo. De esta forma, cuanto más largo sea el conducto, más presión pierde. El problema de esta pérdida reside en que el gas debe llegar a los puntos de suministro con una presión específica, determinada por una cota inferior y otra superior. Este problema se puede resolver mediante compresores, que aumentan la presión del gas para que este pueda seguir fluyendo por la red y llegar así con la presión necesaria. Sin embargo, para llevar a cabo este proceso el compresor consume una proporción del gas.

Surge así la necesidad de modelizar dicho problema como un problema de programación matemática, ya que tratamos de minimizar el consumo de gas en los compresores teniendo en cuenta las cotas de presión de los puntos de consumo y suministro. Además, también nos puede interesar optimizar otras cantidades, como el vapor de gas generado en las plantas de regasificación (también conocido como *boil-off*), el *linepack*... de las que hablaremos posteriormente en el Capítulo 1.

El objetivo de este trabajo será añadir a dicho problema las restricciones necesarias para la propagación de calidad del gas y ajustar la herramienta GANESO™, software desarrollado en la colaboración entre ITMATI y REGANOSA, para resolver esta nueva versión del problema, procedimientos expuestos a lo largo del Capítulo 3.

Por otra parte, un gas puede ser modelado como un fluido Newtoniano viscoso por lo que en el modelado hay que tener en cuenta la ecuación de estado para gases reales y la ley de Fourier para el flujo de calor. Además, se pueden deducir tres ecuaciones de conservación a partir de las ecuaciones de Navier-Stokes para flujos compresibles, para las que consideramos un modelo unidimensional. Se denotará como x ($[m]$) la variable espacial y como t ($[s]$) la variable temporal.

- Ecuación de estado para gases reales:

$$p(x,t) = Z(p(x,t),\theta(x,t))\rho(x,t)R\theta(x,t),$$

  donde $Z = Z(p(x,t),\theta(x,t))$ es el factor de compresibilidad, $p = p(x,t)$ ($[Pa]$) la presión, $\rho = \rho(x,t)$ ($[kg/m^3]$) la densidad, $\theta = \theta(x,t)$ ($[K]$) la temperatura absoluta y R la constante del gas, siendo esta $R = \dfrac{\mathcal{R}}{M}$ donde $\mathcal{R} = 8.31434$ ($[kJ/(kmol \cdot K)]$) es la constante universal de los gases y M ($[kg/kmol]$) la masa molar del gas.

- Ley de Fourier para el flujo de calor: establece que el flujo de transferencia de calor por conducción es proporcional y de sentido contrario al gradiente de temperatura en esa dirección.

$$q(x,t) = -\kappa \frac{\partial \theta(x,t)}{\partial x},$$

siendo κ ($[W/(m \cdot K)]$) el coeficiente de conductividad termal.

- Conservación de masa: Establece que la variación de caudal del gas se compensa con la variación de densidad del mismo.

$$A(x)\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0,$$

siendo $A = A(x)$ ($[m^2]$) el área de la sección del conducto por el que circula el gas y $q = q(x,t)$ ($[kg/s]$) el caudal en masa del gas.

- Conservación del momento: Define todas las fuerzas que actúan sobre las moléculas del gas.

$$A(x)\frac{\partial p(x,t)}{\partial x} + \frac{\partial q(x,t)}{\partial t} + \frac{\lambda(q(x,t))}{2DA(x)\rho(x,t)}|q(x,t)|q(x,t) + A(x)g\rho(x,t)\frac{\partial h(x)}{\partial x} = 0,$$

donde $\lambda = \lambda(q(x,t))$ es el valor de la fricción que se produce a través del conducto, D ($[m]$) el diámetro del conducto, g ($[m/s^2]$) la aceleración de la gravedad y $h = h(x)$ ($[m]$) la altura a la que se encuentra el conducto.

De esta forma el primer término nos indica la variación de presión en función del área de la sección del conducto, el segundo se refiere a la variación de caudal con respecto al tiempo, el tercer término nos indica como interviene la fricción sobre el conducto en el caudal del gas y el último representa la fuerza gravitacional existente en conductos que se encuentren a una cierta altura.

- Conservación de energía: Establece la relación entre la energía interna del gas y el intercambio de calor con el medio.

$$W(t) \cdot (\rho(x,t)A(x)dx) = \frac{\partial}{\partial t}\left[(\rho(x,t)A(x)dx) \cdot \left(c_v\theta(x,t) + \frac{v(x,t)^2}{2} + g \cdot dh(x)\right)\right] + \frac{\partial}{\partial x}\left[(\rho(x,t)v(x,t)A(x)dx) \cdot \left(c_v\theta(x,t) + \frac{p(x,t)}{\rho(x,t)} + \frac{v(x,t)^2}{2} + g \cdot dh(x)\right)\right],$$

siendo $W = W(t)$ ($[kJ/(kg \cdot s)]$) la aportación de calor al gas, c_v ($[kJ/(kg \cdot J)]$) el calor específico del gas con volumen constante y $v = v(x,t)$ ($[m/s]$) la velocidad del gas.

En nuestro modelo se considerará que la temperatura del gas es un parámetro constante, lo que implica que la ecuación de conservación de energía y la Ley de Fourier para el flujo del calor no son necesarias para modelar el problema. Por otra parte, se adaptan las ecuaciones restantes para el caso de gases en estado estacionario, en cuyo caso, el caudal de gas que circula por los conductos de la red es independiente del tiempo, por lo que las derivadas con respecto al tiempo de las ecuaciones anteriores desaparecen. Además, se considera que el área de la sección, $A(x)$, es la misma a lo largo de todo el conducto, por lo que $A(x) = A \; \forall x \in [0, L]$, siendo L la longitud del conducto.

Aplicando estos cambios nos queda el siguiente sistema de ecuaciones:

- Conservación de masa:

$$\frac{\partial q(x)}{\partial x} = 0$$

Esto implica que el caudal en masa permanece constante a través del conducto, es decir, $q(x) = q$, $\forall x \in [0, L]$, siendo L ($[m]$) la longitud del conducto.

- Conservación del momento:

$$A(x)\frac{\partial p(x)}{\partial x} + \frac{\lambda(q(x))}{2DA(x)\rho(x)}|q(x)|q(x) + A(x)g\rho(x)\frac{\partial h(x)}{\partial x} = 0$$

- Ecuación de estado para gases reales:

$$p(x) = Z(p(x), \theta(x))\rho(x)R\theta(x)$$

Por último, se aproxima el modelo unidimensional integrando la ecuación de conservación del momento entre los extremos del conducto, $x = 0$ y $x = L$. Resolviendo el sistema, procedimiento que se puede consultar en González Rueda (2017), se llega a la ecuación de pérdida de presión:

$$u(0) - u(L) = \frac{16\lambda(q)L}{\pi^2 D^5} R\theta_m |q| q Z(\sqrt{u_m}, \theta_m) + \frac{2g}{R\theta_m}\frac{u_m}{Z(\sqrt{u_m}, \theta_m)}(h(L) - h(0))$$

Denotando por $u(x) = p(x)^2$ la presión al cuadrado del gas en el punto x, vemos que esta ecuación en realidad es la diferencia de los cuadrados de las presiones. Como comentamos anteriormente, la pérdida de presión se produce, principalmente, por el rozamiento producido con las paredes del conducto cuando el gas circula por él. Esto se ve representado en la ecuación anterior, pues vemos que a mayor coeficiente de fricción λ, mayor es la pérdida de presión. Depende también del caudal que circule por el conducto, q, así como de las características del conducto, pues vemos que es directamente proporcional a la longitud del mismo, L, e inversamente proporcional a D^5, siendo D el diámetro. También depende de la diferencia de altura entre los extremos del conducto, $h(L) - h(0)$, y de las propiedades del gas, como la temperatura media, θ_m, y la constante del gas, R.

Una vez vistas las características físicas más importantes para nuestro modelo, así como las ecuaciones deducidas de las mismas, se puede pasar al modelado matemático de la red, explicado a lo largo del Capítulo 1.

Capítulo 1

Modelado matemático de una red de transmisión de gas.

Como se explicó anteriormente, a medida que el gas fluye a través de la red, se pierde presión debido principalmente a la fricción que se produce con las paredes del conducto. Esta pérdida de presión puede llegar a suponer un gran problema, ya que el gas debe llegar a los puntos de la red con una cierta presión comprendida entre unas determinadas cotas. Para evitar este problema se activan las estaciones de compresión, pero esto supone un consumo de gas. Por otra parte, debemos tener en cuenta que los conductos de la red tienen una cierta capacidad, por lo que la cantidad de gas que circule por ellos también está acotada.

Teniendo en cuenta los diferentes problemas que se presentan, surge la necesidad de modelar un problema de programación matemática para intentar optimizar la transmisión del gas a través de la red.

Este capítulo está basado principalmente en la tesis de González Rueda (2017), por lo que se puede consultar la misma para más detalle. En él se explica el modelado matemático del problema, dividido en tres partes. En la primera se presenta una representación gráfica de la red, así como las variables a tener en cuenta. La segunda trata sobre las ecuaciones matemáticas que definen las restricciones. Por último, se presentan los términos que forman parte de la función objetivo del problema. Además, en la Sección 2.1 se presentará un algoritmo que usaremos para comprobar el funcionamiento de la herramienta.

1.1. Representación gráfica y variables de la red.

Se representará la red de transmisión del gas como un grafo dirigido $\boldsymbol{G} = (N, E)$ donde N y E representan el conjunto de nodos y aristas del grafo, respectivamente. De esta forma, cada elemento de E es un par ordenado de N, por ejemplo, si $(1, 4) \in E$, tenemos que existe una arista en el grafo $\boldsymbol{G}$ que une los nodos 1 y 4 siendo 1 el nodo inicial y 4 el nodo final de la arista.

Antes de continuar con la representación de la red de transmisión de gas es importante definir los conceptos de subgrafo, camino y ciclo, ya que serán de utilidad más adelante. Para profundizar más en estos conceptos se puede consultar González Díaz et al. (2014). Diremos que $\boldsymbol{G'} = (N', E')$ es un subgrafo de $\boldsymbol{G}$ si los conjuntos de nodos y aristas de $\boldsymbol{G'}$, N' y E', son subconjuntos de N y E, respectivamente. Es decir, $N' \subseteq N$ y $E' \subseteq E$. Por otra parte, llamaremos camino a una concatenación de aristas, de forma que el nodo final de una arista coincida con el nodo inicial de la siguiente. Por último, diremos que un grafo $\boldsymbol{G}$ contiene un ciclo si existe algún camino cerrado (los nodos inicial y final coinciden), donde no se repiten ni nodos ni aristas, excepto los nodos inicial y final que coinciden.

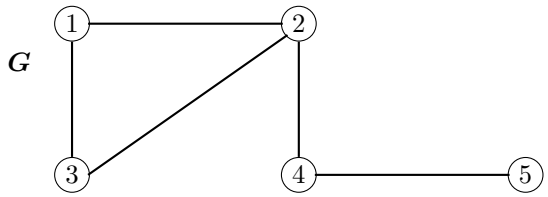


Figura 1.1: Representación de un grafo no dirigido.

En la Figura 1.1 tenemos un grafo $\boldsymbol{G} = (N, E)$ no dirigido, es decir, dada una arista $k = (i, j) \in E$ podemos ir del nodo i al j o del j al i indistintamente. En el caso de los grafos dirigidos, la dirección de las aristas viene indicada por una flecha. Si consideramos por ejemplo, $\boldsymbol{G'} = (N', E')$, formado por el conjunto de nodos $N' = 1, 2, 4$ y el conjunto de aristas $E' = (1, 2), (2, 4)$, se tiene que $\boldsymbol{G'}$ es un subgrafo de $\boldsymbol{G}$. Por otra parte, podemos definir varios caminos en $\mathbf{G}$, por ejemplo el camino $(1, 2), (2, 4), (4, 5)$ o $(1, 3), (3, 2), (2, 4)$. Además, claramente se ve que el grafo posee un ciclo, formado por el camino $(1, 2), (2, 3), (3, 1)$.

Dado un grafo $\boldsymbol{G} = (N, E)$, este puede ser representado mediante la matriz de adyacencia $A \in \mathcal{M}_{n\times n}$ con $n = |N|$ de forma que:

$$a_{ij} = \begin{cases} 1 & \text{si } k = (i, j) \in E \\ 0 & \text{en otro caso.} \end{cases}$$

Otra posible representación del grafo $\boldsymbol{G}$ es mediante la matriz de incidencia $B \in \mathcal{M}_{n\times e}$ con $n = |N|$ y $e = |E|$ de forma que:

$$b_{ik} = \begin{cases} 1 & \text{si } i \text{ es el nodo incial de la arista } k \\ -1 & \text{si } i \text{ es el nodo final de la arista } k \\ 0 & \text{en otro caso.} \end{cases}$$

En nuestro caso, se representará cada conducto de la red como una arista entre dos nodos. Sin embargo, estos nodos pueden representar distintos elementos de la red:

- Puntos de entrada o suministro de gas: son aquellos nodos por los que entra el gas a la red. Principalmente representan plantas de regasificación, donde se convierte el gas natural licuado en gas natural para introducirlo a la red, y conexiones internacionales por las que se importa el gas.
- Puntos de salida o consumo de gas: son aquellos nodos en los que se demanda una cierta cantidad de gas a una determinada presión. También puede representar las conexiones internacionales por las que se exporta el gas.
- Nodos estructurales: son aquellos nodos que no tienen flujos externos, es decir, ni entra ni sale gas de la red. Representan la unión entre conductos que tienen diferentes características técnicas, como el diámetro o el coeficiente de fricción.
- Almacenes subterráneos: pueden ser utilizados para introducir gas a la red o como puntos de consumo donde el fin es almacenar el gas. Sin embargo, estos van a ser irrelevantes, ya que los podemos incluir en alguno de los casos anteriores, dependiendo de la función que lleve a cabo el nodo en el escenario en estudio.

En el caso de las aristas, también pueden representar diferentes elementos de la red:

- Compresor: Es un elemento de la red que en el caso de estar activado, permite aumentar la presión del gas en la dirección en la que fluya el mismo. Los compresores no son elementos independientes de la red, pues forman parte de lo que denominamos estaciones de compresión.
- Válvulas de control: Al igual que los compresores, son elementos de la red que pueden estar activados o no. Existen diferentes tipos de válvulas de control:
  - Válvulas de control de presión (PCV): permiten reducir la presión del gas en la dirección en la que fluye.
  - Válvulas de apertura y cierre (OCV): permiten controlar el flujo de gas a través de una válvula.
  - Válvulas de regulación de presión fija (FRV): permiten reducir la presión del flujo que llega a la válvula a una presión fija.

Para representar gráficamente los compresores utilizaremos una arista con un triángulo que señala la dirección en la que se realice la compresión, es decir, aumenta la presión del gas, ◀ ó ▶. En el caso de las válvulas de control, se representarán con una arista con el siguiente símbolo: $\bigotimes$.

Una vez que están definidos los distintos elementos que podemos encontrar representados en el grafo por un nodo o una arista, es importante definir cuáles son las variables del problema:

- Presión de los nodos ($[Pa]$): denotada por p_i, $i \in N$. Estas variables son no negativas, $p_i \geq 0$, $\forall i \in N$. Se considerarán las presiones al cuadrado, $u_i = p_i^2$ ($[Pa^2]$) $\forall i \in N$, pues de esta forma, tal y como se verá en la Sección 1.2, se soluciona alguna no linealidad en la restricción de la pérdida de presión.
- Caudal másico que circula por las aristas ($[kg/s]$): denotado por q_k, $k \in E$. Sea $k = (i, j) \in E$ el arco que va del nodo i al nodo j, tendremos que q_k es positivo si el flujo circula en la dirección del arco (del nodo i al nodo j), y negativo si circula en dirección contraria (del nodo j al nodo i).

Asociadas a estas variables tendremos unas cotas superior e inferior, de forma que la presión con la que el gas llega a cada nodo y el cuadal másico que circula por cada arista deben estar comprendidos entre estas cotas:

$$\underline{u}_i \leq u_i \leq \overline{u}_i, \ \forall i \in N \tag{1.1}$$

$$\underline{q}_k \leq q_k \leq \overline{q}_k, \ \forall k \in E \tag{1.2}$$

En la Sección 1.2, se puede ver que aparecen nuevas variables auxiliares surgidas a la hora de modelar algunos elementos del problema. Además, a lo largo del Capítulo 3 se presenta una nueva variable que introduciremos en el modelo, el poder calorífico.

1.2. Restricciones del problema.

Como se explicó anteriormente, de las características físicas de un gas en estado estable y suponiendo que la temperatura del gas es constante llegamos a un sistema de ecuaciones formado por la conservación del flujo másico en cada uno de los nodos y la pérdida de presión que se produce cuando el gas circula a través de las aristas. Este sistema, junto las ecuaciones necesarias para modelar el comportamiento de los compresores y de las distintas válvulas de presión forman el conjunto de restricciones de nuestro problema. A continuación, se ve más detalladamente cada una de ellas.

1.2.1. Conservación de flujo.

La ecuación de conservación de flujo establece que el flujo total que llega a un nodo tiene que ser igual al flujo total que sale del mismo. Denotando por $\underline{c}_i$ y por $\overline{c}_i$ las cotas inferior y superior del intercambio de flujo con el exterior en el nodo i, respectivamente, se tiene la siguiente restricción:

$$\underline{c}_i \leq \sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k \leq \overline{c}_i, \ \forall i \in N, \tag{1.3}$$

donde E_i^{ini} representa el conjunto de aristas que tienen como nodo inicial el nodo i y E_i^{fin} representa el conjunto de aristas que tienen como nodo final el nodo i.

A continuación, vamos a ver como quedaría la restricción según el tipo de nodo sobre el que estemos. Denotando por N^s el conjunto de nodos suministro, por N^c el de nodos consumo y por N^{st} el de nodos estructurales, tenemos lo siguiente:

- Si $i \in N^s$, entonces se tiene que $\underline{c}_i = 0$ y $\overline{c}_i = c_i$, siendo c_i la capacidad del nodo i.

$$0 \leq \sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k \leq c_i$$

- Si $i \in N^c$, entonces se tiene que $\underline{c}_i = \overline{c}_i = d_i$, siendo d_i la demanda existente en el nodo i.

$$d_i \leq \sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k \leq d_i \Longleftrightarrow \sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k = d_i$$

- Si $i \in N^{st}$, entonces se tiene que $\underline{c}_i = \overline{c}_i = 0$, ya que no hay ningún intercambio de flujo con el exterior.

$$\sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k = 0$$

Se debe tener en cuenta que los nodos correspondientes al borde de una arista que represente una estación de compresión se considerarán nodos estructurales. Aunque se pierde una cierta cantidad de flujo cuando la estación de compresión está activada, esta es suficientemente pequeña, por lo que se considera despreciable.

1.2.2. Pérdida de presión.

La ecuación de pérdida de presión establece que cuando el gas circula a través de un conducto, pierde presión debido principalmente a la fricción con las paredes del mismo. Denotando por E^{st} el conjunto de aristas estructurales y dada $k = (i, j) \in E^{st}$, se tiene la siguiente restricción:

$$u_i - u_j = \frac{16\lambda(q_k)L_k}{\pi^2 D_k^5} R\theta_{m_k} |q_k| q_k Z(\sqrt{u_{m_k}}, \theta_{m_k}) + \frac{2g}{R\theta_{m_k}} \frac{u_{m_k}}{Z(\sqrt{u_{m_k}}, \theta_{m_k})}(h_j - h_i), \tag{1.4}$$

donde $\lambda(q_k)$ es el coeficiente de fricción, R ($[J/(kg \cdot K)]$) es la constante del gas, g ($[m/s^2]$) es la aceleración de la gravedad, Z el factor de compresibilidad, L_k ($[m]$) y D_k ($[m]$) representan la longitud y el diámetro del conducto k respectivamente, h_i y h_j son las alturas ($[m]$) a las que se encuentra el conducto en los nodos i (nodo inicial) y j (nodo final) respectivamente, u_{m_k} ($[Pa^2]$) la media de las presiones al cuadrado en el conducto k y θ_{m_k} ($[K]$) la temperatura media del gas en dicho conducto. Dado que consideramos que la temperatura del gas permanece constante, también lo es θ_{m_k}.

Como podemos observar, el coeficiente de fricción $\lambda(q_k)$ depende del caudal que circule por la arista k, variable en nuestro modelo. En algunos modelos se calculará mediante la fórmula de Weymouth, lineal en q_k, mientras que en otros se calculará mediante la fórmula de Colebrook, siendo esta una fórmula más precisa. En cualquier caso, una descripción detallada del cálculo de estas dos funciones va más allá de esta memoria.

Hay que tener en cuenta que el resto de aristas representan estaciones de compresión o válvulas de control y ambos sirven para modificar la presión del gas. Es por esto, que la ecuación de pérdida de presión solo se define en las aristas estructurales.

1.2.3. Estaciones de compresión.

Como ya mencionamos anteriormente, la principal funcionalidad de un compresor es aumentar la presión del gas en la dirección del flujo para así contrarrestar la pérdida que se produce durante el avance del gas a través de la red. Por lo tanto, si denotamos por E^c el conjunto de aristas que representan compresores y sea $k = (i, j) \in E^c$, suponiendo que el flujo circula en el sentido de la arista, $q_k \geq 0$, tenemos que la presión al inicio del compresor tiene que ser menor o igual a la presión del nodo final, es decir,

$$u_i \leq u_j \tag{1.5}$$

Sin embargo, los compresores no figuran en la red como elementos independientes, sino que aparecen en lo que se denominan estaciones de compresión, formadas por un grupo de compresores y válvulas. Además, cada compresor puede estar activado o desactivado y deben funcionar acorde a los diagramas de operación relacionados con las características técnicas de la estación. Por estas razones, es necesario profundizar un poco más en este tema y definir algunas restricciones adicionales.

Para empezar, para poder representar todas las posibles formas en las que funcione una estación de compresión, es necesario modelar compresores bidireccionales, es decir, que son capaces de comprimir el gas en ambas direcciones. El principal problema de los compresores bidireccionales es que a priori desconocemos el sentido en el que circula el flujo, por lo que se introducirá una variable binaria auxiliar que explique la dirección del flujo. Denotando por E^{bc} el conjunto de aristas que representen un compresor bidireccional y sea $k = (i, j) \in E^{bc}$, se define la variable binaria $y_k^{bc} \in \{0, 1\}$ de forma que:

$$q_k \leq \overline{q}_k y_k^{bc}$$

$$q_k \geq \underline{q}_k (1 - y_k^{bc})$$

Si $y_k^{bc} = 1$ se verifica que el flujo va en la dirección de i a j pues $0 \leq q_k \leq \overline{q}_k$, es decir, $q_k \geq 0$. Si $y_k^{bc} = 0$ nos indica que el flujo va en dirección contraria, de j a i, pues en este caso tenemos que $\underline{q}_k \leq q_k \leq 0$, es decir, $q_k \leq 0$.

Por otra parte, la presión del gas que fluye por un compresor aumenta en la dirección del flujo, por lo que obtenemos las siguientes restricciones:

$$u_i - u_j \leq M_1 (1 - y_k^{bc})$$

$$u_j - u_i \leq M_2 y_k^{bc}$$

siendo M_1 y M_2 constantes positivas suficientemente grandes.

Sin embargo, los compresores bidireccionales pueden complicar bastante el modelo, ya que para cada compresor necesitaríamos una variable binaria auxiliar, por lo tanto se duplica cada arista

$k = (i,j) \in E^{bc}$ introduciendo un nodo estructural l entre la arista k original y la duplicada, de forma que cada una tendrá un compresor en una única dirección. Es decir, en vez de tener una arista, $k = (i,j)$, con un compresor bidireccional se tienen dos aristas conectadas mediante un nodo estructural, $k_1 = (i,l)$ y $k_2 = (l,j)$ con un compresor en cada una, como se puede ver en la Figura 1.2.

Figura 1.2: Duplicación de aristas en un compresor bidireccional.

De esta forma, si $k_1 = (i,l)$, $k_2 = (l,j) \in E^c$, se verifican las siguientes ecuaciones asegurando que cada compresor solo aumenta la presión en una dirección:

$$u_i \leq u_l$$

$$u_j \leq u_l$$

Esta proceso de duplicación de aristas es útil para saber en que dirección funciona el compresor bidireccional original. Sea $k = (i,j) \in E^{bc}$ y sean $k_1 = (i,l)$, $k_2 = (l,j) \in E^c$, si k_1 está activado, entonces sabemos que el aumento de presión se produce de i a j y $q_k \geq 0$. En el caso de que sea k_2 el que está activado, el aumento de presión se produce de j a i y $q_k \leq 0$.

Por otro lado, debido al modelado de las estaciones de compresión, es necesario asegurar que como máximo en cada estación sólo haya un compresor activado. Para ello denotamos por CS el conjunto de estaciones de compresión del modelo. Sea $cs \in CS$, denotaremos por E^{cs} el conjunto de compresores que están en la estación cs.

Dado $k = (i,j) \in E^c$, definimos la variable binaria $y_k^{oc} \in \{0,1\}$, que determina si el compresor k está o no activado, obteniendo como resultado las siguientes restricciones:

$$u_i - u_j \leq M_1(1 - y_k^{oc}) \tag{1.6}$$

$$u_i - u_j \geq -M_2 y_k^{oc} \tag{1.7}$$

siendo M_1 y M_2 constantes positivas suficientemente grandes.

Una vez definidos los compresores que están activados, para asegurar que en cada estación de compresión funcione a lo sumo un único compresor, se tiene la siguiente restricción:

$$\sum_{k \in E^{cs}} y_k^{oc} \leq 1 \tag{1.8}$$

Por último, cada compresor tiene asociado un diagrama operativo, denotado por D, que impone varias restricciones a su funcionamiento. Dado que las pruebas que se realicen en el marco de este trabajo no incluyen rangos de operaciones, no tendremos en cuenta los diagramas operativos de los compresores. Por lo tanto, presentaremos brevemente las restricciones relativas a los mismos sin entrar en gran detalle.

Considerando una estación de compresión con n compresores idénticos en paralelo, el diagrama operativo viene dado por $D = \bigcup_{i=1}^{n} D^r$, siendo $D^r = \{(p_s, q, p_d)/(p_s, \frac{q}{r}, p_d) \in D^1\}$ donde p_s es la presión de succión, q la cantidad de flujo que circula a través del compresor y p_d la presión de descarga.

Dado $k = (i,j) \in E^c$, se tiene que la presión de succión es p_i, la presión de descarga p_j y el flujo que circula a través del compresor se corresponde con q_k. Denotando por $D^{N_k^{ac}}$ el diagrama operativo si hay exactamente N_k^{ac} $(1 \leq N_k^{ac} \leq n)$ compresores funcionando, y teniendo en cuenta que dicho

diagrama solo es válido para aquellos compresores que estén funcionando, las restricciones asociadas a $D^{N_k^{ac}}$ son las siguientes:

$$\underline{u}_i \leq u_i \leq \overline{u}_i \tag{1.1}$$

$$y_k^{oc} S_{min} su \frac{\sqrt{u_i}}{Z(\sqrt{u_i}, \theta_i) R \theta_i} \leq \frac{q_k}{N_k^{ac}} \leq y_k^{oc} S_{max} st \frac{\sqrt{u_i}}{Z(\sqrt{u_i}, \theta_i) R \theta_i} \tag{1.9}$$

$$y_k^{oc} \left(g_L \left(\sqrt{u_i}, \frac{q_k}{N_k^{ac}} \right) \right)^2 - M(1 - y_k^{oc}) \leq \frac{u_j}{u_i} \leq y_k^{oc} \left(g_U \left(\sqrt{u_i}, \frac{q_k}{N_k^{ac}} \right) \right)^2 + M(1 - y_k^{oc}) \tag{1.10}$$

donde y_k^{oc} es la variable binaria que determina si un compresor está activo, S_{min} la línea de mínima velocidad del compresor, S_{max} la línea de máxima velocidad, su el límite de aumento y st la curva que define el flujo cuando la velocidad del gas alcanza la velocidad del sonido. Por último, g_L y g_U son funciones relacionadas con las curvas inferior y superior, respectivamente, del diagrama operativo.

1.2.4. Válvulas.

Como ya definimos anteriormente, las válvulas son elementos que nos permiten controlar el flujo de gas y disminuir su presión. A continuación se ven las restricciones necesarias en el modelo según el tipo de válvula.

Las válvulas de control de presión (PCV) permiten reducir la presión del gas en la dirección en la que circule el flujo. Denotando por E^{pcv} el conjunto de aristas que representen este tipo de válvulas y dado $k = (i, j) \in E^{pcv}$ se define una variable binaria $y_k^{pcv} \in \{0, 1\}$ que nos indique el sentido en el que circula el flujo, dando lugar a las siguientes restricciones:

$$q_k \leq \overline{q}_k y_k^{pcv} \tag{1.11}$$

$$q_k \geq \underline{q}_k (1 - y_k^{pcv}) \tag{1.12}$$

De esta forma, si $y_k^{pcv} = 0$ se tiene $\underline{q}_k \leq q_k \leq 0$, es decir, el flujo circula en sentido contrario por la arista k, va del nodo j al nodo i. En el caso en que $y_k^{pcv} = 1$ se tiene $0 \leq q_k \leq \overline{q}_k$, y como $q_k \geq 0$ el flujo circula en el sentido de la arista k, del nodo i al nodo j.

Una vez definida la dirección en la que circula el flujo mediante la variable binaria y_k^{pcv}, es necesario definir las siguientes restricciones para que la reducción de presión se realice correctamente, pues como se explicó anteriormente, esta reducción se debe realizar en el sentido del flujo:

$$u_i - u_j \leq M_1 y_k^{pcv} \tag{1.13}$$

$$u_i - u_j \geq M_2 (1 - y_k^{pcv}) \tag{1.14}$$

siendo M_1 y M_2 suficientemente grandes.

De esta forma se tiene, que si el flujo circula de i a j, es decir, $y_k^{pcv} = 1$, entonces $0 \leq u_i - u_j \leq M_1$ y por lo tanto $u_j \leq u_i$, la presión en el nodo j es menor que la presión en el nodo i. Por otro lado, si el flujo circula en sentido contrario, $y_k^{pcv} = 0$, entonces $M_2 \leq u_i - u_j \leq 0$ y por lo tanto $u_i \leq u_j$, la presión en el nodo i es menor que la presión en el nodo j.

Las válvulas de apertura y cierre (OCV) permiten controlar el flujo de gas a través de una válvula, de forma que si está cerrada, el gas no podrá fluir a través de la misma, y la presión a ambos lados de la válvula estará desacoplada. Por otra parte, si la válvula está abierta, la presión a ambos lados deberá ser la misma.

Denotemos por E^{ocv} el conjunto de aristas que representan una válvula de apertura y cierre, sea $k = (i, j) \in E^{ocv}$, se define una variable binaria $y_k^{ocv} \in \{0, 1\}$ que describe si la válvula está abierta ($y_k^{ocv} = 1$) o cerrada ($y_k^{ocv} = 0$), obteniendo las siguientes ecuaciones:

$$q_k \leq \overline{q}_k y_k^{ocv} \tag{1.15}$$

$$q_k \geq \underline{q}_k y_k^{ocv} \tag{1.16}$$

En el caso de que la válvula esté cerrada, $y_k^{ocv} = 0$, tenemos $0 \leq q_k \leq 0$, es decir, $q_k = 0$ y por lo tanto el gas no fluye por la arista k. Por otra parte, si la válvula está abierta, $y_k^{ocv} = 1$, tenemos que $\underline{q}_k \leq q_k \leq \overline{q}_k$.

En este caso, las restricciones relacionadas con el comportamiento de la presión en este tipo de válvulas son las siguientes:

$$u_i - u_j \leq M_1(1 - y_k^{ocv}) \tag{1.17}$$

$$u_i - u_j \geq M_2(y_k^{ocv} - 1) \tag{1.18}$$

siendo M_1 y M_2 constantes positivas suficientemente grandes.

Se puede ver que si la válvula está abierta ($y_k^{ocv} = 1$), se tiene $0 \leq u_i - u_j \leq 0$ y por lo tanto la presión a ambos lados de la válvula será la misma pues $u_i = u_j$. Por otro lado, si la válvula está cerrada ($y_k^{ocv} = 0$) se tiene $u_i - u_j \leq M_1$ y $u_i - u_j \leq M_2$, por lo que se ve que las presiones en los nodos i y j están desacopladas, no hay una relación entre ellas.

Por último, se va a realizar el mismo estudio para las válvulas de regulación de presión fija (FRV), que permiten reducir la presión del gas que fluye por la válvula a una determinada presión fija. Estas válvulas son útiles entre conductos que tengan una cota superior de presión distinta, pues de esta forma, pasando por la válvula, se puede reducir la presión del gas hasta un nivel adecuado para seguir fluyendo por el siguiente conducto.

Denotemos por E^{frv} el conjunto de aristas que representan una válvula de regulación de presión fija, y sea $k = (i, j) \in E^{frv}$, si denotamos por p_k^{frv} la presión fija a la que debe regularse la válvula, necesitamos que se verifique lo siguiente:

- Si $p_i > p_k^{frv}$ entonces $p_j = p_k^{frv}$, es decir, si la presión a la que llega el gas al nodo i es mayor a la presión fija de la válvula, entonces esta reduce la presión del gas al nivel fijado.
- Si $p_i \leq p_k^{frv}$ entonces $p_j = p_i$, es decir, en el caso en que el gas llegue al nodo i con una presión inferior o igual a la establecida en la válvula, no es necesario reducirla, y por lo tanto llega al nodo j con la misma presión.

Se define una variable binaria $y_k^{frv} \in \{0, 1\}$ que nos indique si la válvula está activada o no. En el caso de que la válvula esté activada, $y_k^{frv} = 1$, se tiene que $u_i \geq u_k^{frv}$ por lo que se definen las siguientes restricciones:

$$u_i - u_k^{frv} \leq M_1 y_k^{frv} \tag{1.19}$$

$$u_i - u_k^{frv} \geq M_2(y_k^{frv} - 1) \tag{1.20}$$

siendo M_1 y M_2 constantes positivas suficientemente grandes. Como se puede comprobar, si $y_k^{frv} = 1$ se tiene que $0 \leq u_i - u_k^{frv} \leq M_1$ de donde obtenemos que $u_i \geq u_k^{frv}$.

Por otro lado, si la válvula está activada, la presión en el nodo j debe ser igual a la presión fija de la válvula, $u_j = u_k^{frv}$, por lo que definimos las siguientes restricciones:

$$u_j - u_k^{frv} \leq M_1(1 - y_k^{frv}) \tag{1.21}$$

$$u_j - u_k^{frv} \geq M_2(y_k^{frv} - 1) \tag{1.22}$$

siendo M_1 y M_2 constantes positivas suficientemente grandes. Como se puede comprobar, si $y_k^{frv} = 1$ se tiene que $0 \leq u_j - u_k^{frv} \leq 0$ de donde obtenemos que $u_j = u_k^{frv}$.

Por último, en el caso en que la válvula esté desactivada, $y_k^{frv} = 0$, la presión del gas en los dos extremos del conducto debe ser la misma, $u_i = u_j$, por lo que definimos las siguientes restricciones:

$$u_i - u_j \leq M_1 y_k^{frv} \tag{1.23}$$

$$u_i - u_j \geq -M_2 y_k^{frv} \tag{1.24}$$

donde M_1 y M_2 son constantes positivas suficientemente grandes. De esta forma vemos que si $y_k^{frv} = 0$ se tiene que $0 \leq u_i - u_j \leq 0$, es decir, $u_i = u_j$.

1.3. Función objetivo del problema.

Una vez definidas las principales restricciones del problema, podemos definir la función objetivo del mismo. Como ya mencionamos anteriormente, nos interesa principalmente minimizar el consumo de gas que se produce en las estaciones de compresión cuando un compresor está activo. Sin embargo, también nos puede interesar optimizar el vapor de gas que se genera en las plantas de regasificación (denominado también como *boil-off*), o el *linepack*, es decir, el almacenamiento de gas en los propios conductos de la red, entre otras. A continuación veremos más detalladamente cada una de estas componentes.

1.3.1. Consumo de gas en las estaciones de compresión.

Las estaciones de compresión necesitan una fuente de energía para funcionar, utilizando generalmente parte del gas de la red. Por lo tanto, es importante, como ya mencionamos en otras ocasiones, minimizar este consumo. Denotando por g_k ($[kg/s]$) el consumo de gas que se produce en el compresor $k = (i, j) \in E^c$ representando N compresores, tendríamos:

$$g_k = \frac{1}{LCV}\frac{1}{\varepsilon\xi\eta_k^*}\frac{\gamma}{\gamma - 1}Z(\sqrt{u_i}, \theta_i)R\theta_i\left(\left(\frac{u_j}{u_i}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right)|q_k| \tag{1.25}$$

donde LCV ($[J/kg]$) es la cota inferior de valor calorífico, η_k^* la eficiencia isentrópica óptima del compresor, ε la eficiencia mecánica, ξ la eficiencia de la turbina de gas que impulsa el compresor y γ el ratio de calores específicos.

Con respecto a la eficiencia isentrópica óptima del compresor, esta debe verificar:

$$\eta_k^* = \max_{r \in \mathcal{R}} \eta\left(\frac{q_k}{r}, p_i, p_j\right)$$

siendo $\mathcal{R}$ el conjunto factible que determina la cantidad de compresores que pueden funcionar dado $(q, p_i, p_j) \in D$, es decir,

$$\mathcal{R} = \{r / r \in \mathbb{Z}, 1 \leq r \leq N, (q, p_i, p_j) \in D^r\}$$

Además, si tenemos en cuenta que la variable N_k^{ac} indica el número de compresores que están funcionando ($1 \leq N_k^{ac} \leq N$), llegamos a la siguiente relación:

$$\eta_k^* = \eta\left(\frac{q_k}{N_k^{ac}}, p_i, p_j\right) = \max_{r\in\mathcal{R}} \eta\left(\frac{q_k}{r}, p_i, p_j\right) \tag{1.26}$$

Volviendo a la ecuación de pérdida de gas (1.25), vemos que el gas consumido en la estación de compresión depende principalmente del ratio de compresión $\frac{u_j}{u_i}$ y de la cantidad de flujo que entra al compresor, $|q_k|$.

Además, es fácil ver que si el compresor k está desactivado no se produce ningún consumo de gas, ya que la presión permanecería constante entre los nodos i y j y por lo tanto $\frac{u_j}{u_i} = 1$.

Por último es importante destacar la necesidad del valor absoluto en la cantidad de flujo que circula por la arista k, $|q_k|$, ya que como comentamos en la Sección 1.2, nos encontramos situaciones en las que el el compresor $k = (i, j)$ funciona como una válvula de control de presión, en cuyo caso el flujo es negativo $q_k \leq 0$ pero el ratio $\frac{u_j}{u_i}$ es mayor o igual a 1 debido a la ecuación (1.5). De esta forma las soluciones con compresores funcionando como válvulas PCV serían preferibles, ya que tendrían asociado un coste negativo. Una vez explicada la necesidad del valor absoluto y considerando que el ratio entre los cuadrados de las presiones es positivo, tenemos que $g_k \geq 0$, $\forall k \in E^c$.

De esta forma, en la función objetivo aparecerá el término:

$$\mu^{gc} \sum_{k\in E^c} g_k$$

donde μ^{gc} es un parámetro positivo que determinará el peso asociado al consumo de gas en las estaciones de compresión.

1.3.2. Diferencia de presión.

En la Sección 1.2 vimos que se modelaban compresores bidireccionales y de forma equivalente, se procedía a una duplicación de la arista para así tener un único compresor en cada una de ellas. A continuación vamos a ver los casos que se pueden presentar según las presiones que tenga el gas en cada nodo. Sean $k_1 = (i, l), k_2 = (j, l) \in E^c$ y denotemos por p_i^d, p_j^d y p_l^d las presiones relativas a los nodos i, j y l después de la duplicación. Si suponemos que $q_{k_1} \geq 0$ y teniendo en cuenta que se debe verificar que $p_i^d \leq p_l^d$ y $p_j^d \leq p_l^d$ se tienen los siguientes casos:

- Si $p_i^d \leq p_l^d$ y $p_l^d = p_j^d$ entonces se tiene que $p_i \leq p_j$, es decir, el compresor original aumenta la presión en el sentido del flujo, de i a j.
- Si $p_i^d = p_l^d$ y $p_l^d \geq p_j^d$ entonces se tiene que $p_j \leq p_i$, es decir, el compresor original está funcionando como una válvula, ya que reduce la presión en el sentido del flujo.
- Si $p_i^d \leq p_l^d$ y $p_l^d \geq p_j^d$ se distinguen dos subcasos:
  - $p_i^d \leq p_j^d$. En este caso $p_i \leq p_j$, el compresor original aumenta la presión en el sentido del flujo.
  - $p_i^d \geq p_j^d$. En este caso $p_j \leq p_i$, el compresor original está funcionando como una válvula ya que se reduce la presión en el sentido del flujo.

Si $q_{k_1} \leq 0$ el razonamiento sería totalmente análogo, por lo que omitimos su explicación.

Nos interesan las situaciones en las que únicamente uno de los dos compresores estén modificando la presión del flujo, es decir, los dos primeros casos pues en el último de ellos los dos compresores están

funcionando, uno como compresor y otro como válvula. Para evitar esta última situación introducimos una penalización en la función objetivo. Definimos Δ_k^c como la diferencia de los cuadrados de las presiones en el compresor k:

$$\Delta_k^c = u_j - u_i \tag{1.27}$$

Por la ecuación (1.5) tenemos que $\Delta_k^c \geq 0$. Entonces nos interesa minimizar la suma de estas diferencias, introduciendo en la función objetivo el término

$$\mu^c \sum_{k \in E^c} \Delta_k^c$$

donde $\mu^c > 0$ es el parámetro de penalización de la diferencia de presión en los compresores. Este parámetro se ajustará según la importancia que le queramos asociar a dicho término.

Por otro lado, recordemos que las válvulas de control de presión (PCV) permitían reducir la presión del gas en la dirección en la que circule el flujo. En situaciones de máxima demanda de la red, se ve que las soluciones obtenidas tienen un gran número de válvulas PCV activas y que la reducción de presión que realizan muchas de ellas es muy pequeña. Sin embargo, es preferible tener soluciones con un menor número de válvulas activas y una mayor reducción de presión asociada a cada una de ellas. De esta forma, surge la necesidad de incluir una penalización en la función objetivo para poder satisfacer esta preferencia.

Dado $k = (i, j) \in E^{pcv}$, dependiendo de la dirección del flujo en la arista k tendremos que la diferencia $u_j - u_i$ es positiva o negativa. Si $q_k \geq 0$ entonces $u_j - u_i \leq 0$, en cambio si $q_k \leq 0$ se tiene que $u_j - u_i \geq 0$. Por lo tanto, definimos la diferencia del cuadrado de las presiones como:

$$\Delta_k^{pcv} = \text{máx}\{u_i - u_j, u_j - u_i\}$$

pues así tenemos $\Delta_k^{pcv} \geq 0$ independientemente del sentido en el que circule el flujo por la arista k. Sin embargo, para solucionar la no linealidad de esta expresión podemos definir las siguientes restricciones:

$$\Delta_k^{pcv} \geq u_i - u_j \tag{1.28}$$

$$\Delta_k^{pcv} \geq u_j - u_i \tag{1.29}$$

introduciendo también en la función objetivo el término:

$$\mu^{pcv} \sum_{k \in E^{pcv}} \Delta_k^{pcv}$$

donde μ^{pcv} es un parámetro positivo que determinará el peso que le queramos dar a la diferencia de presión en las válvulas PCV.

1.3.3. Evaporación del gas en las plantas de regasificación.

Cuando las plantas de regasificación introducen gas en la red, el sistema de *boil-off* de la planta recoge los vapores del gas natural, los calienta hasta alcanzar aproximadamente una temperatura ambiente, lo comprime y lo introduce de nuevo en el sistema de distribución del gas, proceso que tiene un coste asociado. Denotemos por N^{rp} el conjunto de nodos que representan plantas de regasificación en la red, por bo^p el consumo *boil-off* de la planta $p \in N^{rp}$ y por Q_p la variable no negativa que represente la cantidad de gas aportada por dicha planta p:

$$Q_p = \sum_{k \in E_p^{ini}} q_k - \sum_{k \in E_p^{fin}} q_k \tag{1.30}$$

Podemos definir la función de coste *boil-off*, $B(Q_p)$, como:

$$B(Q_p) = \begin{cases} P_c & \text{si} \quad Q_p = 0 \\ TD_c - \frac{TD_c}{O_s - TD_s}(Q_p - TD_s) + D_c Q_p & \text{si} \quad TD_s \leq Q_p \leq O_s \\ D_c Q_p & \text{si} \quad Q_p \geq O_s \end{cases}$$

donde P_c es el consumo de la planta de regasificación cuando no está aportando gas a la red, $TD = (TD_s, TD_c)$ es el punto de bajada de la planta, que determina la cantidad mínima de gas que puede ser aportada, TD_s, y el consumo en este caso, TD_c. Por otra parte, O_s representa el punto de suministro óptimo para el cuál el consumo de la planta es despreciable y D_c el consumo de descarga por unidad de gas suministrado desde la planta.

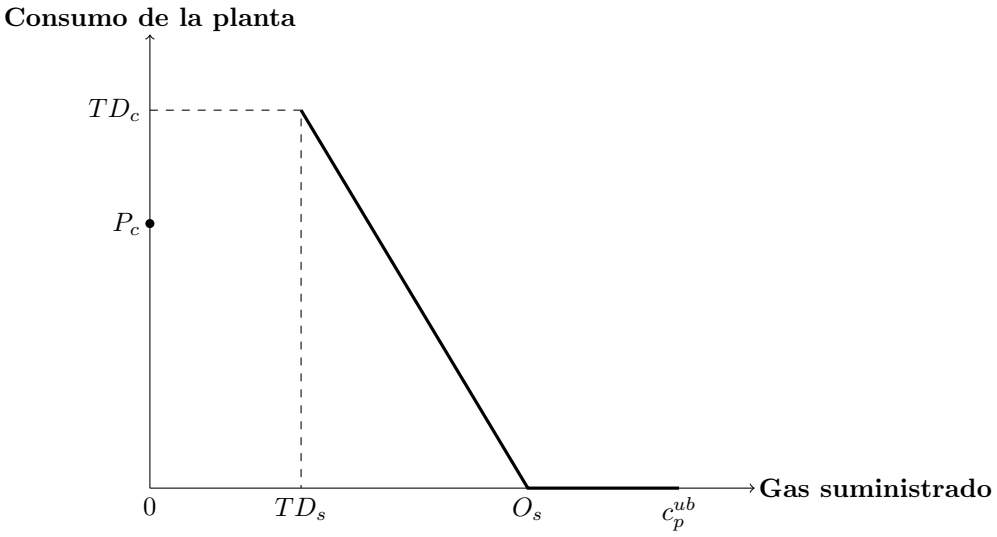


Figura 1.3: Representación gráfica de la función de coste *boil-off*.

En la Figura 1.3 se puede ver la representación gráfica de la función de coste *boil-off*, sin considerar el consumo de descarga, D_c, ya que se considerará despreciable.

Una vez tenemos definida la función de coste *boil-off*, vamos a ver como modelarla. Para empezar, definimos una variable binaria, $y_p^{bo} \in \{0, 1\}$, que nos indique si la planta $p \in N^{rp}$ está activa o no:

$$Q_p \geq TD_s y_p^{bo} \tag{1.31}$$

$$Q_p \leq c_p^{ub} y_p^{bo} \tag{1.32}$$

Como podemos observar, si la planta está activa ($y_p^{bo} = 1$), la cantidad de gas aportada por la planta p está acotada inferior y superiormente, $TD_s \leq Q_p \leq c_p^{ub}$. En el caso de que la planta no esté activa, $y_p^{bo} = 0$, esta no introduce gas a la red pues $Q_p = 0$.

Por otro lado, definimos la variable r_p como el máximo entre 0 y la función lineal que une los puntos TD y O_s:

$$r_p = \text{máx}\{0, TD_c - \frac{TD_c}{O_s - TD_s}(Q_p - TD_s)\}$$

Dado que esta nueva variable r_p aparecerá en la función objetivo, nos interesa evitar las no linealidades, por lo que equivalentemente la definimos mediante las siguientes ecuaciones:

$$r_p \geq 0 \tag{1.33}$$

$$r_p \geq TD_c - \frac{TD_c}{O_s - TD_s}(Q_p - TD_s) \tag{1.34}$$

Finalmente, ya podemos definir el coste asociado al *boil-off* en las plantas de regasificación, de forma que nos queda:

$$bo_p = (1 - y_p^{bo})P_c + D_c Q_p + r_p - (1 - y_p^{bo})\left(TD_c + \frac{TD_c}{O_s - TD_s}TD_s\right) \tag{1.35}$$

En el caso de que la planta de regasificación esté activa se tiene que $bo_p = D_c Q_p + r_p$. Por otro lado, si la planta de regasificación no está funcionando, $y_p^{bo} = 0$, se tiene, por las ecuaciones (1.31) y (1.32) que $Q_p = 0$ y por lo tanto, $r_p = TD_c + \frac{TD_c}{O_s - TD_s}TD_s$, así nos quedaría:

$$bo_p = P_c + r_p - \left(TD_c + \frac{TD_c}{O_s - TD_s}TD_s\right) = P_c$$

De esta forma, en la función objetivo aparecerá el término:

$$\mu^{bo} \sum_{p \in N^{rp}} bo_p$$

donde μ^{bo} es un parámetro positivo que determinará el peso asociado al coste de *boil-off*.

1.3.4. Linepack.

El *linepack* es un mecanismo que consiste en utilizar el gas almacenado en las propias tuberías de la red. Si consideramos una estimación de la demanda, la optimización del *linepack* surge como una herramienta para superar los cambios inesperados que se puedan producir en la demanda. Es por esto, que puede ser interesante modelar la red de forma que se maximice la cantidad de gas almacenado en las propias tuberías.

La capacidad de almacenamiento de un conducto de la red puede ser calculada como la diferencia entre las cantidades de gas dentro del conducto bajo condiciones de empaquetado y desempaquetado. Diremos que un conducto está en condición de empaquetado cuando la cantidad de gas que sale del conducto es el mínimo posible, con una presión de descarga máxima para poder suministrar gas constantemente. Por otro lado, diremos que el conducto está en condición de desempaquetado cuando la cantidad de gas que sale del mismo es el máximo posible con una presión de descarga mínima para poder suministrar gas constantemente.

Considerando la ecuación de estado para gases reales, $p = Z(p, \theta)\rho R\theta$, tenemos que la densidad del gas que circula a través de un conducto es:

$$\rho = \frac{p}{Z(p, \theta)R\theta}.$$

Sea $k = (i, j) \in E^{st}$, aproximamos la densidad del gas a través del conducto usando la presión media, $p_{m_k} = \dfrac{p_i + p_j}{2}$. Para calcular el *linepack* debemos multiplicar la densidad del conducto por su volumen, quedándonos lo siguiente:

$$\rho_k v_k = \frac{p_i + p_j}{2Z(p_{m_k}, \theta)R\theta}\pi L_k \left(\frac{D_k}{2}\right)^2.$$

donde v_k ($[m^3]$) es el volumen del conducto y estamos suponiendo que la sección del mismo es circular. Dado que en nuestro modelo estamos trabajando con las presiones al cuadrado, u_i y u_j, nos quedaría

$$lp_k = \frac{u_i + u_j}{2Z(\sqrt{u_{m_k}}, \theta)R\theta}\pi L_k \left(\frac{D_k}{2}\right)^2 \tag{1.36}$$

que aumenta o disminuye en el mismo sentido que la ecuación anterior.

De esta forma, en la función objetivo se implementará la siguiente expresión:

$$\mu^{lp} \sum_{k \in E^{st}} lp_k$$

donde μ^{lp} es un parámetro negativo que determinará el peso asociado al *linepack*. Este parámetro es negativo porque en nuestro problema nos interesa minimizar la función objetivo, sin embargo, tratamos de maximizar el *linepack*.

Capítulo 2

Optimización en redes de gas.

Como acabamos de ver en el Capítulo 1, estamos ante un problema de optimización matemática no lineal y no convexo, NLP (*Non Linear Problem*), cuya principal dificultad se encuentra en la no convexidad del mismo, ya que optimalidad local no equivale a optimalidad global, hecho que sí ocurre en los problemas de programación convexa. Recordemos que para hablar de un problema de programación convexa, tanto la función objetivo como la región factible del mismo deben de ser convexos, puede consultarse González Díaz (2017).

Además nos encontramos con otra complejidad ya que no tenemos un problema puramente continuo, pues a las variables continuas fundamentales del problema como son los flujos q_k y las presiones p_i, se añaden las variables binarias necesarias para controlar otros elementos de la red como válvulas y compresores. Es por esto que el problema resultante se conoce como problema de Programación No Lineal Entera Mixta, MINLP por sus siglas en inglés. Esto supone añadir una capa de optimización combinatoria por encima de un problema ya de por sí complejo.

Por todo esto, y dado el tamaño de problemas realistas de optimización de redes, normalmente se opta por algoritmos que garanticen la optimalidad local de las soluciones escogidas. A lo largo de este capítulo vamos a presentar los distintos algoritmos que usaremos en este trabajo, centrándonos especialmente en el algoritmo 2-SLP desarrollado por el equipo de GANESO™, expuesto en la Sección 2.1.

2.1. Algoritmo 2-SLP.

Como vimos en el Capítulo 1 estamos ante un problema de Programación No Lineal Entera Mixta (MINLP), ya que además de las variables continuas principales del problema, para su correcta modelización fue necesario incluir ciertas variables binarias. Sin embargo, este tipo de variables tienen un cierto inconveniente ya que un algoritmo de programación lineal sucesiva con región de confianza, como el SLP clásico, es incompatible con ellas. Para afrontar este problema, el equipo de GANESO™ desarrolló un algoritmo a medida al que denotaremos por 2-SLP.

Cuando hablamos del 2-SLP nos estamos refiriendo a un algoritmo compuesto por dos pasos, donde una primera etapa sería aplicar el algoritmo de Programación Lineal Sucesiva sin Región de Confianza, SLP-NTR por sus siglas en inglés, para el problema MINLP de forma que obtenemos una solución para las variables binarias. De esta forma, fijando como valores de dichas variables las dadas en la solución anterior pasamos a resolver, en una segunda etapa, el problema NLP resultante mediante el algoritmo de Programación Lineal Sucesiva Penalizada, PSLP. A continuación, vamos a explicar brevemente cada uno de estos algoritmos, se puede consultar González Díaz (2017), González Rueda et al. (2019) y Bazaraa et al. (2006).

Para empezar, vamos a presentar la formulación general de un problema de programación no lineal (P), así como la versión linealizada del mismo entorno a un punto $\bar{\boldsymbol{x}}$ dado:

Problema P	Problema LP($\bar{\boldsymbol{x}}$)
minimizar $f(x)$	minimizar $f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x})$
sujeto a $g_i(x) \leq 0 \quad i = 1, \ldots, m$	sujeto a $g_i(\bar{x}) + \nabla g_i(\bar{x})^\top (x - \bar{x}) \leq 0 \quad i = 1, \ldots, m$
$h_j(x) = 0 \quad j = 1, \ldots, l$	$h_j(\bar{x}) + \nabla h_j(\bar{x})^\top (x - \bar{x}) = 0 \quad j = 1, \ldots, l$
$x \in \mathbf{S} = \{x : \mathbf{A}x \leq \mathbf{b}\}$	$x \in \mathbf{S} = \{x : \mathbf{A}x \leq \mathbf{b}\}$

donde f, $g_i \ \forall \ i \in \{1, \ldots, m\}$ y $h_j \ \forall \ j \in \{1, \ldots, l\}$ son funciones no lineales de clase C^1 y S es el conjunto de restricciones lineales del problema.

De esta forma, la idea del SLP es la siguiente. En cada iteración t dispondremos de un iterante $\boldsymbol{x}^t$, se resolverá el problema LP($\boldsymbol{x}^t$) obteniendo un punto $\boldsymbol{x}^{t+1}$, y así sucesivamente hasta que tengamos convergencia, llegando así a un punto óptimo para el problema linealizado, y por lo tanto, KKT del problema P.

En nuestro caso, como comentamos al principio de la sección, estamos ante un problema MINLP por lo que en la primera etapa aplicamos el algoritmo SLP-NTR.

Método SLP-NTR (programación lineal sucesiva sin región de confianza)

Determinamos un parámetro de convergencia $\varepsilon > 0$ y un punto inicial $\boldsymbol{x}^1 \in \mathbf{S}$. Además, definimos $t = 1$.

PASO 1:

Se define $\boldsymbol{x}^{t+1}$ como solución óptima del problema LP($\boldsymbol{x}^t$).

- Si $\dfrac{\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^t\|}{\|\boldsymbol{x}^t\| + \varepsilon} \leq \varepsilon$, devolvemos $\boldsymbol{x}^{t+1}$.
- En otro caso, consideramos $t = t + 1$ y repetimos el PASO 1.

Una vez encontrada una solución para el problema MINLP, tenemos determinados los valores de las variables binarias del problema. Fijando dichas variables nos queda entonces un problema NLP que resolveremos por el método de PSLP (Programación Lineal Sucesiva Penalizada). En este caso, como estamos ante un problema NLP podemos establecer una región de confianza que controle la distancia entre iterantes consecutivos, obteniendo de esta forma mejores garantías de convergencia.

La versión penalizada del problema de programación no lineal P es la siguiente:

Problema P^ρ

$$\text{minimizar}_{x \in \mathbf{S}} f(x) + \rho \left(\sum_{i=1}^{m} \text{máx}\{0, g_i(x)\} + \sum_{j=1}^{l} |h_j(x)| \right)$$

De esta forma, es fácil ver que las restricciones del problema P^ρ son lineales. Además, se definen para $i \in \{1, \ldots, m\}$ una variable $y_i = \text{máx}\{0, g_i(x)\}$ y para $j \in \{1, \ldots, l\}$ las variables $h_j^+ \geq 0$ y $h_j^- \geq 0$, de forma que $|h_j(x)| = h_j^+ + h_j^-$. Aplicando estos cambios se puede expresar el problema de la siguiente manera:

Problema P^ρ

$$\begin{aligned}
\text{minimizar } & f(x) + \rho\left(\sum_{i=1}^{m} y_i + \sum_{j=1}^{l}(h_j^+ + h_j^-)\right) \\
\text{sujeto a } & y_i \geq g_i(x) && i = 1, \ldots, m \\
& h_j^+ - h_j^- = h_j(x) && j = 1, \ldots, l \\
& y_i \geq 0 && i = 1, \ldots, m \\
& h_j^+ \geq 0, h_j^- \geq 0 && j = 1, \ldots, l \\
& x \in \mathbf{S} = \{x : \mathbf{A}x \leq \mathbf{b}\}
\end{aligned}$$

Al igual que en el caso del algoritmo SLP se resolverá iterativamente el problema linealizado $\mathrm{LP}^\rho(\bar{\boldsymbol{x}})$ teniendo en cuenta que en este caso se define una región de confianza, obteniendo así cierto control sobre la distancia existente entre dos iterantes consecutivos y asegurando que cada solución sea mejor que la anterior con respecto al problema P^ρ, mejorando las propiedades de convergencia del algoritmo. De esta forma nos queda el siguiente problema:

Problema $\mathrm{LP}^\rho(\bar{\boldsymbol{x}}, \mathbf{r})$

$$\begin{aligned}
\text{minimizar } & f(\bar{\boldsymbol{x}}) + \nabla f(\bar{x})^\top \mathbf{d} + \rho\left(\sum_{i=1}^{m} y_i + \sum_{j=1}^{l}(h_j^+ + h_j^-)\right) \\
\text{sujeto a } & y_i \geq g_i(\bar{x}) + \nabla g_i(\bar{x})^\top \mathbf{d} && i = 1, \ldots, m \\
& h_j^+ - h_j^- = h_j(x) + \nabla h_j(\bar{x})^\top \mathbf{d} && j = 1, \ldots, l \\
& y_i \geq 0 && i = 1, \ldots, m \\
& h_j^+ \geq 0, h_j^- \geq 0 && j = 1, \ldots, l \\
& -r_k \leq d_k \leq r_k && k = 1, \ldots, n \\
& x \in \mathbf{S} = \{x : \mathbf{A}x \leq \mathbf{b}\}
\end{aligned}$$

siendo $\mathbf{d} = (x - \bar{x})$ y $\mathbf{r}$ un vector que delimita la región de confianza.

El algoritmo PSLP decide si aceptar o no la solución dada en una iteración dependiendo del decrecimiento de f^ρ en comparación al decrecimiento estimado f_L^ρ, siendo f^ρ la función objetivo del problema P^ρ y f_L^ρ la función objetivo del problema LP^ρ. Para comparar ambos decrecimientos es necesario que definamos dos nuevos parámetros, $\Delta f_t^\rho = f_t^\rho(x^t) - f_t^\rho(x^t + d^t)$ y $\Delta f_{L,t}^\rho = f_{L,t}^\rho(x^t) - f_{L,t}^\rho(x^t + d^t)$ que determinen dichos decrecimientos, y el siguiente ratio:

$$\gamma^t = \frac{\Delta f_t^\rho}{\Delta f_{L,t}^\rho}$$

Método PSLP (programación lineal sucesiva penalizada con región de confianza)

Determinamos un parámetro de convergencia $\varepsilon > 0$ y uno de penalización $\rho > 0$. Elegimos α_1, α_2 y α_3 con $0 < \alpha_1 < \alpha_2 < \alpha_3 < 1$, $\beta > 0$ y $\mathbf{r}^{min} > 0$. Definimos un punto inicial $\boldsymbol{x}^1 \in \mathbf{S}$ y un vector inicial $\mathbf{r}^1 > 0$. Además, definimos $t = 1$.

PASO 1:

Se define $\mathbf{d}^t$ como solución óptima del problema $\mathrm{LP}^\rho(\boldsymbol{x}^t, \mathbf{r}^t)$.

- Si $\Delta f^\rho_{L,t} \leq \varepsilon$ y $\|\mathbf{d}^t\| < \varepsilon$, devolvemos $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \mathbf{d}^t$.
- En otro caso, evaluamos γ^t:
  - Si $\gamma^t < \alpha_1$, definimos $\mathbf{r}^t = \beta \mathbf{r}^t$ y repetimos el PASO 1.
  - Si $\gamma^t \geq \alpha_1$, continuamos con el PASO 2.

PASO 2:

Se define $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \mathbf{d}^t$ y se ajusta la región de confianza:

- Si $\alpha_1 \leq \gamma^t < \alpha_2$, definimos $\mathbf{r}^{t+1} = \beta \mathbf{r}^t$.
- Si $\alpha_2 \leq \gamma^t < \alpha_3$, definimos $\mathbf{r}^{t+1} = \mathbf{r}^t$.
- Si $\gamma^t \geq \alpha_3$, definimos $\mathbf{r}^{t+1} = \dfrac{\mathbf{r}^t}{\beta}$.

Se define para todo $k \in \{1, \ldots, n\}$, $\mathbf{r}_k^{t+1} = \text{máx}\{\mathbf{r}^{min}, \mathbf{r}_k^{t+1}\}$. Se remplaza t por $t+1$ y repetimos el PASO 1.

Como podemos ver, para ejecutar el algoritmo PSLP con región de confianza, debemos especificar una serie de parámetros, entre ellos α_1, α_2, α_3 y β, de forma que verifiquen unas ciertas condiciones. Una posible elección para estos parámetros, puede consultarse Bazaraa et al. (2006), es $\alpha_1 = 10^{-6}$, $\alpha_2 = 0.25$, $\alpha_3 = 0.75$ y $\beta = 0.5$.

2.2. Otras herramientas utilizadas.

Como ya comentamos anteriormente, además de usar el algoritmo 2-SLP creado a medida por el equipo de GANESO™, también se va a comprobar el rendimiento de otras herramientas de optimización local, como Knitro e Ipopt, a través de AMPL y Pyomo. A continuación, vamos a presentar brevemente cada uno de ellos.

Para empezar, es necesario mencionar que AMPL es un lenguaje de modelización y programación matemática, a través del cual podemos expresar un problema de optimización en lenguaje algebraico. Como se puede consultar en Fourer et al. (2003), los modelos de AMPL estarán formados por variables, restricciones y función objetivo, expresados con la ayuda de conjuntos y parámetros. Además, es necesario mencionar que por defecto AMPL realiza ciertas simplificaciones en las restricciones antes de resolver un problema, pudiendo reducir así significativamente el tamaño del mismo. Esto es lo que se conoce por *presolve*.

Por otro lado, Pyomo (Python Optimization Modeling Objects) es un software libre basado en el lenguaje de programación Python para la formulación, resolución y análisis de problemas de optimización. Como se puede consultar en Hart et al. (2017), con Pyomo podemos definir problemas simbólicos generales, crear instancias específicas de un problema y resolver las mismas.

Tanto en AMPL como en Pyomo, podemos resolver un problema usando diferentes solvers, en

nuestro caso, usaremos Ipopt y Knitro. Ambos son solvers de optimización local, es decir, en caso de encontrar solución será un óptimo local, lo que no nos asegura que sea el óptimo global, ya que estamos ante problemas de programación no convexa. Ipopt (Interior Point OPTimizer) es un solver diseñado para encontrar soluciones locales de problemas de optimización no lineal, se puede consultar Wätcher et al. (2015). Por otra parte, como se puede consultar en Byrd et al. (2006), Knitro es un solver diseñado para encontrar soluciones óptimas locales en problemas no lineales tanto de optimización continua como de optimización discreta con variables enteras o binarias.

De esta forma, a lo largo de este trabajo se probarán distintas configuraciones del software GANESO™ para resolver los problemas que se presentarán en el Capítulo 3. Algunas de estas configuraciones se apoyarán principalmente en el algoritmo 2-SLP, mientras que otras lo harán en los solvers que acabamos de mencionar.

Capítulo 3

Propagación de calidad.

Como vimos a lo largo del Capítulo 1, se está modelando el problema de forma que a cada nodo llegue una cierta cantidad de gas con una presión adecuada. Sin embargo, dependiendo de la composición del gas, este tendrá una determinada calidad, que mediremos a partir del denominado poder calorífico.

Se define el poder calorífico de un gas PC $[J/kg]$, puede consultarse Koch et al. (2015), como la cantidad de calor generada por la combustión de una unidad de masa de dicho gas. Este valor dependerá principalmente de la composición del propio gas.

En los puntos en los que se mezclen distintos gases, se forma un nuevo gas con una composición distinta, y por lo tanto, con un nuevo poder calorífico. De esta forma, puede consultarse Hager et al. (2012), se puede definir el mismo como una propiedad del gas que se mueve a través de la red a la misma velocidad que este y que se mezcla en los puntos de conexión, es decir, en los nodos a los que llegue gas a través de dos o más conductos distintos.

De esta forma, no solo nos interesa que llegue una cierta cantidad de gas con una determinada presión a los nodos de demanda, si no que es importante que el flujo de gas llegue con un cierto poder calorífico, para así garantizar que llega la energía necesaria. Por este motivo, es importante añadir las restricciones relativas a la propagación de calidad del gas. Para ello, debemos incluir dichas restricciones en el modelo y calibrar la herramienta de optimización GANESO™, procedimientos en los que participé yo misma y que se explicarán a lo largo de este capítulo.

3.1. Restricciones para la propagación de calidad del gas.

Como mencionamos anteriormente, el poder calorífico es una propiedad del gas que se mueve a través de la red a la misma velocidad con la que circula el flujo del mismo, por lo tanto, si una red tuviese un único nodo de suministro por el cual se introduce gas con un determinado poder calorífico, este se mantendría igual por toda la red. En este caso, el gas que llegue a los puntos de demanda llegará con el mismo poder calorífico, independientemente de la cantidad de flujo que llegue a cada uno de ellos.

Debemos tener en cuenta que la situación anterior raramente ocurre, ya que en una red pueden entrar varios tipos de gas distintos, por lo que el poder calorífico de los mismos no siempre coincide. Además, como se puede consultar en Hiller et al. (2018), cuando a un nodo llegan distintos tipos de gas, estos se mezclan, obteniendo así un gas con un poder calorífico distinto a los anteriores. A continuación, vamos a ver las ecuaciones que modelan esta propagación de poder calorífico.

Para empezar, dado $i \in N$ y denotando por PC_i el poder calorífico con el que llega el gas al nodo i, asociadas a esta nueva variable tendremos las siguientes cotas:

$$\underline{PC}_i \leq PC_i \leq \overline{PC}_i \tag{3.1}$$

Ahora bien, como se acaba de comentar, hay casos en los que se introducen gases distintos en la red, estos gases tendrán un poder calorífico distinto debido a la composición de cada uno de ellos. Nos interesa calcular el poder calorífico resultante en los nodos en los que se mezclen los distintos tipos de gas, que dependerá de la cantidad de flujo que se mezcle de cada uno de ellos, así como del poder calorífico correspondiente.

Además, supondremos que la mezcla que se realiza es perfecta. Como consecuencia, el gas que circule por cada uno de los arcos salientes del nodo en el que se realice dicha mezcla será el mismo, y por lo tanto, tendrá el mismo poder calorífico.

Dado un nodo $i \in N^c \cup N^{st}$, es decir, un nodo consumo o estructural, se tiene la siguiente ecuación, denominada regla de Bialek, que determina el poder calorífico resultante al mezclar distintos tipos de gas:

$$PC_i = \frac{\sum_{k=(j,i)\in E_i^{fin}} PC_j q_k}{\sum_{k\in E_i^{fin}} q_k}$$

siendo E_i^{fin} el conjunto de aristas que tienen como nodo final el nodo i, es decir, las aristas incidentes en dicho nodo. Como podemos observar, el poder calorífico en un nodo en el que se mezclan diferentes gases es una media ponderada de los poderes caloríficos de los gases que llegan a dicho nodo. Puede consultarse Hager et al. (2012) y Van der Hoeven (2004).

Nótese que no estamos considerando los nodos suministro, ya que en este tipo de nodos el poder calorífico viene determinado por la propia composición del gas.

Equivalentemente, podemos escribir la ecuación anterior de la siguiente manera:

$$PC_i \sum_{k\in E_i^{fin}} q_k = \sum_{k=(j,i)\in E_i^{fin}} PC_j q_k$$

Supongamos que tenemos la siguiente red formada por 5 nodos y 4 aristas, donde los nodos i y j son puntos de entrada del gas, de cuya composición obtenemos PC_i y PC_j, y queremos calcular el poder calorífico del gas en el resto de los nodos, siendo m y n nodos de consumo. Supongamos también que el flujo de gas circula en el sentido de las aristas, es decir, $q_k \geq 0,\ \forall k \in E$.

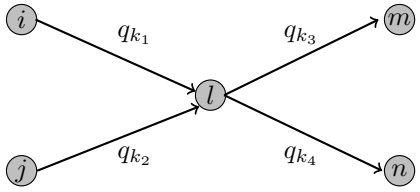


Figura 3.1: Ejemplo de una red formada por 5 nodos y 4 aristas.

En la red tenemos 4 aristas, $k_1 = (i, l)$, $k_2 = (j, l)$, $k_3 = (l, m)$ y $k_4 = (l, n)$. Aplicando la regla de Bialek sobre este ejemplo tendríamos que el poder calorífico para el nodo l sería:

$$PC_l = \frac{PC_i q_{k_1} + PC_j q_{k_2}}{q_{k_1} + q_{k_2}}$$

Por otro lado, para los nodos de demanda m y n se tiene:

$$PC_m = \frac{PC_l q_{k_3}}{q_{k_3}} = PC_l$$
$$PC_n = \frac{PC_l q_{k_4}}{q_{k_4}} = PC_l$$

Supongamos que $PC_i = 10^7\ J/kg$ y $PC_j = 3 \cdot 10^7\ J/kg$, supongamos además que el caudal que se introduce a la red por el nodo i es $q_{k_1} = 20\ kg/s$, el que se introduce por el nodo j es $q_{k_2} = 80\ kg/s$, y que las demandas de los nodos m y n son de 40 kg/s y 60 kg/s, respectivamente. Fácilmente podemos ver que para resolver este problema se tiene $q_{k_3} = 40\ kg/s$ y $q_{k_4} = 60\ kg/s$. Para simplificar las operaciones, en lo referente al poder calorífico trabajaremos con unidades por millón, por lo que tendremos que los poderes caloríficos en los nodos i y j serán 10 y 30, respectivamente. De esta forma, aplicando la regla de Bialek tenemos:

$$PC_l = \frac{PC_i q_{k_1} + PC_j q_{k_2}}{q_{k_1} + q_{k_2}} = \frac{10 \cdot 20 + 30 \cdot 80}{20 + 80} = \frac{200 + 2400}{100} = 26,$$
$$PC_m = \frac{PC_l q_{k_3}}{q_{k_3}} = PC_l = 26,$$
$$PC_n = \frac{PC_l q_{k_4}}{q_{k_4}} = PC_l = 26.$$

Como podemos observar en el ejemplo anterior, en la red entran dos gases, a través de los nodos i y j, con distinta composición, y por lo tanto distinto poder calorífico, que se mezclan en el nodo l, formando así un nuevo gas con un poder calorífico distinto.

Hasta ahora estamos considerando que conocemos el sentido en el que circula el gas por cada una de las aristas de la red, sin embargo en nuestro caso, desconocemos dicho sentido. Como consecuencia, no es tan sencillo definir la regla de Bialek para saber el poder calorífico de la mezcla que se produce, ya que para ello es necesario conocer el sentido del flujo en cada arista.

Como ya tenemos mencionado anteriormente a lo largo del Capítulo 1, dada una arista $k = (i, j) \in E$ se tiene $q_k \geq 0$ si el flujo circula en el sentido de la arista, de i a j, y $q_k \leq 0$ si lo hace en sentido contrario, del nodo j al nodo i.

Para solucionar este problema se lleva a cabo la siguiente descomposición, que consiste en dividir q_k en dos componentes, una parte positiva y otra parte negativa, mediante las variables q_k^+ y q_k^- ambas no negativas, de forma que:

$$q_k = q_k^+ - q_k^- \tag{3.2}$$

donde q_k^+ representa la cantidad de flujo que circula en el sentido de la arista k y q_k^- la cantidad de flujo que circula en sentido contrario.

Sin embargo, esta descomposición no basta por sí sola, ya que por una arista $k \in E$ solo puede circular flujo en un sentido, es decir, una de las dos componentes debe de ser 0. Para modelizar esto es necesario introducir en la función objetivo el siguiente término, penalizándolo de forma que tome un valor lo más pequeño posible:

$$q_k^+ + q_k^-$$

Como podemos ver, esta expresión es una linealización del valor absoluto, ya que $|q_k| = q_k^+ + q_k^-$. De esta forma, evitamos una no linealidad en la función objetivo, pues trataremos de minimizar $q_k^+ + q_k^-$ para que una de sus componentes sea 0.

Al llevar a cabo este procedimiento, queda definido el sentido en el que circula el flujo en cada arista, ya que si $q_k > 0$ tendremos que $q_k^+ = q_k$ y $q_k^- = 0$, es decir, el gas fluye en el sentido de la arista. Por otro lado, si $q_k < 0$ se tiene que $q_k^+ = 0$ y $q_k^- = -q_k$, por lo tanto, el gas fluye en sentido contrario.

Teniendo en cuenta esta descomposición ya podemos definir la regla de Bialek, de forma que tomando como ejemplo la red anterior (véase la Figura 3.1) tendríamos que el poder calorífico de la mezcla que se produce en el nodo l es:

$$PC_l = \frac{PC_i q_{k_1}^+ + PC_j q_{k_2}^+ + PC_m q_{k_3}^- + PC_n q_{k_4}^-}{q_{k_1}^+ + q_{k_2}^+ + q_{k_3}^- + q_{k_4}^-}$$

De esta forma vemos que para calcular el poder calorífico en un nodo debemos tener en cuenta todas las aristas en las que interviene dicho nodo.

Dado un nodo $i \in N^c \cup N^{st}$, denotando por E_i^{ini} el conjunto de aristas que tengan como nodo inicial el nodo i y por E_i^{fin} el conjunto de aristas que tengan dicho nodo como nodo final, tendremos que:

$$PC_i = \frac{\displaystyle\sum_{k=(j,i)\in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j)\in E_i^{ini}} PC_j q_k^-}{\displaystyle\sum_{k\in E_i^{fin}\cup E_i^{ini}} q_k}$$

equivalentemente, llegamos a la siguiente restricción:

$$PC_i \sum_{k\in E_i^{fin}\cup E_i^{ini}} q_k = \sum_{k=(j,i)\in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j)\in E_i^{ini}} PC_j q_k^- \tag{3.3}$$

Volviendo al ejemplo de la Figura 3.1, supongamos que ahora tenemos que los nodos de suministro son los nodos i, m y n con un poder calorífico asociado a cada uno, en unidades por millón, de 10, 30 y 50, respectivamente. Además por el nodo i entran 20 unidades de flujo, por el nodo m 50 unidades de flujo y por el nodo n 30 unidades. Por último, supongamos que la demanda del nodo j es de 100 unidades de flujo. En este caso tenemos:

- Por la arista $k_1 = (i, l)$ circulan 20 unidades de flujo en el sentido de la arista, por lo tanto se tiene $q_{k_1}^+ = 20$ y $q_{k_1}^- = 0$.
- Por la arista $k_2 = (j, l)$ circulan 100 unidades de flujo en sentido contrario, por lo tanto se tiene $q_{k_2}^+ = 0$ y $q_{k_2}^- = 100$.
- Por la arista $k_3 = (l, m)$ circulan 50 unidades de flujo en sentido contrario a la definición de la arista, por lo tanto se tiene $q_{k_3}^+ = 0$ y $q_{k_3}^- = 50$.
- Por la arista $k_4 = (l, n)$ circulan 30 unidades de flujo en el sentido contrario de la arista, por lo tanto se tiene $q_{k_4}^+ = 0$ y $q_{k_4}^- = 30$.

Una vez definidas las variables q_k^+ y q_k^- para toda arista k de la red, podemos aplicar la regla de Bialek para calcular el poder calorífico, obteniendo:

$$PC_l = \frac{PC_i q_{k_1}^+ + PC_j q_{k_2}^+ + PC_m q_{k_3}^- + PC_n q_{k_4}^-}{q_{k_1}^+ + q_{k_2}^+ + q_{k_3}^- + q_{k_4}^-} = \frac{10\cdot 20 + PC_j \cdot 0 + 30\cdot 50 + 50\cdot 30}{20+0+50+30} = 32,$$

$$PC_j = \frac{PC_l q_{k_2}^-}{q_{k_2}^-} = PC_l = 32.$$

Como acabamos de ver, se descompone el flujo que circula por una arista en dos componentes no negativas, q_k^+ y q_k^-, de forma que $q_k = q_k^+ - q_k^-$. Sin embargo, esta descomposición no era suficiente, si no que para que alguna de ellas fuese cero lo que hacíamos era minimizar en la función objetivo el valor absoluto del flujo, $|q_k| = q_k^+ + q_k^-$.

Sin embargo, debemos minimizar dicha cantidad para cada una de las aristas de la red, es decir, que tratamos de minimizar la siguiente expresión:

$$\mu^{abs} \sum_{k \in E} (q_k^+ + q_k^-)$$

donde μ^{abs} es un parámetro positivo que determinará el peso que le queremos asignar a este término.

3.2. Propagación de calidad penalizada.

En la sección anterior acabamos de ver las restricciones asociadas a la propagación de calidad del gas. Sin embargo, puede ser interesante pasar estas restricciones a la función objetivo, penalizándolas.

Recordemos que para cada nodo $i \in N$ teníamos la siguiente restricción:

$$PC_i \sum_{k \in E_i^{fin} \cup E_i^{ini}} q_k = \sum_{k=(j,i) \in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j) \in E_i^{ini}} PC_j q_k^-, \tag{3.3}$$

equivalentemente, se tiene que:

$$PC_i \sum_{k \in E_i^{fin} \cup E_i^{ini}} q_k - \left(\sum_{k=(j,i) \in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j) \in E_i^{ini}} PC_j q_k^- \right) = 0.$$

Debemos tener en cuenta que esa diferencia puede ser negativa, por lo que para poder añadirla a la función objetivo lo haremos con la restricción al cuadrado, asegurándonos así que al minimizarla estamos buscando soluciones de forma que se aproxime a cero.

De esta forma, es fácil ver que lo que nos interesa es minimizar el cuadrado de la diferencia anterior para todo nodo de la red, de forma que se aproxime lo máximo posible a cero. Esto se consigue añadiendo a la función objetivo el siguiente término:

$$\mu^{PCpen} \sum_{i \in N} \left(PC_i \sum_{k \in E_i^{fin} \cup E_i^{ini}} q_k - \left(\sum_{k=(j,i) \in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j) \in E_i^{ini}} PC_j q_k^- \right) \right)^2,$$

donde μ^{PCpen} es el parámetro de penalización, es decir, es un parámetro positivo que determinará el peso que le queremos asociar a la restricción de propagación de calidad en la función objetivo.

3.3. Conservación de flujo en energía.

Una vez implementada la propagación de calidad del gas explicada en la Sección 3.1, seremos capaces de modificar las ecuaciones relativas a la conservación del flujo que teníamos hasta ahora, de forma que hablemos del caudal de flujo en energía y no en masa.

Para empezar debemos recordar que la ecuación de conservación de flujo establecía que la cantidad de flujo que entraba a un nodo debía ser igual a la cantidad de flujo que salía del mismo, teniendo en cuenta los intercambios producidos con el exterior, que denotamos por c_i.

$$\sum_{k\in E_i^{ini}} q_k - \sum_{k\in E_i^{fin}} q_k = c_i,\ \forall i \in N.$$

Dado que ya tenemos implementada la propagación de calidad del gas, es fácil calcular el poder calorífico con el que llega el gas a cada nodo y de esta forma, dado un nodo $i \in N$ podemos calcular el caudal de flujo en energía. Como se puede consultar en Frøysa y Lunde (2005), el caudal másico y el caudal en energía de un gas siguen la siguiente relación:

$$c_i^{ener} = \frac{PC_i c_i}{\rho_0},$$

donde ρ_0 es la densidad del gas en condiciones normales, calculada a partir de la ecuación de estado para gases reales. De esta forma tenemos:

$$c_i^{ener} = \frac{24 PC_i c_i}{10^9}.$$

Aplicando este cambio de unidades, somos capaces de escribir la restricción asociada a la conservación de flujo en energía, de forma que nos queda:

$$\sum_{k\in E_i^{ini}} q_k - \sum_{k\in E_i^{fin}} q_k = \frac{10^9}{24 PC_i} c_i^{ener},\ \forall i \in N,$$

o, equivalentemente:

$$24 \cdot 10^{-9} PC_i \left(\sum_{k\in E_i^{ini}} q_k - \sum_{k\in E_i^{fin}} q_k \right) = c_i^{ener},\ \forall i \in N. \tag{3.4}$$

De esta forma, podemos trabajar con cotas de flujo en energía.

3.4. Líneas de ataque y proceso de calibrado.

Como bien mencionamos en la Sección 3.1, nos interesa introducir en la herramienta matemática las restricciones asociadas a la propagación del gas, ya que nos interesa que el gas llegue a los nodos de demanda con un determinado poder calorífico. Dada la complejidad del problema resultante, pues estamos añadiendo una familia de restricciones no lineales a un problema ya complejo, se han estudiado distintas líneas de ataque para su resolución.

Recordemos que si denotamos por PC_i el poder calorífico del gas en el nodo i, este debe estar comprendido entre unas cotas específicas:

$$\underline{PC}_i \leq PC_i \leq \overline{PC}_i \tag{3.1}$$

Por otro lado, para todo $i \in N$ dicha propagación debe verificar la regla de Bialek, de donde obtenemos la siguiente restricción:

$$PC_i \sum_{k\in E_i^{fin}\cup E_i^{ini}} q_k = \sum_{k=(j,i)\in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j)\in E_i^{ini}} PC_j q_k^- \tag{3.3}$$

donde E_i^{fin} representa el conjunto de aristas que tienen el nodo i como nodo final, E_i^{ini} el conjunto de aristas que tienen como nodo inicial al nodo i, q_k^+ la cantidad de flujo que circula por la arista k en el sentido de dicha arista y q_k^- la cantidad de flujo que circula por la arista k en sentido contrario, verificando que el flujo que circula por la arista k es $q_k = q_k^+ - q_k^-$.

Sin embargo, como se explicó anteriormente, esta descomposición del caudal implica que debemos introducir en la función objetivo el siguiente término

$$\mu^{abs} \sum_{k \in E} |q_k| = \mu^{abs} \sum_{k \in E} (q_k^+ + q_k^-)$$

pues de esta forma obtenemos soluciones de forma que el flujo que circula por la arista k lo haga en un único sentido.

Cabe destacar que a lo largo de todo el proceso de calibrado se dieron casos en los que obteníamos soluciones factibles a pesar de que no eran válidas, pues obteníamos soluciones con aristas en las que tanto q^+ como q^- eran distintos de cero, es decir, en estas aristas circulaba flujo tanto en el sentido de la misma como en el contrario. En estos casos tratamos de ajustar el parámetro de penalización μ^{abs}, llegando así a soluciones verdaderamente factibles.

Por otra parte, como se había mencionado en el Capítulo 2, AMPL por defecto tiene activada la opción de *presolve*, por lo que antes de resolver el problema se realizaban ciertas simplificaciones en las restricciones. Estas simplificaciones, por muy pequeñas que sean, pueden afectar a los resultados por lo que decidimos desactivar dicha opción, obteniendo mejores soluciones con ambos solvers (Ipopt y Knitro) a través de AMPL.

Una vez definidas las restricciones correspondientes a la propagación de calidad, se determinan 7 escenarios de 'juguete' (*toy examples*) para comprobar el funcionamiento de la herramienta. Además, denotaremos por modelo básico del problema, aquel cuyas únicas restricciones son las de conservación de flujo y pérdida de presión con las respectivas cotas de las variables correspondientes. En el caso de que el escenario contenga alguna estación de compresión, también tendremos en cuenta las restricciones correspondientes a las mismas en el modelo básico.

A continuación, se explica el proceso de calibrado llevado a cabo paso por paso. Para facilitar la comprensión del mismo, nos apoyaremos en los escenarios 1 y 3, cuyos grafos podemos ver en las Figuras 3.2 y 3.3, respectivamente.

3.4.1. Modelo básico con restricciones de propagación de calidad.

Comenzamos resolviendo el modelo básico con las restricciones de propagación de calidad incluidas, es decir, tenemos en cuenta las restricciones de conservación de flujo, de pérdida de presión y de propagación de calidad, así como las correspondientes a las estaciones de compresión en el caso en que estas pertenezcan al escenario. De esta forma, en la mayoría de los escenarios no obtenemos una solución factible. Esto podría deberse a que las unidades en las que medimos el poder calorífico son mucho mayores a las unidades de flujo, por lo que reescalamos la restricción de propagación de calidad. Para ello, multiplicamos a ambos lados de la ecuación por 10^{-6} de forma que no nos quedan escalas tan distintas entre unas unidades y otras (recordemos que cuando hablábamos de $PC_i = 10$ estábamos trabajando con unidades por millón). Al aplicar esto la restricción de propagación de calidad nos quedaría de la siguiente manera:

$$10^{-6} \cdot \left(PC_i \sum_{k \in E_i^{fin} \cup E_i^{ini}} q_k \right) = 10^{-6} \cdot \left(\sum_{k=(j,i) \in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j) \in E_i^{ini}} PC_j q_k^- \right)$$

De esta forma obtenemos soluciones factibles para algún escenario, sin embargo en la mayoría seguimos sin encontrar dicha solución.

Observando alguno de los escenarios, como por ejemplo en el que se presenta en la Figura 3.2, intuimos fácilmente una posible solución que verifique las condiciones propuestas, ya que lo que se busca en este caso es que por el nodo de suministro 5 entre la mínima cantidad de flujo posible de forma que al llegar al nodo consumo, el flujo resultante tenga un PC prácticamente igual a 15, ya que es la cota inferior que le asignamos a dicho nodo. De esta forma, encontraríamos una solución que minimice el gas consumido en la estación de compresión existente en la arista k_3.

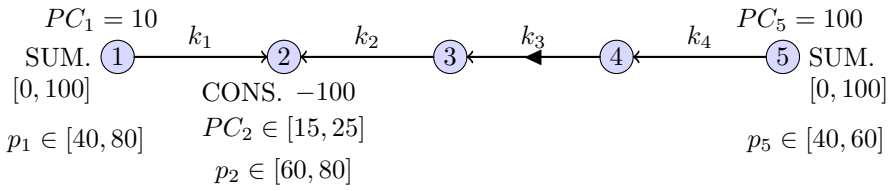


Figura 3.2: Grafo y datos correspondientes al escenario 1.

Sin embargo, la herramienta no es capaz de encontrar dicha solución ni con el solver Knitro ni Ipopt, mientras que a través del algoritmo 2-SLP llega a una solución factible. Resolviendo el modelo básico tanto con Knitro como Ipopt, sí encuentra una solución factible de forma que el caudal que entra en la red por el nodo 1 es de 88.78 unidades y por el nodo 5 entra un caudal de 11.22 unidades. Si ahora consideramos el poder calorífico de los nodos suministro y aplicamos la regla de Bialek obtenemos que el poder calorífico para el nodo de consumo sería:

$$PC_2 = \frac{PC_3 q_{k_2} + PC_1 q_{k_1}}{q_{k_2} + q_{k_1}} = \frac{PC_4 q_{k_3} + PC_1 q_{k_1}}{q_{k_3} + q_{k_1}} = \frac{PC_5 q_{k_4} + PC_1 q_{k_1}}{q_{k_4} + q_{k_1}} = \frac{10 \cdot 88.78 + 100 \cdot 11.22}{100} = 20.098$$

Como observamos, el poder calorífico para el nodo 2 estaría dentro de las cotas, por lo que obtendríamos una solución factible para el modelo básico con propagación de calidad. A pesar de esto, la herramienta no encuentra dicha solución.

3.4.2. Tándem modelo básico y básico con propagación de calidad.

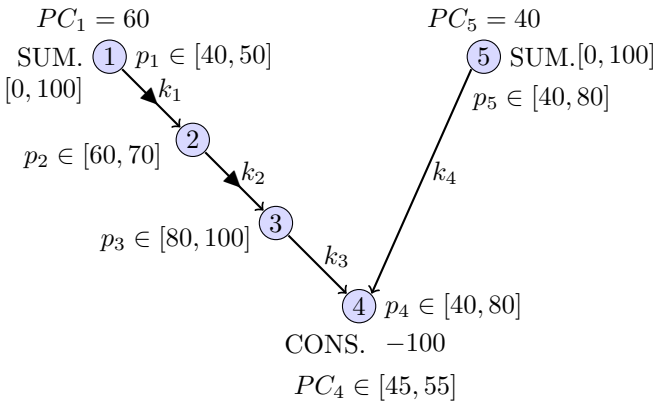


Figura 3.3: Grafo y datos correspondientes al escenario 3.

Para tratar de solventar estos problemas decidimos implementar el tándem del modelo básico y el modelo básico con propagación de calidad. De esta forma lo que hacemos es, en un primer paso, buscar una solución factible para el modelo básico y pasar esta solución como punto inicial al modelo

básico con propagación de calidad. Recordemos que estamos trabajando con la restricción asociada a la propagación de calidad reescalada.

Realizando este tándem obtenemos diversos resultados, por ejemplo, en algunos escenarios en los que antes no conseguíamos una solución factible ahora la herramienta es capaz de encontrarla. Por otro lado, hay casos en los que no se obtiene una solución factible a pesar de que resolviendo únicamente el modelo básico con la restricción de propagación de calidad sí se encontraba. Este último caso ocurría por ejemplo al resolver a través de Pyomo y con Knitro el escenario 3, representado en la Figura 3.3.

Si resolvemos el modelo básico en este caso, obtenemos como solución factible aquella de forma que por el nodo 1 entren 50 unidades de flujo con una presión de 50 Pa y por el nodo 5 otras 50 unidades de flujo con la máxima presión posible, 80 Pa. Si ahora a esta solución le aplicamos la regla de Bialek, obtenemos que el poder calorífico con el que llega el gas al nodo consumo es:

$$PC_4 = \frac{PC_3 q_{k_3} + PC_5 q_{k_4}}{q_{k_3} + q_{k_4}} = \frac{PC_2 q_{k_2} + PC_5 q_{k_4}}{q_{k_2} + q_{k_4}} = \frac{PC_1 q_{k_1} + PC_5 q_{k_4}}{q_{k_1} + q_{k_4}} = \frac{60 \cdot 50 + 40 \cdot 50}{100} = 50$$

Como podemos observar, el poder calorífico del nodo consumo se encuentra entre las cotas, pues $PC_4 = 50 \in [45, 55]$, por lo tanto esta solución sería válida para el modelo con propagación de calidad. Además, esta es la solución factible que obtuvimos al resolver el modelo básico con las restricciones de propagación de calidad. Sin embargo, al pasarle esta solución a dicho modelo como punto inicial, no es capaz de encontrar una solución factible.

3.4.3. Tándem modelo básico con holgura de presión y básico con propagación de calidad.

Antes de desarrollar esta nueva línea de ataque, vamos a presentar brevemente como se introduce esta holgura de presión en el modelo. Para ello se crea una nueva variable no negativa, denotada $pextra$ que tendrá una cota superior $pextra_{max}$ de forma que, para cada nodo i se verifique:

$$0 \leq pextra_i \leq pextra_{max},$$

$$p_i \leq \overline{p}_i + pextra_i.$$

Recordemos que en nuestro modelo trabajábamos con los términos de presiones al cuadrado, por lo que esta última restricción es equivalente a la siguiente:

$$u_i \leq \left(\sqrt{\overline{u}_i} + pextra_i\right)^2.$$

De esta forma, estamos aumentando artificialmente la región factible, sin embargo, es necesario penalizar estas infactibilidades para que sean lo más bajas posibles. Para conseguir esta penalización añadimos a la función objetivo el siguiente término:

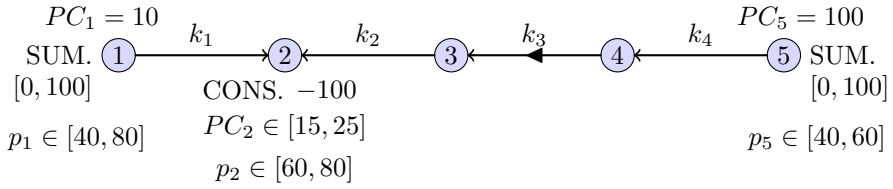
$$\mu^{pextra} \sum_{i \in N} pextra_i$$

donde μ^{pextra} es un parámetro positivo que determinará el peso que le queremos dar a la holgura de la presión.

Una vez desarrollada esta idea, para intentar solucionar los problemas mencionados anteriormente se prueba a lanzar el tándem del modelo básico con $pextra$ y el modelo básico sin $pextra$ con propagación de calidad.

Aplicando este tándem, se encuentra una solución factible para la mayoría de escenarios. Para el escenario 3, representado en la Figura 3.3, encuentra una solución de forma que por el nodo 1 entran 50 unidades de flujo con una presión de 50 Pa, por el nodo 5 entran a la red 50 unidades de flujo con una presión de 80 Pa y en el nodo consumo el poder calorífico con el que llega el gas es $PC_4 = 50$. Como se puede observar, esta es la solución que intuíamos anteriormente. No obstante, para el escenario 1 esto no es suficiente pues seguimos sin obtener una solución factible con Ipopt a través de AMPL y Pyomo.

Recordemos que el grafo y los datos asociados al escenario 1 eran los siguientes:



Como ya mencionamos en el apartado 3.4.1, aplicando Bialek a la solución que encontraba la herramienta para el modelo básico llegábamos a que la calidad del gas en el nodo de consumo estaba comprendido entre las cotas asociadas a dicho nodo, y por lo tanto esta solución debería ser factible para este tándem. A pesar de esto, la herramienta no era capaz de encontrar dicha solución.

3.4.4. Holgura para la calidad del gas.

Debido a los problemas explicados en las subsecciones anteriores, decidimos implementar, al igual que se hizo para las presiones, una holgura inferior para el poder calorífico en cada uno de los nodos.

Para implementar dicha holgura vamos a considerar una nueva variable positiva $infraPC_i, \forall i \in N$. Esta nueva variable estará acotada por un valor máximo que denotaremos por $infraPCmax$.

$$0 \leq infraPC_i \leq infraPCmax \tag{3.5}$$

De esta forma, dado $i \in N$ definimos una nueva restricción para nuestro modelo:

$$\underline{PC}_i - infraPC_i \leq PC_i \tag{3.6}$$

Además, es necesario que modifiquemos las cotas del poder calorífico en cada uno de los nodos, ya que ahora estamos permitiendo una pequeña holgura inferior, de esta forma las cotas asociadas a la variable PC para cada uno de los nodos quedarían de la siguiente manera:

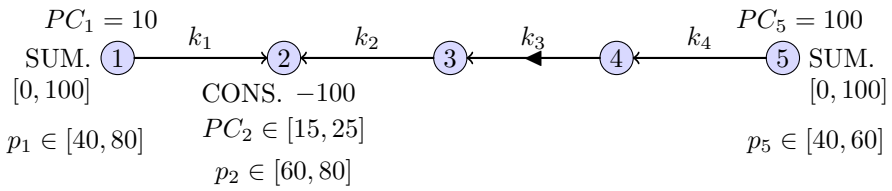
$$\underline{PC}_i - infraPCmax \leq PC_i \leq \overline{PC}_i \tag{3.7}$$

Sin embargo, estos dos cambios no son suficientes, ya que nos interesa que el poder calorífico esté comprendido entre las cotas dadas. Para lograr esto, debemos introducir un nuevo término en la función objetivo:

$$\mu^{infraPC} \sum_{i \in N} infraPC_i$$

siendo $\mu^{infraPC}$ un parámetro positivo que determinará el peso que le queremos asignar a la holgura total del poder calorífico. De esta forma tratamos de minimizar las holguras, consiguiendo así que se verifiquen las cotas reales del poder calorífico siempre que estas sean factibles.

Aplicando ahora el tándem correspondiente al apartado anterior (modelo básico con holgura de presión y básico con propagación de calidad) incluyendo la holgura para el poder calorífico, la herramienta encuentra solución factible para los 7 escenarios que estábamos probando.



Por ejemplo, en el caso del escenario 1, que como vimos anteriormente daba bastantes problemas, la solución factible encontrada es la que intuíamos, por el nodo 1 entran 88.78 unidades de flujo con una presión de 80 Pa y por el nodo 5 se suministran 11.22 unidades de flujo a una presión de 50.18 Pa. Además el poder calorífico en el nodo consumo es $PC_2 = 20.09 \in [15, 25]$.

Como podemos observar, la introducción de esta holgura al problema facilita la resolución del mismo, pero también sirve de ayuda para la identificación de posibles cuellos de botella, también denominados problemas de congestión del transporte, que imposibiliten la factibilidad de los escenarios. Se puede consultar Koch et al. (2015).

3.5. Resumen del modelo.

A lo largo de estos capítulos se presentaron las restricciones del modelo, explicando cada una de ellas, así como los términos que aparecerán en la función objetivo y el motivo por el cual nos interesa optimizarlos. A continuación se juntan todos estos elementos para poder presentar de una forma más sencilla el problema que modela la red. Es importante ver que estamos ante un problema de programación matemática mixto no lineal, MINLP (Mixed-Integer Non Linear Programming), multiobjetivo. Una forma de aproximar este problema multiobjetivo como un problema con una única función objetivo es asociarle a cada término que queramos optimizar un peso, quedando las expresiones mencionadas anteriormente, donde μ^c, μ^{pcv}, μ^{gc}, μ^{bo}, μ^{lp} y μ^{abs} son los pesos asociados a cada uno de ellos. De esta forma nuestro problema sería:

$$\text{Min } \mu^c \sum_{k \in E^c} \Delta_k^c + \mu^{pcv} \sum_{k \in E^{pcv}} \Delta_k^{pcv} + \mu^{gc} \sum_{k \in E^c} g_k + \mu^{bo} \sum_{p \in N^{rp}} bo_p + \mu^{lp} \sum_{k \in E^{st}} lp_k + \mu^{abs} \sum_{k \in E} (q_k^+ + q_k^-)$$

sujeto a:

Conservación del flujo:

$$c_i^{lb} \leq \sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k \leq c_i^{ub} \qquad \forall i \in N \qquad (1.3)$$

Conservación del flujo en energía:

$$24 \cdot 10^{-9} PC_i \left(\sum_{k \in E_i^{ini}} q_k - \sum_{k \in E_i^{fin}} q_k \right) = c_i^{ener} \qquad \forall i \in N \qquad (1.3)$$

Pérdida de presión:

$$u_i - u_j = \frac{16\lambda(q_k)L_k}{\pi^2 D_k^5} R\theta_{m_k}|q_k|q_k Z(\sqrt{u_{m_k}}, \theta_{m_k}) + \frac{2g}{R\theta_{m_k}} \frac{u_{m_k}}{Z(\sqrt{u_{m_k}}, \theta_{m_k})}(h_j - h_i) \qquad \forall k = (i,j) \in E^{st} \tag{1.4}$$

Aumento de la presión en un compresor:

$$u_i \leq u_j \qquad \forall k = (i,j) \in E^c \tag{1.5}$$

Compresores activos:

$$u_i - u_j \leq M_1(1 - y_k^{oc}) \qquad \forall k = (i,j) \in E^c \tag{1.6}$$
$$u_i - u_j \geq -M_2 y_k^{oc} \qquad \forall k = (i,j) \in E^c \tag{1.7}$$

A lo sumo un compresor activo en cada estación:

$$\sum_{k \in E^{cs}} y_k^{oc} \leq 1 \qquad \forall cs \in CS \tag{1.8}$$

Diagramas operativos:

$$y_k^{oc} S_{min} su \frac{\sqrt{u_i}}{Z(\sqrt{u_i}, \theta_i) R\theta_i} \leq \frac{q_k}{N_k^{ac}} \leq y_k^{oc} S_{max} st \frac{\sqrt{u_i}}{Z(\sqrt{u_i}, \theta_i) R\theta_i} \qquad \forall k = (i,j) \in E^c \tag{1.9}$$
$$y_k^{oc}(g_L(\sqrt{u_i}, \frac{q_k}{N_k^{ac}}))^2 - M(1 - y_k^{oc}) \leq \frac{u_j}{u_i} \leq y_k^{oc}(g_U(\sqrt{u_i}, \frac{q_k}{N_k^{ac}}))^2 + M(1 - y_k^{oc}) \ \forall k = (i,j) \in E^c \tag{1.10}$$

Óptimo de eficiencia isentrópica:

$$\eta_k^* = \eta\left(\frac{q_k}{N_k^{ac}}, p_i, p_j\right) = \max_{r \in \mathcal{R}} \eta\left(\frac{q_k}{r}, p_i, p_j\right) \qquad \forall k = (i,j) \in E^c \tag{1.26}$$

Válvulas de control de presión:

$$q_k \leq \overline{q}_k y_k^{pcv} \qquad \forall k = (i,j) \in E^{pcv} \tag{1.11}$$
$$q_k \geq \underline{q}_k (1 - y_k^{pcv}) \qquad \forall k = (i,j) \in E^{pcv} \tag{1.12}$$
$$u_i - u_j \leq M_1 y_k^{pcv} \qquad \forall k = (i,j) \in E^{pcv} \tag{1.13}$$
$$u_i - u_j \geq M_2(1 - y_k^{pcv}) \qquad \forall k = (i,j) \in E^{pcv} \tag{1.14}$$

Válvulas de apertura y cierre:

$$q_k \leq \overline{q}_k y_k^{ocv} \qquad \forall k=(i,j) \in E^{ocv} \qquad (1.15)$$
$$q_k \geq \underline{q}_k y_k^{ocv} \qquad \forall k=(i,j) \in E^{ocv} \qquad (1.16)$$
$$u_i - u_j \leq M_1(1 - y_k^{ocv}) \qquad \forall k=(i,j) \in E^{ocv} \qquad (1.17)$$
$$u_i - u_j \geq M_2(y_k^{ocv} - 1) \qquad \forall k=(i,j) \in E^{ocv} \qquad (1.18)$$

Válvulas de regulación de presión fija:

$$u_i - u_k^{frv} \leq M_1 y_k^{frv} \qquad \forall k=(i,j) \in E^{frv} \qquad (1.19)$$
$$u_i - u_k^{frv} \geq M_2(y_k^{frv} - 1) \qquad \forall k=(i,j) \in E^{frv} \qquad (1.20)$$
$$u_j - u_k^{frv} \leq M_1(1 - y_k^{frv}) \qquad \forall k=(i,j) \in E^{frv} \qquad (1.21)$$
$$u_j - u_k^{frv} \geq M_2(y_k^{frv} - 1) \qquad \forall k=(i,j) \in E^{frv} \qquad (1.22)$$
$$u_i - u_j \leq M_1 y_k^{frv} \qquad \forall k=(i,j) \in E^{frv} \qquad (1.23)$$
$$u_i - u_j \geq -M_2 y_k^{frv} \qquad \forall k=(i,j) \in E^{frv} \qquad (1.24)$$

Propagación calidad:

$$q_k = q_k^+ - q_k^- \qquad \forall k=(i,j) \in E \qquad (3.2)$$
$$PC_i \sum_{k \in E_i^{fin} \cup E_i^{ini}} q_k = \sum_{k=(j,i) \in E_i^{fin}} PC_j q_k^+ + \sum_{k=(i,j) \in E_i^{ini}} PC_j q_k^- \qquad \forall i \in N \qquad (3.3)$$
$$\underline{PC}_i - infraPC_i \leq PC_i \qquad \forall i \in N \qquad (3.6)$$

Boil-off:

$$Q_p = \sum_{k \in E_p^{ini}} q_k - \sum_{k \in E_p^{fin}} q_k \qquad \forall p \in N^{rp} \qquad (1.30)$$
$$Q_p \geq TD_s y_p^{bo} \qquad \forall p \in N^{rp} \qquad (1.31)$$
$$Q_p \leq c_p^{ub} y_p^{bo} \qquad \forall p \in N^{rp} \qquad (1.32)$$
$$r_p \geq 0 \qquad \forall p \in N^{rp} \qquad (1.33)$$
$$r_p \geq TD_c - \frac{TD_c}{O_s - TD_s}(Q_p - TD_s) \qquad \forall p \in N^{rp} \qquad (1.34)$$

Cotas de las variables:

$$\underline{u}_i \leq u_i \leq \overline{u}_i \qquad \forall i \in N \qquad (1.1)$$
$$\underline{q}_k \leq q_k \leq \overline{q}_k \qquad \forall k \in E \qquad (1.2)$$
$$0 \leq infraPC_i \leq infraPCmax \qquad \forall i \in N \qquad (3.5)$$
$$\underline{PC}_i - infraPCmax \leq PC_i \leq \overline{PC}_i \qquad \forall i \in N \qquad (3.7)$$

$$y_k^{oc} \in \{0,1\} \qquad \forall k \in E^c$$
$$y_k^{pcv} \in \{0,1\} \qquad \forall k \in E^{pcv}$$
$$y_k^{ocv} \in \{0,1\} \qquad \forall k \in E^{ocv}$$
$$y_k^{frv} \in \{0,1\} \qquad \forall k \in E^{frv}$$
$$y_p^{bo} \in \{0,1\} \qquad \forall p \in N^{rp}$$
$$Q_p \geq 0 \qquad \forall p \in N^{rp}$$
$$\eta_k^* \geq 0 \qquad \forall k \in E^c$$
$$q_k^+ \geq 0 \qquad \forall k \in E$$
$$q_k^- \geq 0 \qquad \forall k \in E$$
$$g_k \geq 0 \qquad \forall k \in E^c$$
$$bo_p \geq 0 \qquad \forall p \in N^{rp}$$
$$lp_k \geq 0 \qquad \forall k \in E^{st}$$
$$\Delta_k^c \geq 0 \qquad \forall k \in E^c$$
$$\Delta_k^{pcv} \geq 0 \qquad \forall k \in E^{pcv}$$
$$0 \leq N_k^{ac} \leq N^k,\ N_k^{ac} \in \mathbb{N} \qquad \forall k \in E^c$$

Capítulo 4

Implementación y resultados obtenidos.

Una vez calibrada la herramienta de optimización para el modelo con propagación de calidad, proceso que se acaba de explicar a lo largo del Capítulo 3, vamos a ejecutar un conjunto de diferentes configuraciones sobre una batería de escenarios compuesta por los siete escenarios de prueba representados en el Anexo A, un escenario de la red gallega y un escenario de la red española. Además, estos dos últimos escenarios serán modificados según el tipo de consumo (bajo, medio o alto) y según el poder calorífico relativo de los gases que se introducen en cada caso, obteniendo así un mayor número de instancias distintas, procedimiento explicado a lo largo de la Sección 4.1.

Por otro lado, en la Sección 4.2 presentaremos los resultados obtenidos al ejecutar las diferentes configuraciones y una discusión de los mismos.

Debemos mencionar que, tanto para la generación de escenarios como para la ejecución de las distintas configuraciones se crearon diferentes scripts en Python, incluidos en el Apéndice B. Una de las mayores dificultades en el desarrollo de los mismos residía en el procesado de la información asociada a las ejecuciones realizadas, ya que estamos ante problemas con un gran número de variables. Además, para la preparación de las visualizaciones de los resultados se llevó a cabo la creación de un script que almacenase la información necesaria y, posteriormente, se utilizó el entorno Jupyter Notebooks para la creación de las distintas gráficas.

4.1. Generación de escenarios y enfoques aplicados.

Como tenemos mencionado anteriormente, estamos considerando 9 escenarios, los 7 escenarios de prueba con los que calibramos la herramienta, un escenario de la red gallega y otro de la red española. Para el caso de los escenarios de la red gallega y española, distinguiremos 12 situaciones distintas para cada uno de ellos, por lo que en total tenemos una batería formada por 31 escenarios.

Como acabamos de explicar, en el caso de la red gallega y española se obtienen distintos escenarios. Para llevar a cabo la generación de los mismos vamos a considerar 3 tipos de consumo al 50 %, 60 % y 75 % del máximo de emisión. De esta forma obtendremos escenarios con un consumo bajo, medio y alto, respectivamente. A partir de cada uno de ellos, consideramos dos escenarios distintos según la composición del gas que entra en la red, teniendo así un total de 6 escenarios. Por último, obtenemos una solución factible para cada uno de ellos maximizando el consumo de gas en las estaciones de compresión y conservando el caudal en energía, sacando de esta forma cotas para el poder calorífico en los nodos de consumo, y creamos así a partir de cada escenario otros dos, uno donde las cotas del poder calorífico vienen determinadas por una horquilla relativamente pequeña (5 %), y otro con una horquilla más holgada (25 %). Vamos a representar este proceso con el ejemplo de la Figura 4.1:

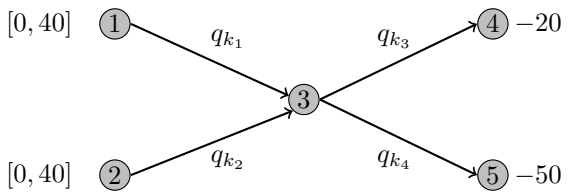


Figura 4.1: Ejemplo de una red formada por 5 nodos y 4 aristas.

Supongamos que tenemos una red compuesta por 5 nodos y 4 aristas, donde los nodos 1 y 2 son nodos suministro, ambos con una capacidad máxima de emisión de 40 unidades de flujo, y 4 y 5 nodos consumo con una demanda de 20 y 50 unidades de flujo, respectivamente. Además, fijemos el poder calorífico en los nodos de suministro, por ejemplo, $PC_1 = 30$ y $PC_2 = 50$. Si resolvemos este problema maximizando el gas consumido en las estaciones de compresión, nótese que nuestra red no tiene compresores, obtenemos como solución aquella en la que del nodo 1 salen 40 unidades de flujo y del nodo 2 30 unidades. Nótese que hay más soluciones factibles y óptimas a este problema, pero nos centraremos en esta.

Aplicando la regla de Bialek vista en el Capítulo 3, tendríamos que el poder calorífico del gas que llega a los nodos de consumo sería:

$$PC_4 = PC_5 = PC_3 = \frac{PC_1 \cdot q_{k_1} + PC_2 \cdot q_{k_2}}{q_{k_1} + q_{k_2}} = \frac{30 \cdot 40 + 50 \cdot 30}{70} = 38.57$$

De esta forma obtenemos el poder calorífico que correspondería a cada nodo de consumo a partir de la solución factible encontrada anteriormente. Ahora pasaremos a construir dos instancias distintas a partir de las cotas de poder calorífico. Para ello se determinan dos horquillas a partir del PC calculado anteriormente en cada nodo de demanda, una al 5 % y otra más holgada al 25 %, de forma que tendríamos como cotas de poder calorífico [36.64,40.49] y [28.93,48.21], respectivamente.

De esta forma, sabemos que para los escenarios correspondientes con las cotas anteriormente indicadas tendríamos como mínimo una solución factible, aquella que maximiza el gas consumido en las estaciones de compresión. Así, podemos comprobar si por los distintos métodos la herramienta es capaz de encontrar solución factible, sabiendo que existe como mínimo una, y si esta es mejor, ya que trataríamos de minimizar el gas consumido.

Una vez generados los diferentes escenarios, ejecutaremos sobre ellos las siguientes configuraciones resolviéndolos con Ipopt y Knitro ambos a través de AMPL y Pyomo y con el algoritmo 2-SLP explicado a lo largo del Capítulo 2:

- Método 1: Método iterativo a partir del modelo básico de forma que tanto el poder calorífico en cada uno de los nodos como el caudal en energía correspondiente, se calcularán por fuera del modelo en cada una de las iteraciones. Una vez calculado el poder calorífico, los consumos en masa se actualizan y se vuelve a resolver el modelo en una nueva iteración. Este método devuelve soluciones factibles para el modelo básico y el poder calorífico en cada nodo para dicha solución. Sin embargo, no nos garantiza que dicho poder calorífico esté comprendido entre las cotas especificadas ni que la entrega en masa conlleve la entrega en energía deseada.

- Método 2: Modelo básico con restricciones de propagación de calidad incluidas. En este caso no se están considerando las restricciones de conservación del flujo en energía. Este método devuelve soluciones para el modelo básico con propagación de calidad, es decir, en el caso de obtener una solución factible, el poder calorífico en cada nodo estará entre las cotas deseadas. Sin embargo, no se está garantizando que la entrega en energía sea la deseada, ya que se modela el problema con las restricciones de conservación de caudal en masa.

- Método 3: Modelo básico con las restricciones de propagación de calidad y de conservación del

flujo en energía incluidas. En este caso con el algoritmo 2-SLP se van a distinguir dos configuraciones distintas:

- Opción 1: Tanto las restricciones de propagación de calidad como las de conservación del flujo en energía están dentro del modelo.
- Opción 2: Las restricciones de propagación de calidad están dentro del modelo. Sin embargo, el flujo en energía se actualiza en cada iteración del algoritmo a partir del poder calorífico de cada nodo.

En este método, se obtienen soluciones de forma que el poder calorífico en cada nodo esté comprendido entre las cotas especificadas y que la energía que llega a cada nodo de consumo sea la correcta, pues estamos teniendo en cuenta las restricciones de conservación de flujo en energía.

- Método 4: Se llevan a cabo ejecuciones en tándem. Para ello consideramos las dos mejores configuraciones del método 2 y las dos mejores del método 3.

  - Modelo básico con las mejores configuraciones del método 2 → Modelo básico con propagación de calidad con las mejores configuraciones del método 2 → Modelo básico con propagación de calidad y conservación del flujo en energía con las mejores configuraciones del método 3.
  - Modelo básico con las mejores configuraciones del método 2 → Modelo básico con poder calorífico penalizado con las mejores configuraciones del método 2 → Modelo básico con propagación de calidad con las mejores configuraciones del método 2 → Modelo básico con propagación de calidad y conservación del flujo en energía con las mejores configuraciones del método 3.

  Al igual que en el método anterior, en este caso se obtienen soluciones de forma que la energía que llega a cada nodo consumo es la energía demandada y que el poder calorífico esté comprendido entre las cotas. Esto se debe a que la última etapa de cada uno de los tándems se corresponde con el método 3.

Además de estos métodos, también se ejecutó otro que consistía en un método iterativo a partir del modelo básico con las restricciones de propagación de calidad incluidas en él, de forma que iterativamente se calculase el caudal en energía a partir del poder calorífico de cada nodo. Sin embargo, dado que su rendimiento computacional era deficiente, este método no era comparable con los demás, por lo que omitiremos los resultados relativos al mismo.

Es importante mencionar que, en aquellos modelos en los que se incluya la restricción de propagación de calidad, estamos introduciendo también la holgura para la calidad del gas mencionada en el Capítulo 3.

4.1.1. Recálculos dentro de procedimientos iterativos.

Como acabamos de ver, en el método 1 teníamos que en cada iteración se iban actualizando los consumos en masa, al igual que al ejecutar el algoritmo 2-SLP, con el que también se actualizaban ciertas variables en cada iteración. A continuación, vamos a explicar más detalladamente este proceso.

Dada la complejidad del modelo, estos cálculos en procesos iterativos se hacen para facilitar la resolución de los subproblemas de cada iteración, haciendo alguna simplificación en los mismos. Por ejemplo, con el algoritmo 2-SLP algunos términos como el coeficiente de fricción o el factor de compresibilidad, dependientes de las variables del modelo tal y como mencionamos en el Capítulo 1, se considerarán constantes en cada iteración, de forma que se simplifica así la resolución de cada subproblema. Una vez finalizada una iteración se calcula el valor real de los mismos a partir de la solución obtenida en dicha solución, y este valor se pasa como parámetro en la siguiente iteración. En este caso,

estamos tomando como fijos algunos términos que tienen un impacto de segundo orden en la resolución del modelo.

Por otro lado, alguno de los métodos mencionados anteriormente, como el método 1 o la segunda opción del método 3, estarían aplicando este proceso a términos que tienen un mayor impacto dentro del modelo, ya que toman como constante el caudal de flujo en masa o energía, actualizándolos iterativamente según la solución obtenida en cada subproblema. Esto afectaría a términos de primer orden de nuestro modelo, como son las restricciones de conservación de flujo en masa o energía, respectivamente.

De esta forma vemos que a pesar de estar facilitando la resolución de nuestro problema lo alteran en menor o mayor orden, de forma que se pierde precisión en la resolución del modelo real.

4.2. Resultados.

A continuación vamos a presentar los resultados alcanzados al ejecutar cada uno de los métodos expuestos en la sección anterior. Es necesario recordar que nuestro objetivo es minimizar el gas consumido en las estaciones de compresión, por lo que representaremos en una gráfica dicho coste y en otra el coste total de la función objetivo de nuestro problema.

Por otra parte, el objetivo de este trabajo es validar y calibrar la herramienta GANESO™ al añadir al modelo las restricciones relacionadas con propagación de calidad de gas, por lo que se representará la violación de dichas restricciones en cada uno de los casos, así como la violación del resto de restricciones del modelo y la violación total.

Además, también representaremos el tiempo de ejecución para cada una de las configuraciones de cada método.

4.2.1. Método 1: cálculo iterativo del poder calorífico a partir del modelo básico.

Recordemos que en el primer método lanzábamos el modelo básico iterativamente sin tener en cuenta las restricciones de propagación de calidad, de forma que en cada iteración se calculaba el poder calorífico en cada nodo a partir de la solución obtenida en dicha iteración, y se actualizaban los consumos en masa. En el caso en que dicho método converja, es decir, los consumos en masa entre una iteración y la anterior no varíen, llegamos a una solución factible para el modelo básico.

Es importante ver que en este método no estamos asociando cotas al poder calorífico para cada nodo, por lo que este es libre. Además, dado que no estamos incluyendo las restricciones de propagación de calidad del gas en el modelo, en este caso solo representaremos las violaciones en el resto de restricciones. Por último, es necesario comentar que en algunos casos se tiene que el método no converge debido a que oscila entre dos soluciones.

Como se puede observar en la Figura 4.2, todas las configuraciones convergen en al menos uno de los escenarios simples, aunque tan solo dos son capaces de resolver algún escenario de la red gallega y otros dos alguno de la red española. Por otro lado, se puede ver que el algoritmo 2-SLP es el que más escenarios resuelve, siendo este incremento tanto a nivel de los escenarios simples como de la red gallega y española.

En las siguientes gráficas representaremos con puntos los escenarios para los cuales se ha encontrado una solución factible con cada una de las distintas configuraciones. Además se pueden observar unas líneas verticales algo más gruesas, estas líneas sirven para diferenciar los escenarios por dificultad, dividiendo así los escenarios simples creados por nosotros, los de la red gallega y los de la red española.

En la Figura 4.3 podemos ver el coste en las estaciones de compresión. Recordemos que para asociar unas cotas al poder calorífico en cada nodo resolvíamos los escenarios tratando de maximizar el gas consumido en las estaciones de compresión. El coste que se obtenía al resolver dicho modelo es lo que representamos por solución inicial. Así bien, se puede ver que entre las soluciones obtenidas, parece haber una predominancia del algoritmo 2-SLP, pues el coste es menor al resto de configuraciones.

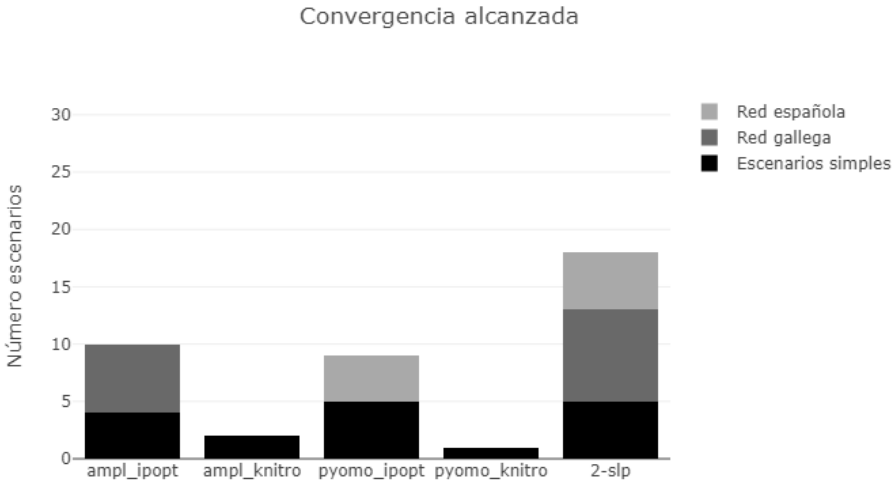


Figura 4.2: Convergencia del método iterativo.

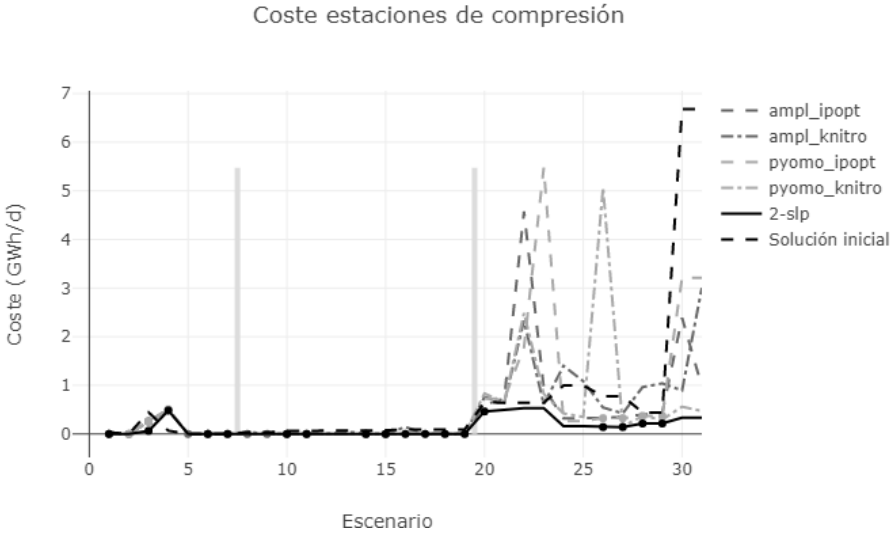


Figura 4.3: Coste en las estaciones de compresión.

Además exceptuando el escenario 4, en el resto de los escenarios el coste es menor que el de la solución inicial.

En relación a la Figura 4.4, donde se representa el valor total de la función objetivo, se vuelve a ver esa predominancia del algoritmo 2-SLP pues el método iterativo converge encontrando una solución con un menor coste total.

En la Figura 4.5 vemos la violación total de las restricciones del modelo para cada uno de los casos. En general, este resultado va a ser de una magnitud elevada debido a las unidades con las que trabajamos y seguirá una proporción directa con el tamaño del escenario, ya que cuanto más grande sea este, más restricciones tendremos y por lo tanto mayor será la violación total de las mismas. Para solucionar este problema y obtener así resultados más visuales, dividimos dicha violación total por

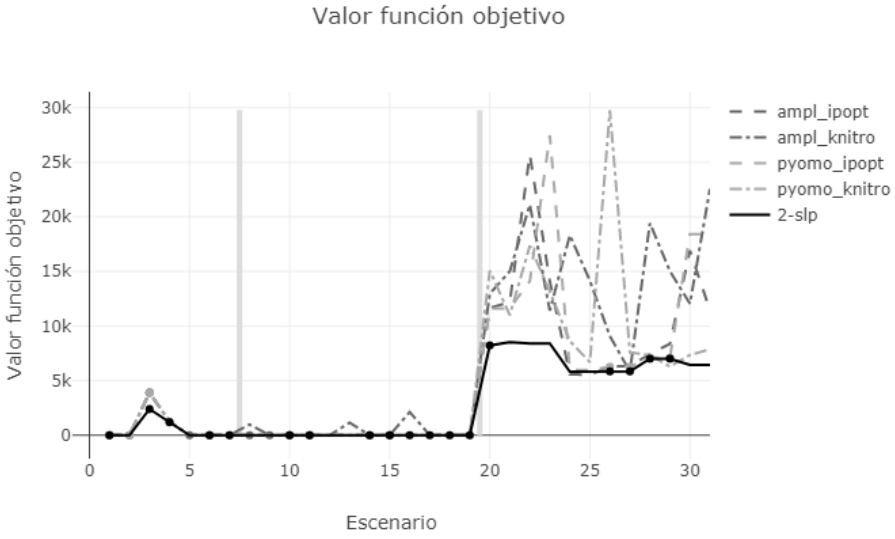


Figura 4.4: Coste total de la función objetivo.

Figura 4.5: Violación total de las restricciones del modelo.

el número de restricciones de ese escenario. Como se puede ver, tanto AMPL como Pyomo con los solvers Ipopt y Knitro siguen un patrón similar, mientras que la violación total de restricciones con el algoritmo 2-SLP es mucho menor, excepto en el escenario 5 donde observamos un pico. En este caso vemos que el escenario 5 no ha convergido con dicho algoritmo.

Por último, en la Figura 4.6 podemos observar los tiempos de ejecución en segundos para cada una de las configuraciones. Es fácil ver que tanto para los escenarios simples como para los de la red gallega los tiempos entre las distintas configuraciones son bastante similares. Sin embargo, para los escenarios de la red española vemos que el algoritmo 2-SLP es el que más tarda, siendo esta diferencia bastante grande en algunos escenarios, aunque menor en los escenarios en los que se alcanzaba convergencia.

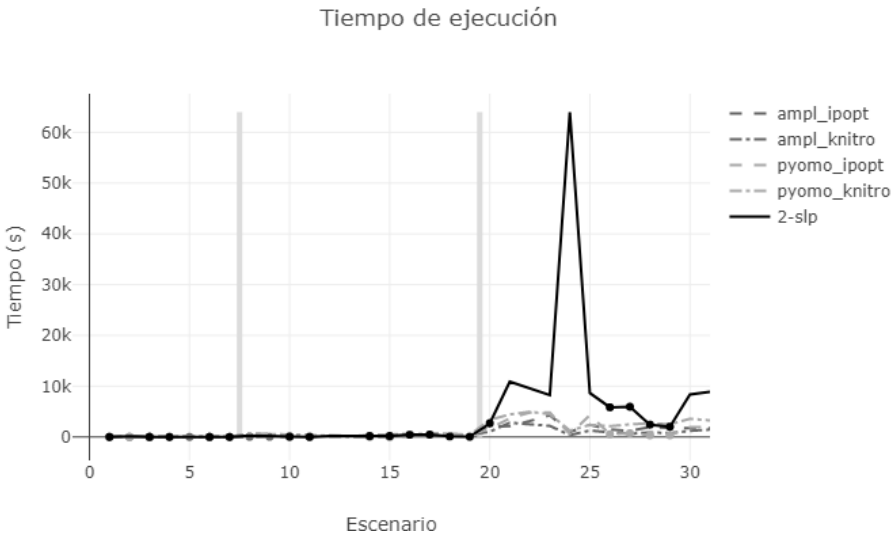


Figura 4.6: Tiempo de ejecución.

4.2.2. Método 2: modelo básico con propagación de calidad.

En este método lanzábamos el modelo básico con las restricciones de propagación de calidad incluidas en él y no se tenía en cuenta la conservación del flujo en energía. Al igual que con el método anterior, primero presentaremos un diagrama de barras apiladas donde se recoja la información de la factibilidad de los escenarios con cada una de las configuraciones y después presentaremos distintas gráficas que nos presenten más detalles sobre las soluciones encontradas.

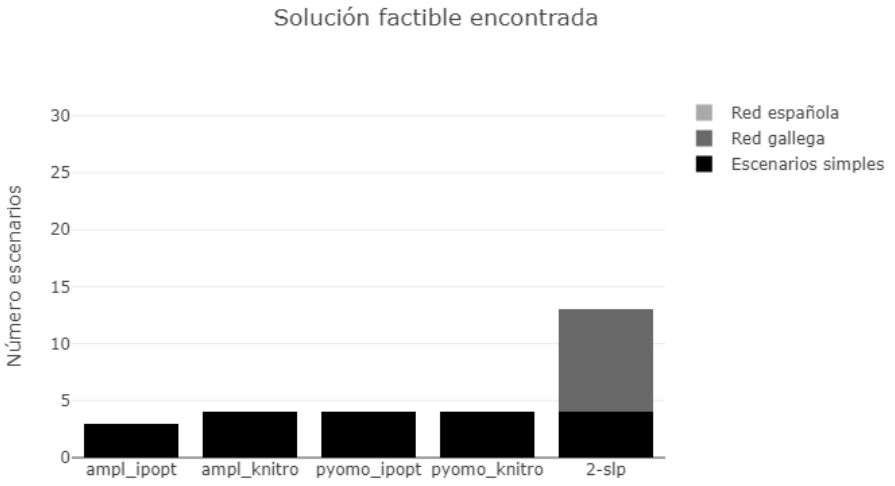


Figura 4.7: Soluciones factibles encontradas con el método 2.

Para empezar, en la Figura 4.7 vemos la cantidad de escenarios en los que las distintas configuraciones encontraron una solución factible. Como podemos ver, ninguna es capaz de encontrar solución factible para algún escenario de la red española, y únicamente el algoritmo 2-SLP es capaz de encon-

trar dicha solución para los escenarios de la red gallega. Además, el peor caso es el de Ipopt a través de AMPL ya que solo encuentra solución para 3 escenarios, mientras que el mismo solver a través de Pyomo la encuentra para 4 escenarios, todos ellas correspondientes a los escenarios simples que habíamos creado para calibrar la herramienta.

En las siguientes gráficas representaremos con puntos los escenarios en los que cada una de las distintas configuraciones encontraba una solución factible.

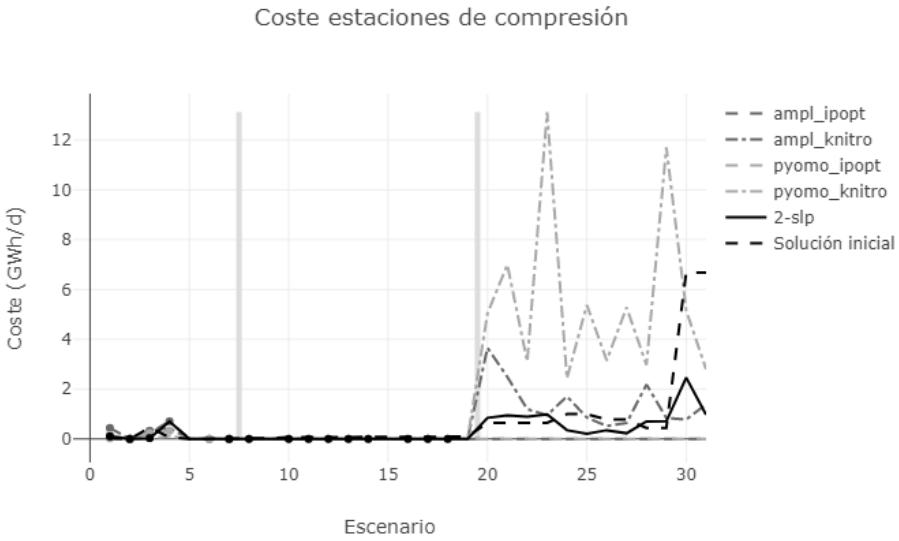


Figura 4.8: Coste en las estaciones de compresión.

En la Figura 4.8 podemos ver que el coste en las estaciones de compresión sigue aproximadamente el mismo comportamiento que el método anterior, ya que se puede ver que el algoritmo 2-SLP sigue encontrando soluciones factibles con un consumo de gas bajo en las estaciones de compresión.

Observando ahora el coste total en términos del valor que alcanza la función objetivo, véase la Figura 4.9, se vuelve a ver esa predominancia del algoritmo 2-SLP y de ambos solvers a través de AMPL. Además vemos que en general con Ipopt obtenemos un coste inferior que con Knitro, pues la función objetivo alcanza un valor menor.

En la Figura 4.10 vemos la violación total de las restricciones del modelo para cada uno de los casos. Al igual que en el método anterior esta gráfica está reescalada según el número de restricciones de cada escenario. Sin embargo, la escala sigue siendo muy elevada debido a las restricciones asociadas a la propagación de calidad del gas. Es importante mencionar que en la actual versión de GANESO™, posterior a las ejecuciones realizadas durante este trabajo, dentro del modelo se realiza un reescalado de las restricciones de propagación de calidad de forma que queden en una escala similar al resto. En ella podemos ver que el algoritmo 2-SLP sigue dominando, ya que la violación total de las restricciones es menor que para las otras configuraciones.

Como podemos observar en la Figura 4.11, la gráfica es similar a la anterior. Esto es debido a que la mayor parte de la violación en las restricciones ocurre en las correspondientes a la propagación de calidad del gas. Debemos recordar que cuando hablábamos de poder calorífico en un nodo, estábamos hablando de unidades por millón, por lo que la suma de las violaciones en estas restricciones puede ser muy elevada. Además en esta gráfica, vemos que los escenarios en los que encontrábamos una solución factible con los solvers Ipopt y Knitro, alcanzan una violación muy elevada en este tipo de restricciones, por lo que podríamos estar ante soluciones que en realidad no son factibles para nuestro modelo.

Observando ahora la Figura 4.12 vemos la violación que se produce en el resto de restricciones. En

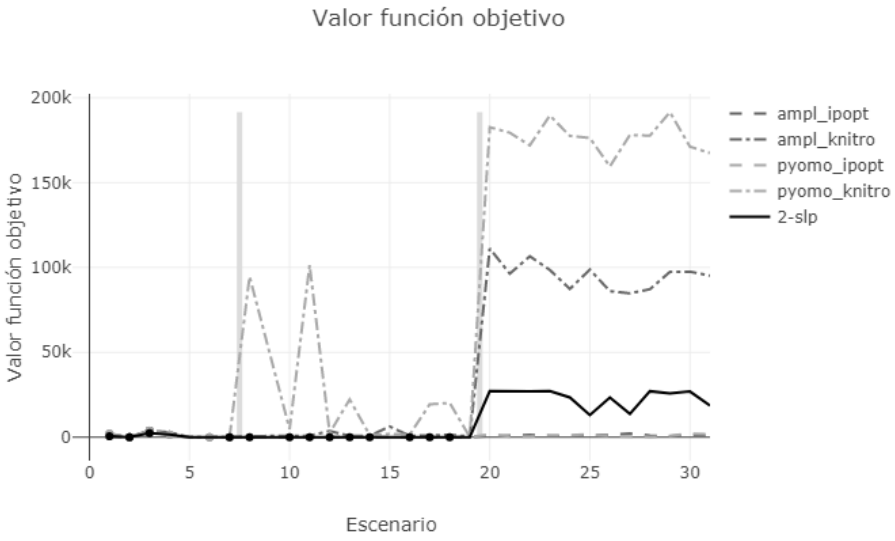


Figura 4.9: Coste total de la función objetivo.

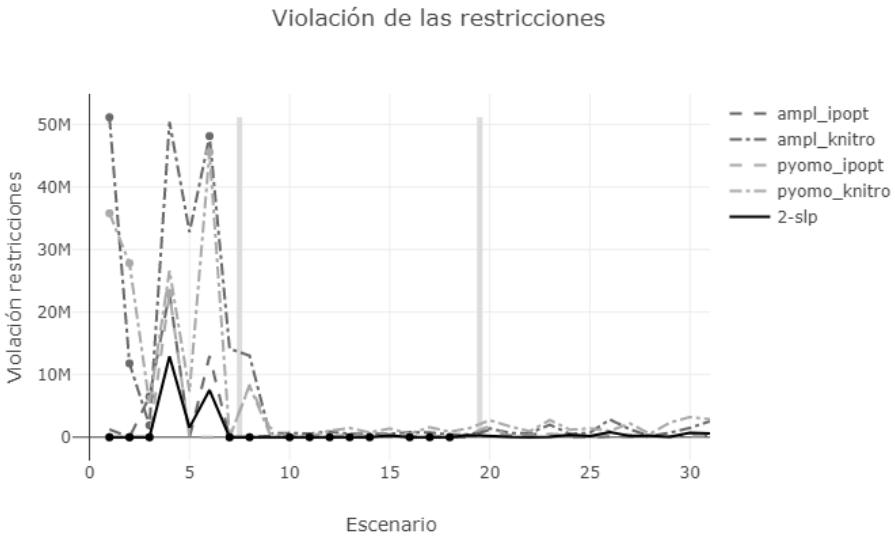


Figura 4.10: Violación total de las restricciones del modelo.

este caso tenemos que las violaciones son más pequeñas exceptuando dos casos, que se pueden observar en los picos más pronunciados de la gráfica. De nuevo, vemos que el algoritmo 2-SLP obtiene mejores soluciones, ya que la violación de las restricciones es menor.

Por último, en la Figura 4.13 podemos observar los tiempos de ejecución en segundos para cada una de las configuraciones. Es fácil ver que tanto para los escenarios simples como para los de la red gallega los tiempos entre las distintas configuraciones son bastante similares, aunque se puede observar que el algoritmo 2-SLP en los escenarios de la red gallega requiere un menor tiempo computacional. Esto se ve más claramente para los escenarios de la red española, pues el tiempo de dicho algoritmo es el más pequeño, seguido del solver Ipopt a través de AMPL.

Figura 4.11: Violación de las restricciones de propagación de calidad.

Figura 4.12: Violación de las restricciones excepto las de propagación de calidad.

4.2.3. Método 3: modelo básico con propagación de calidad y conservación del flujo en energía.

En este caso lanzábamos el modelo básico con las restricciones de propagación de calidad y de conservación de flujo en energía incluidas en el modelo. A diferencia de los métodos anteriores, es necesario recordar que en este método se ejecuta el algoritmo 2-SLP con dos configuraciones distintas, una en la que tanto la propagación de calidad como la conservación del flujo en energía están incluidas dentro del modelo, y otra en la que la propagación de calidad está incluida en el modelo pero el flujo en energía se calcula en cada iteración del algoritmo a partir del poder calorífico de cada nodo. De

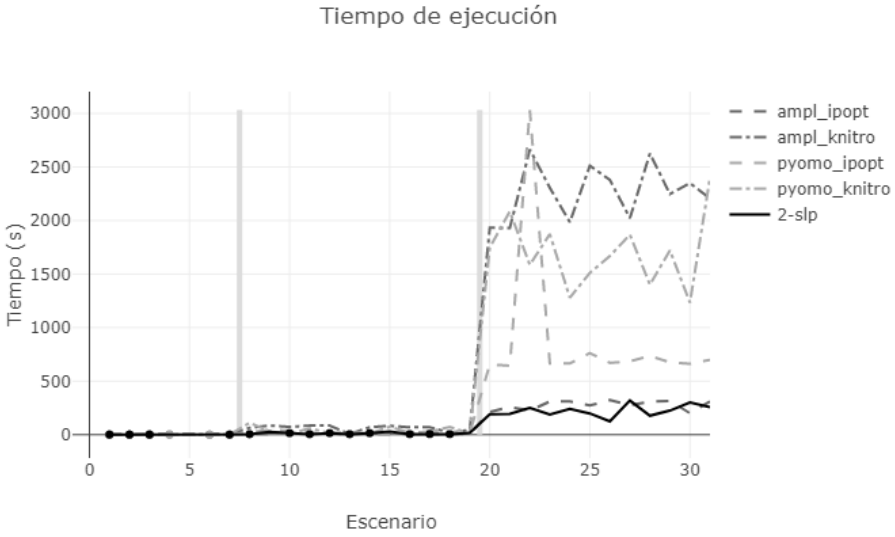


Figura 4.13: Tiempo de ejecución.

esta forma, en las gráficas denotaremos por 2-slp_1 y 2-slp_2 a los resultados de cada una de estas configuraciones, respectivamente.

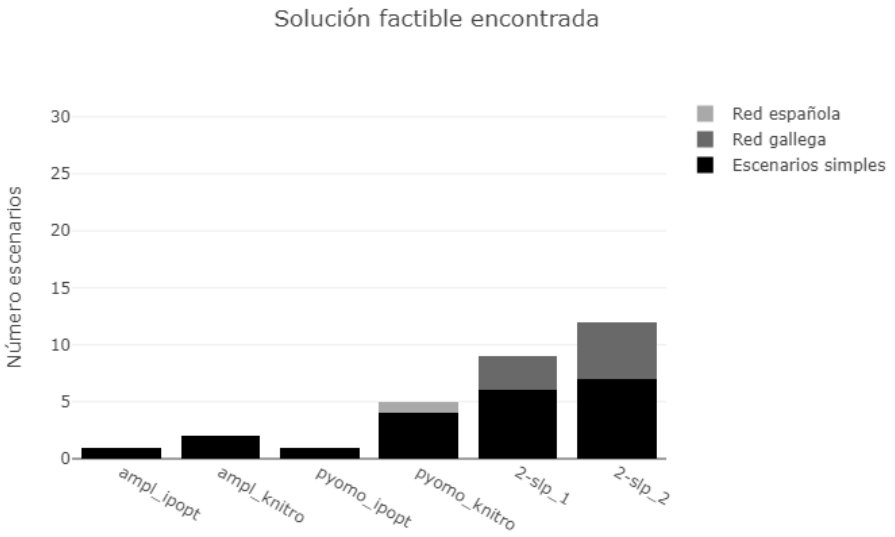


Figura 4.14: Soluciones factibles encontradas.

Para empezar, en la Figura 4.14 vemos la cantidad de escenarios en los que las distintas configuraciones encontraron una solución factible. Como podemos ver, solamente el solver Knitro a través de Pyomo es capaz de encontrar solución factible para un escenario de la red española, y al igual que el método anterior, únicamente el algoritmo 2-SLP es capaz de encontrar dicha solución para los escenarios de la red gallega, siendo más efectiva en ello la segunda configuración, en la que el flujo en energía se calculaba en cada iteración a partir del poder calorífico de cada nodo. Además, el peor caso

es el de Ipopt a través tanto de AMPL como de Pyomo ya que solo encuentra solución factible para 1 escenario.

Como podemos observar, en general en este método se encuentra solución factible para menos escenarios que en el caso anterior, por lo que vemos que si en vez de las restricciones de flujo en masa utilizamos las de flujo en energía, esto añade complejidad al problema. Este hecho no nos sorprende, ya que como habíamos comentado en el Capítulo 3, el flujo en energía depende del poder calorífico del gas que llega a cada nodo.

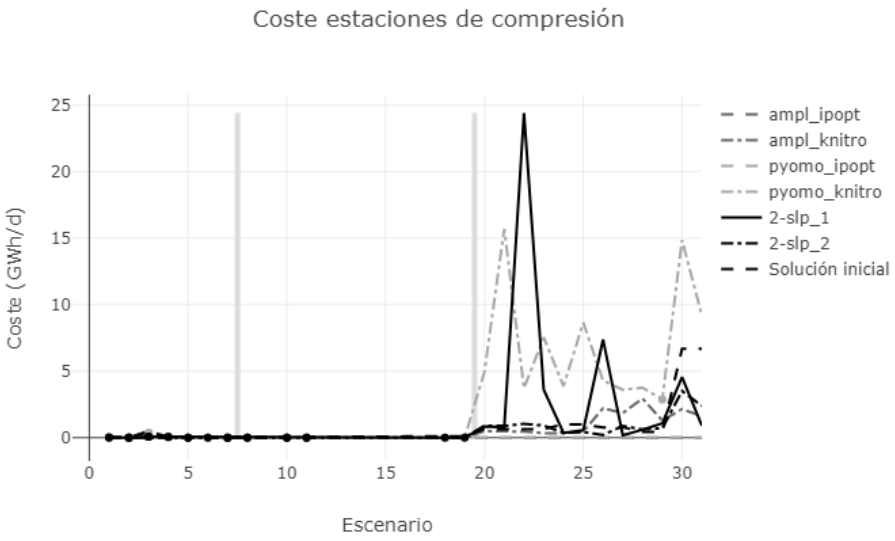


Figura 4.15: Coste en las estaciones de compresión.

En la Figura 4.15 podemos ver que el coste en las estaciones de compresión en los escenarios simples y de la red gallega es muy pequeño con cualquiera de las configuraciones, sin embargo, en los escenarios de la red española se ven más fluctuaciones, especialmente con el algoritmo 2-SLP y la primera configuración, en la que las restricciones de conservación de flujo en energía estaban dentro del modelo, y con el solver Knitro a través de Pyomo.

Observando ahora el coste total en términos del valor que alcanza la función objetivo en la Figura 4.16, se ve que en general los dos solvers a través de AMPL y el algoritmo 2-SLP con la segunda configuración son los que alcanzan los costes más bajos en la función objetivo. Además vemos que en general, al igual que en el método anterior, con Ipopt obtenemos un coste inferior que con Knitro, pues la función objetivo alcanza un valor menor.

En la Figura 4.17 vemos la violación total de las restricciones del modelo para cada uno de los casos. Recordemos que esta gráfica está reescalada según el número de restricciones de cada escenario. En ella podemos ver que el algoritmo 2-SLP sigue dominando en los escenarios en los que encuentra una solución factible.

Como podemos observar en la Figura 4.18, la gráfica es similar a la anterior. Recordemos que esto era debido a la magnitud de las unidades de las restricciones de propagación de calidad del gas.

Observando ahora la Figura 4.19 vemos que la violación que se produce en el resto de restricciones es mayor con los solvers Ipopt y Knitro a través de AMPL y Pyomo. De nuevo vemos que el algoritmo 2-SLP, con las dos configuraciones, obtiene mejores soluciones, ya que la violación de las restricciones es menor.

Por último, observando los tiempos de ejecución representados en la Figura 4.20 vemos que el algoritmo 2-SLP con la segunda configuración, en la que se calculaba el flujo en energía a partir del

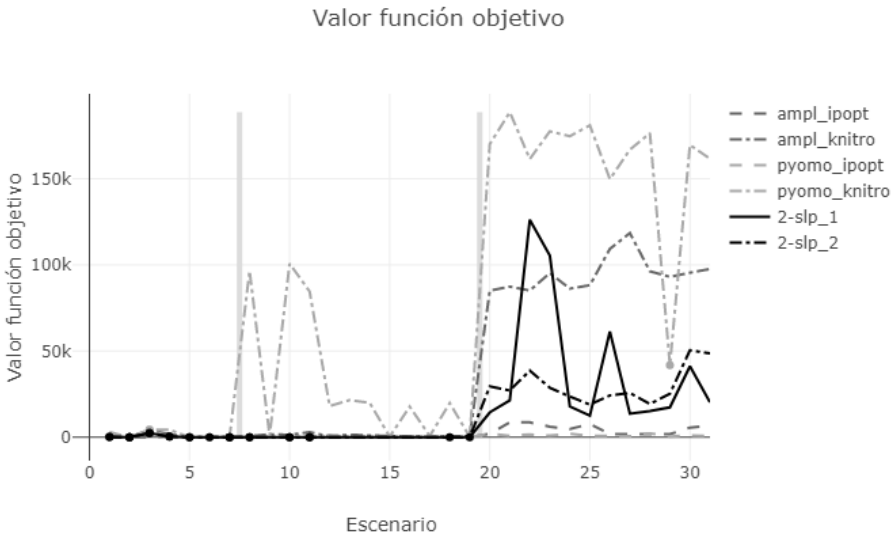


Figura 4.16: Coste total de la función objetivo.

Figura 4.17: Violación total de las restricciones del modelo.

poder calorífico en cada iteración, es el que menos tiempo tarda en la ejecución de los escenarios, en especial los de la red española, donde observamos una mayor diferencia entre las distintas ejecuciones.

4.2.4. Método 4: ejecución en tándem.

Como mencionamos en la Sección 4.1, para realizar las ejecuciones en tándem considerábamos las dos mejores configuraciones de los métodos 2 y 3. Para ello, vamos a escoger aquellas configuraciones que encuentran una solución factible para un mayor número de escenarios.

En el caso del segundo método, claramente el algoritmo 2-SLP es el que más soluciones factibles

Figura 4.18: Violación de las restricciones de propagación de calidad.

Figura 4.19: Violación de las restricciones excepto las de propagación de calidad.

encuentra. Sin embargo, tanto el solver Ipopt a través de Pyomo como el solver Knitro a través tanto de AMPL como de Pyomo encuentran solución factible para 4 escenarios. En este caso, con Ipopt a través de Pyomo obtenemos en general un menor coste de la función objetivo y un menor tiempo de ejecución, por lo que nos quedamos con esta configuración.

En el caso del tercer método claramente el algoritmo 2-SLP con las dos configuraciones distintas son los que encuentran solución factible para un mayor número de escenarios, por lo que consideraremos estas dos opciones.

Por otro lado es necesario recordar que se van a considerar dos tándems distintos, de forma que lanzaremos un total de 4 configuraciones distintas para cada uno de ellos.

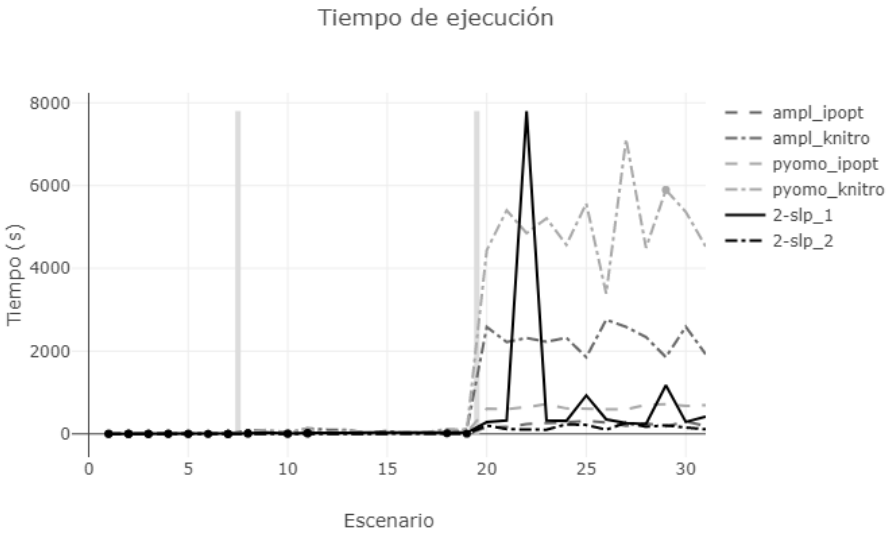


Figura 4.20: Tiempo de ejecución.

A continuación presentaremos los resultados obtenidos para el primer tándem, siendo este el del modelo básico con la configuración del método 2, modelo básico con propagación de calidad con la configuración del método 2 y modelo básico con propagación de calidad y conservación de flujo en energía con la configuración del método 3.

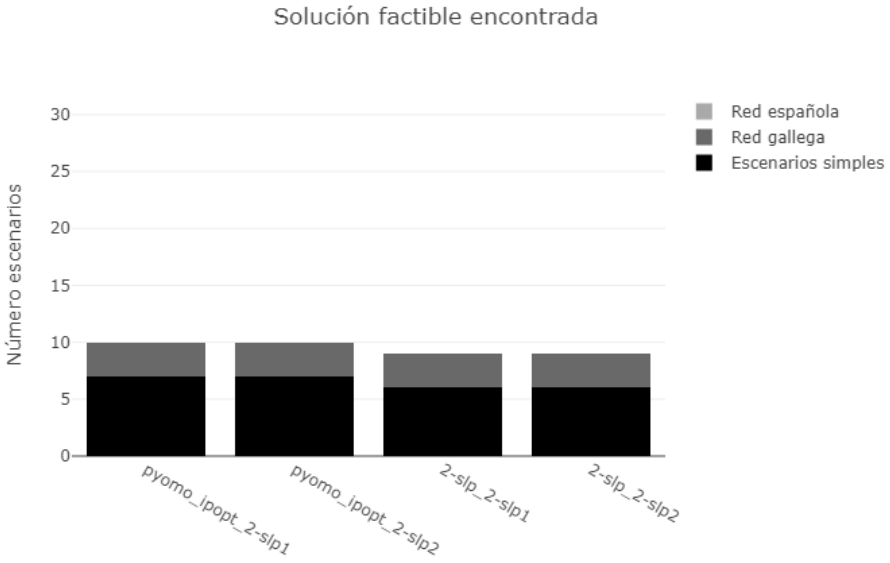


Figura 4.21: Soluciones factibles encontradas.

Para empezar, en la Figura 4.21 vemos la cantidad de escenarios en los que el tándem con las distintas configuraciones encontró una solución factible. Como podemos ver, las cuatro configuraciones encuentran solución factible en un número similar de escenarios, entre 9 y 10, pero ninguna es capaz

de encontrar una solución factible para algún escenario de la red española.

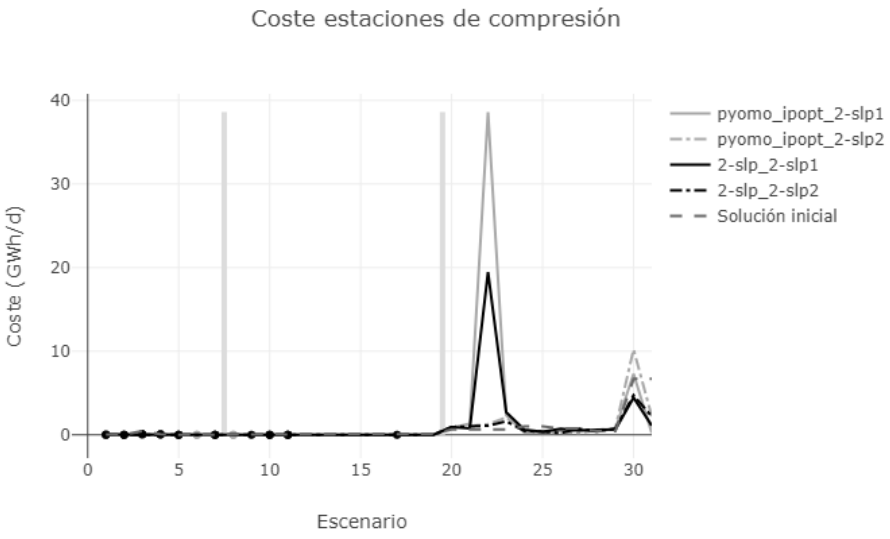


Figura 4.22: Coste en las estaciones de compresión.

En la Figura 4.22 podemos ver que el coste en las estaciones de compresión en los escenarios simples y de la red gallega es muy pequeño con cualquiera de las configuraciones, sin embargo, en los escenarios de la red española vemos que hay dos escenarios en los que el coste es muy elevado en los tándems donde el modelo con propagación de calidad y conservación de flujo en energía se resuleven con la primera configuración del algoritmo 2-SLP.

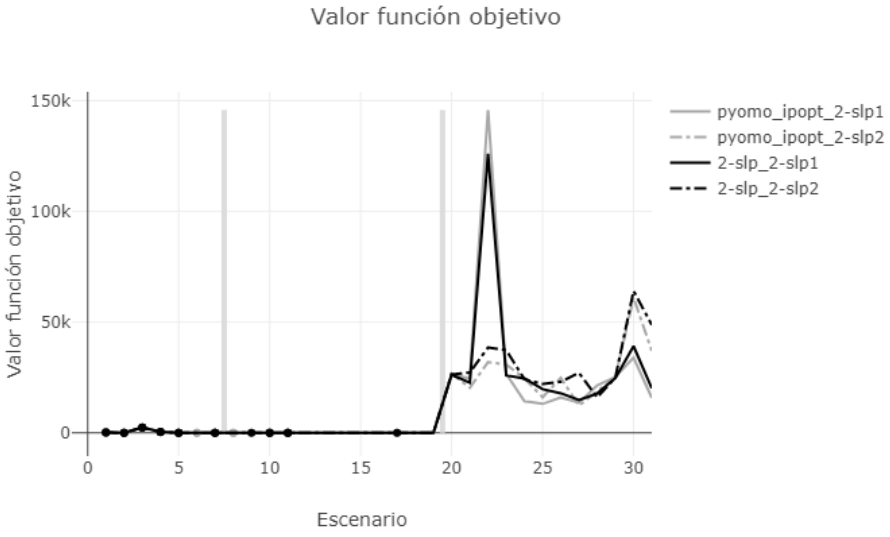


Figura 4.23: Coste total de la función objetivo.

Observando ahora el coste total en términos del valor que alcanza la función objetivo en la Figura 4.23, se ve que en general se sigue el mismo patrón que en la Figura anterior. Además los costes de la función objetivo en los escenarios simples y de la red gallega son muy pequeños en comparación a los de la red española, en donde destacan de nuevo dos escenarios, pero en especial el escenario 22.

Figura 4.24: Violación total de las restricciones del modelo.

En la Figura 4.24 vemos la violación total de las restricciones del modelo para cada uno de los casos. Recordemos que esta gráfica está reescalada según el número de restricciones de cada escenario. En ella podemos ver que los tándems en los que el modelo completo se resuelve con la segunda configuración del algoritmo 2-SLP tienen una menor violación de las restricciones. Además, como vimos en las gráficas anteriores, estos se correspondían con los de menor coste en la función objetivo y, en particular, menor gas consumido en las estaciones de compresión.

Como podemos observar en la Figura 4.25, la gráfica es similar a la anterior. Esto se debe, tal y como explicamos anteriormente, a que la mayor violación se produce en las restricciones de propagación de calidad, siendo la magnitud de estas muy elevada.

Observando ahora la Figura 4.26 vemos que la violación que se produce en el resto de restricciones es prácticamente nula exceptuando el escenario 22. De nuevo vemos que la mayor violación se produce cuando se resuelve el modelo completo con la primera configuración del algoritmo 2-SLP, en especial el tándem realizado con este algoritmo.

Por último, si observamos la Figura 4.27 podemos ver que claramente el tándem ejecutado con el algoritmo 2-SLP y con la segunda configuración de este para el modelo completo es el que obtiene mejores tiempos computacionales.

Una vez presentados los resultados para el primer tándem, vamos a proceder con los resultados relativos al segundo. Recordemos que en este caso lo que hacemos es añadir una etapa intermedia al tándem anterior donde las restricciones de propagación de calidad del modelo están penalizadas, proceso explicado en el Capítulo 3. Además, este modelo penalizado también se resolverá con las mejores configuraciones del método 2.

Para empezar, en la Figura 4.28 vemos la cantidad de escenarios en los que este tándem encontró una solución factible. Como podemos ver, en este caso todas las configuraciones encontraron solución factible para el mismo número de escenarios. En concreto los casos en los que se ejecuta Ipopt a través de Pyomo en las primeras etapas resuelven todos los escenarios simples y dos de la red gallega, mientras

Figura 4.25: Violación de las restricciones de propagación de calidad.

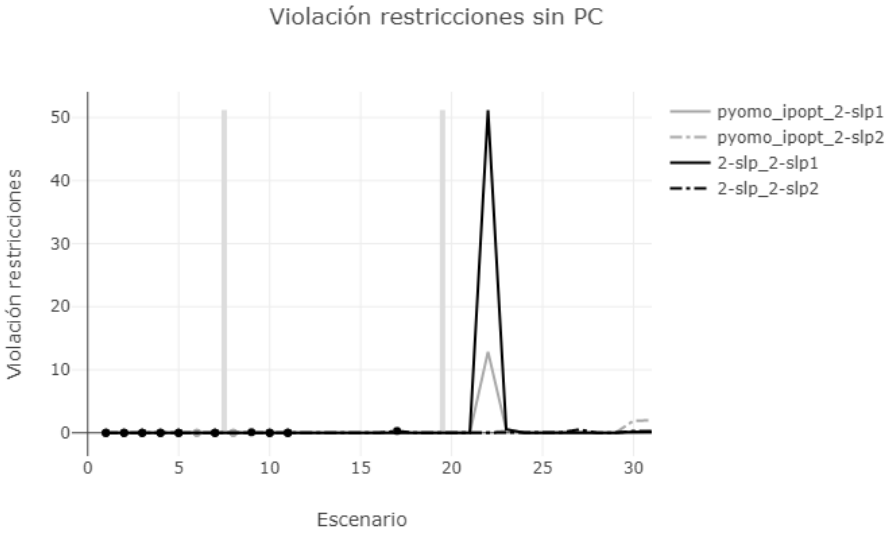


Figura 4.26: Violación de las restricciones excepto las de propagación de calidad.

que los casos en los que se lanza el algoritmo 2-SLP en todas las etapas resuelven un escenario más de la red gallega, aunque no son capaces de encontrar solución factible para el escenario 6.

En la Figura 4.29 podemos ver que al igual que en el tándem anterior, el gas consumido en las estaciones de compresión en los escenarios simples y de la red gallega es muy pequeño con cualquiera de las configuraciones, mientras que en los escenarios de la red española vemos que hay dos escenarios en los que el coste es muy elevado en los tándems donde el modelo completo se resuelve con la primera configuración del algoritmo 2-SLP. Sin embargo vemos que al añadir una etapa con la propagación de

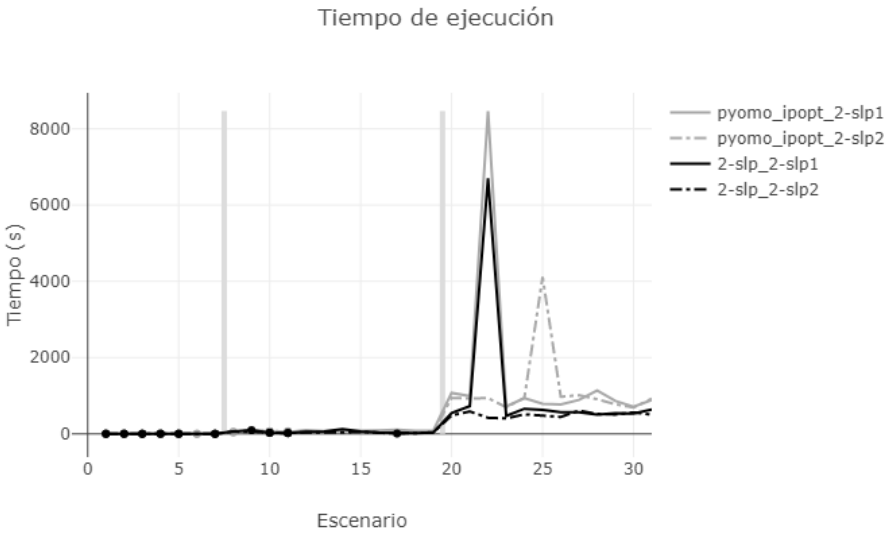


Figura 4.27: Tiempo de ejecución.

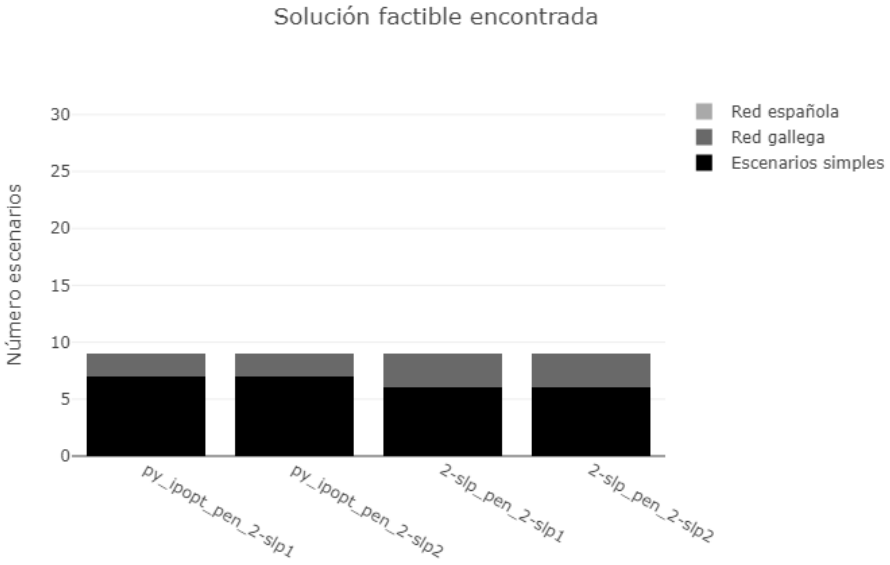


Figura 4.28: Soluciones factibles encontradas.

calidad penalizada, en estos picos se reduce el coste.

Observando ahora el coste total en términos del valor que alcanza la función objetivo en la Figura 4.30, se ve que en general se sigue el mismo patrón que en la Figura anterior. Además los costes de la función objetivo en los escenarios simples y de la red gallega son muy pequeños en comparación a los de la red española, en donde destacan de nuevo dos escenarios, pero en especial el escenario 22. La diferencia con el tándem anterior en este escenario es que en este caso la configuración con Pyomo y la primera opción del algoritmo 2-SLP tiene un coste total mucho mayor, a pesar de que el coste en

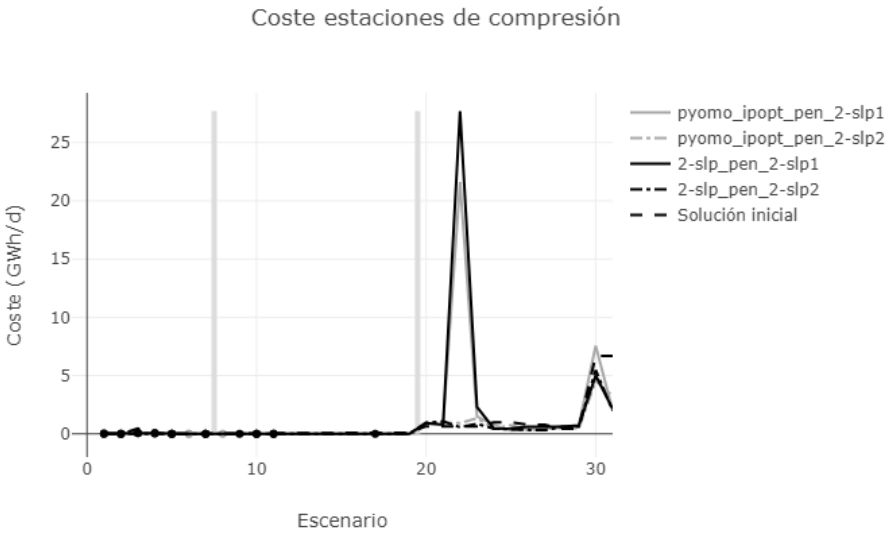


Figura 4.29: Coste en las estaciones de compresión.

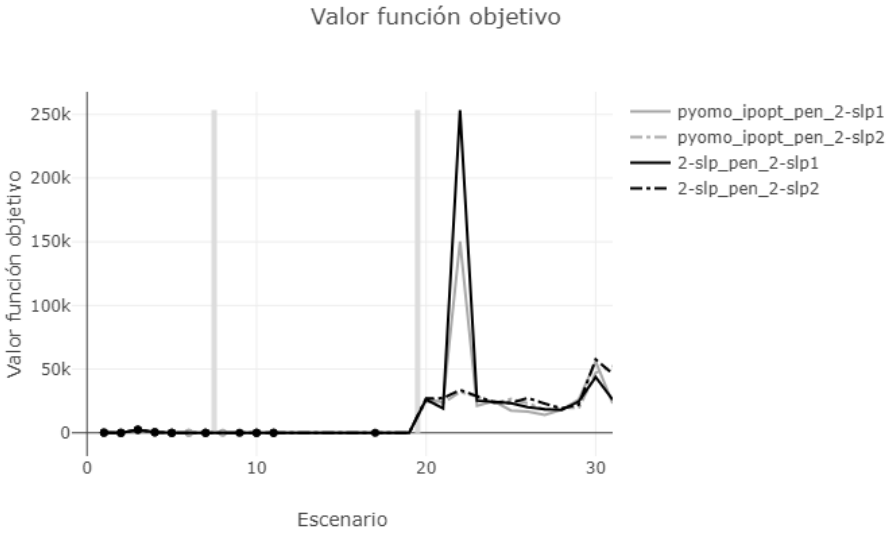


Figura 4.30: Coste total de la función objetivo.

las estaciones de compresión era menor.

En la Figura 4.31 vemos la violación total de las restricciones del modelo para cada uno de los casos. Recordemos que esta gráfica está reescalada según el número de restricciones de cada escenario. En ella podemos ver que los tándems en los que el modelo completo se resuelve con la segunda configuración del algoritmo 2-SLP tienen una menor violación de las restricciones. Además, como vimos en las gráficas anteriores, estos se correspondían con los de menor coste en la función objetivo y, en particular, menor gas consumido en las estaciones de compresión. Específicamente, se tiene que la mayor violación se

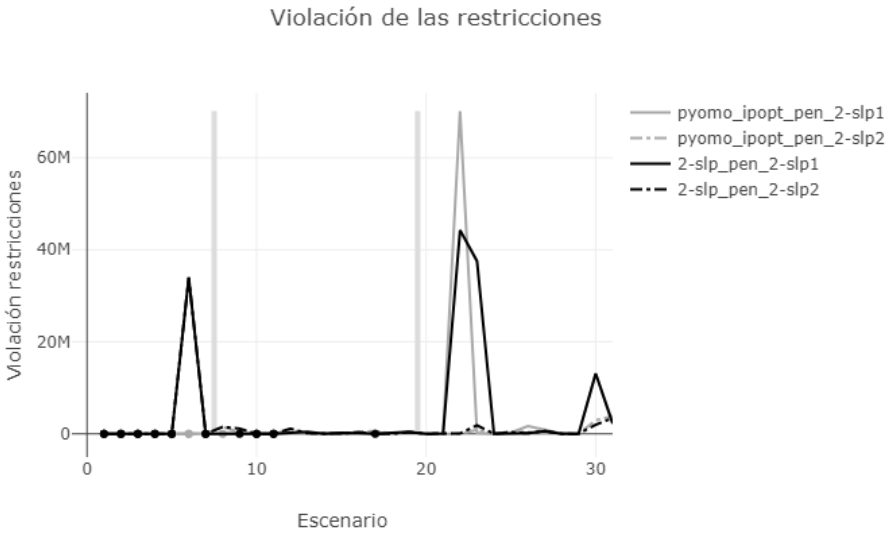


Figura 4.31: Violación total de las restricciones del modelo.

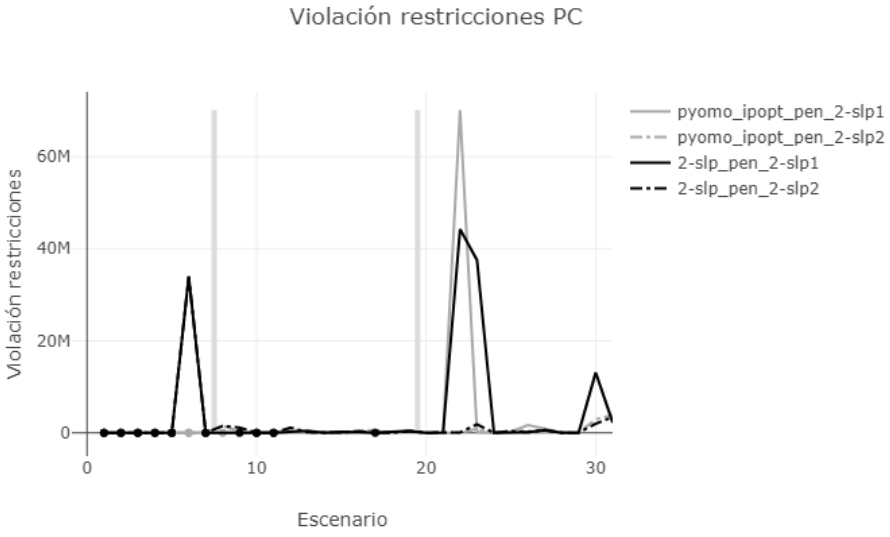


Figura 4.32: Violación de las restricciones de propagación de calidad.

produce en las restricciones de propagación de calidad (Figura 4.32), donde la gráfica es similar a la anterior.

Observando ahora la Figura 4.33 vemos que la violación que se produce en el resto de restricciones es prácticamente nula en los escenarios simples y de la red gallega, mientras que en el caso de la red española son bastante pequeñas, exceptuando el escenario 22. Además, vemos que hay casos en los que la violación de restricciones de los tándems con la primera configuración del 2-SLP son menores que las correspondientes a los tándems con la segunda configuración.

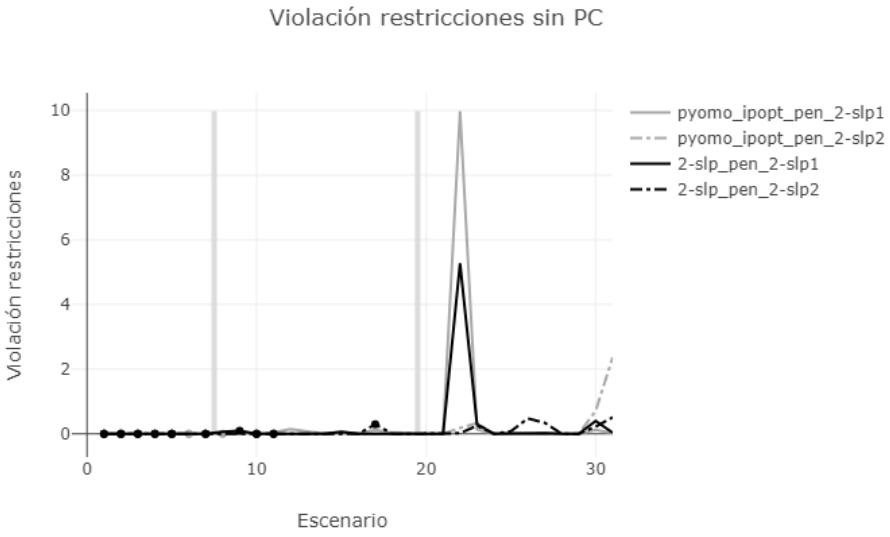


Figura 4.33: Violación de las restricciones excepto las de propagación de calidad.

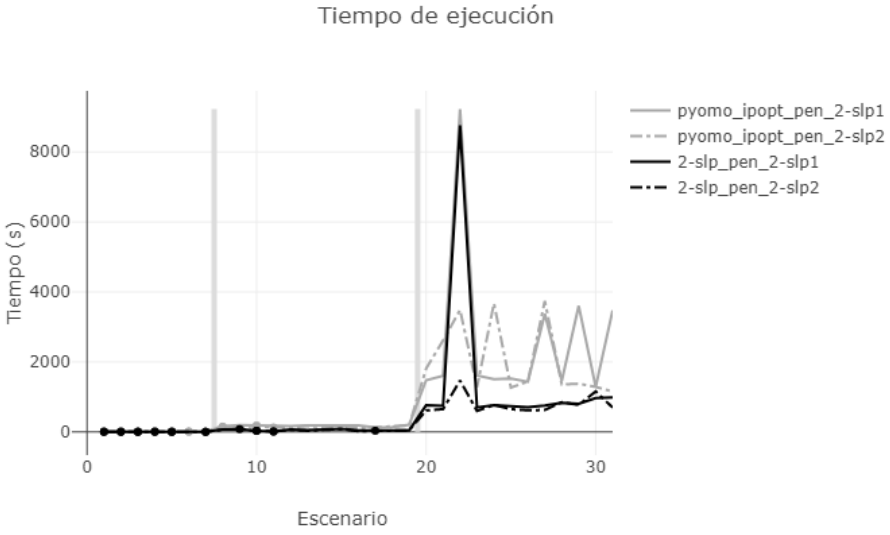


Figura 4.34: Tiempo de ejecución.

Finalmente en lo relacionado a los tiempos de ejecución (Figura 4.34), vemos que en los escenarios de la red gallega se ven tiempos más pequeños para el caso de los tándems con el algoritmo 2-SLP en todas las etapas. Este comportamiento se observa también en los escenarios de la red española, exceptuando el escenario 22 para el caso en el que la última etapa del tándem se resuelve con la primera configuración del 2-SLP, recordemos que en esta configuración tanto las restricciones de propagación de calidad como las de conservación de flujo en energía están incluidas en el modelo.

4.3. Comparativa de los distintos métodos.

Una vez vistos los resultados para cada uno de los distintos métodos en la Sección 4.2, y teniendo en cuenta que el método 3 y las dos opciones del método 4 resuelven el mismo modelo, estos son comparables. Por lo tanto a continuación vamos a presentar una comparativa de los mismos a partir de la mejor configuración para cada uno de ellos.

Para el caso del método 3, teníamos que la configuración que encontraba solución factible para un mayor número de escenarios era el 2-SLP con propagación de calidad incluida en el modelo y que en cada iteración calculaba el caudal en energía a partir del poder calorífico en cada nodo. Además, veíamos que era el que tenía un mejor rendimiento computacional, ya que el tiempo de ejecución era menor que para el resto de configuraciones. Por último, es necesario comentar que la violación de las restricciones con esta configuración era en general pequeña en comparación al resto de configuraciones.

Por otro lado, para la primera opción del método 4, teníamos que los tándems con Ipopt a través de Pyomo y la última etapa con el algoritmo 2-SLP resolvían un escenario más que en el caso de todas las etapas con este último algoritmo. Más concretamente, nos quedaremos con el tándem cuya última etapa se resuelve con la segunda configuración del 2-SLP, ya que aunque en la mayoría de casos ambos tienen un tiempo de ejecución, un coste y una violación de las restricciones similares, en los casos en los que difieren, los resultados son mejores con la segunda configuración.

Por último en la segunda opción del método 4, basada en el tándem anterior pero con una etapa intermedia añadida a mayores en la que se resolvía el modelo básico con las restricciones de propagación de calidad penalizadas, teníamos que las 4 configuraciones encontraban solución factible para el mismo número de escenarios. En vista del coste tanto en las estaciones de compresión como el coste total de la función objetivo y de la violación de las restricciones, veíamos que las mejores configuraciones son aquellas que resuelven todas las etapas del tándem con el algoritmo 2-SLP, y visto que los tiempos de ejecución son menores para aquella que tiene en la última etapa la segunda configuración del 2-SLP, decidimos quedarnos con esta.

Una vez determinada la mejor configuración para cada uno de los métodos, podemos llevar a cabo la comparación de las mismas. Comenzando con el número de escenarios para los que cada una encuentra solución factible, tenemos tal y como se puede observar en la Figura 4.35, que el algoritmo 2-SLP lanzado sobre el modelo completo encuentra solución factible en más escenarios que en el caso de los tándems. Esto se puede deber a que la solución proporcionada por alguna etapa del tándem no sea factible y por lo tanto al algoritmo 2-SLP de la última etapa del mismo le resulte más complicado llegar a la solución factible que encuentra en el caso del método 2, donde solamente resuelve el modelo completo.

Observando ahora la Figura 4.36, vemos que el coste en las estaciones de compresión está en general por debajo del correspondiente a la solución inicial, exceptuando algunos escenarios resueltos con algunas configuraciones, en especial el tándem con ipopt y 2-SLP. Además, se ve que tanto el tándem con las restricciones de propagación de calidad penalizadas como el modelo completo resuelto con la segunda configuración del algoritmo 2-SLP son las configuraciones que mejor resultados obtienen en referencia al gas consumido en las estaciones de compresión.

Si observamos ahora el valor total de la función objetivo (Figura 4.37), observamos que en la mayoría de escenarios de la red española el tándem resuelto con Ipopt y 2-SLP tiene un coste total menor que el resto de configuraciones, a diferencia de lo que vimos en la Figura 4.36.

Si ahora vemos la violación total en las restricciones del modelo en la Figura 4.38, vemos que exceptuando el escenario 6 donde el tándem con 2-SLP muestra un pico, en el resto de escenarios la violación de las restricciones es bastante similar entre las configuraciones de los distintos métodos. Esto se debe principalmente a que en el caso de los tándems la última etapa se resuelve con el algoritmo 2-SLP y como habíamos visto a lo largo del Capítulo 2, el primer paso de este era el algoritmo SLP-NTR que no es sensible a soluciones iniciales, ya que al no tener una región de confianza, las soluciones de cada iteración pueden estar muy distantes ya que no hay un control sobre esta distancia. Una posible solución para esto sería lanzar los tándems con Ipopt, Knitro o directamente el algoritmo PSLP, ya

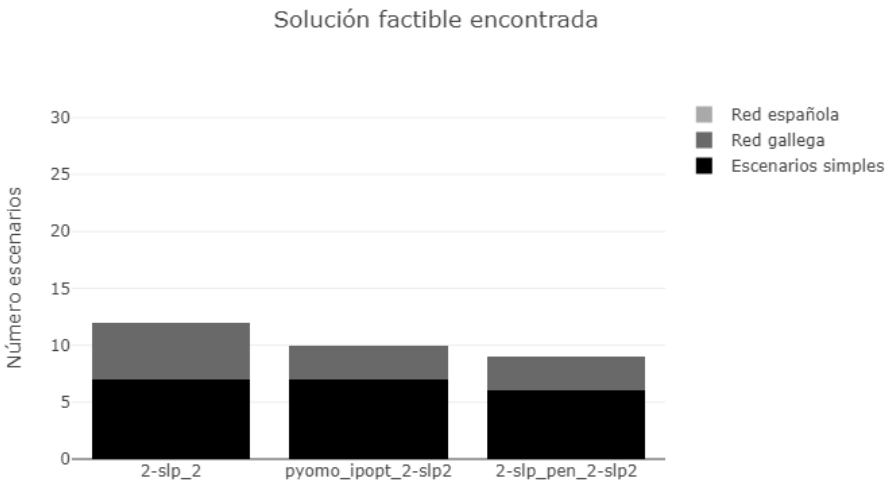


Figura 4.35: Soluciones factibles encontradas.

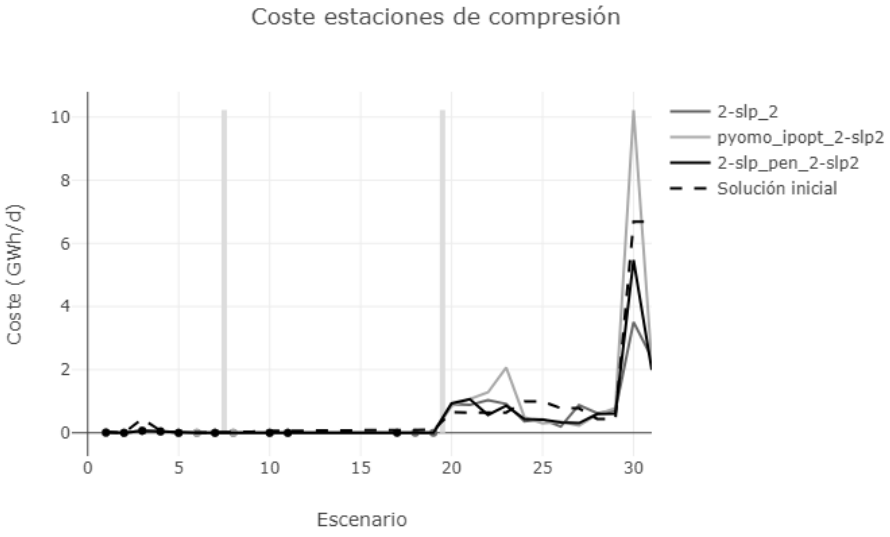


Figura 4.36: Coste en las estaciones de compresión.

que estos son sensibles a soluciones iniciales.

Por otro lado, si observamos la Figura 4.40 vemos que en las restricciones que no corresponden a la propagación de calidad, las violaciones varían más de un método a otro. Por ejemplo, se puede ver que el tándem Ipopt y 2-SLP obtiene soluciones con violaciones muy pequeñas, excepto en los dos últimos escenarios, en donde se incrementa algo más del doble que con el resto de métodos. Por otra parte, se ve que en general el tándem con propagación de calidad penalizado y resuelto con el algoritmo 2-SLP obtiene violaciones menores que el modelo completo resuelto con el mismo algoritmo.

Figura 4.37: Coste total de la función objetivo.

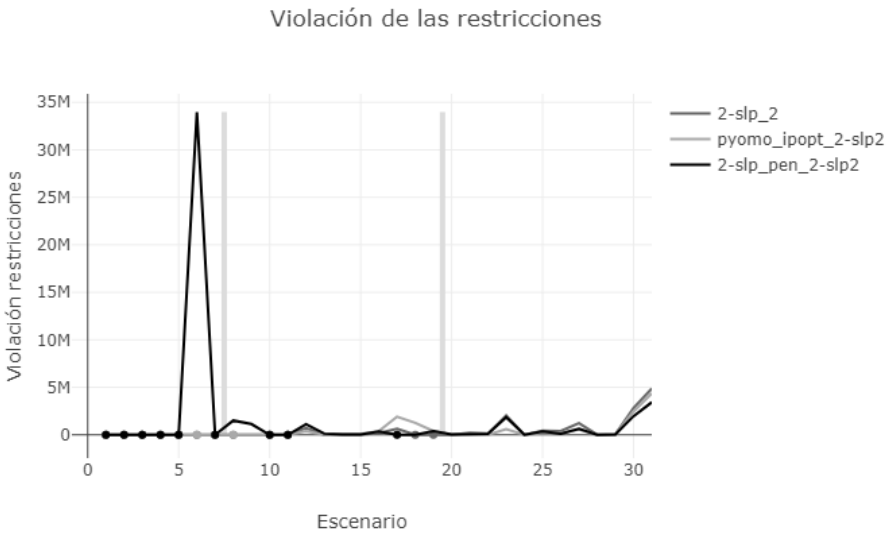


Figura 4.38: Violación total de las restricciones del modelo.

Por último, en la Figura 4.41 tenemos una representación de los tiempos de ejecución. Como es lógico pensar, el modelo completo tiene tiempos más pequeños que los tándems, ya que en este último caso estamos lanzando una concatenación de modelos, siendo el último el modelo completo con propagación de calidad y conservación del flujo en energía. Además, se puede ver que entre los dos tándems, exceptuando los escenarios 23 y 30, el segundo tándem resuelve los distintos escenarios en un menor tiempo, a pesar de que tiene una etapa intermedia a mayores. Esto se debe a que como habíamos visto al analizar los resultados del método 4, en general el algoritmo 2-SLP tenía un tiempo

Figura 4.39: Violación de las restricciones de propagación de calidad.

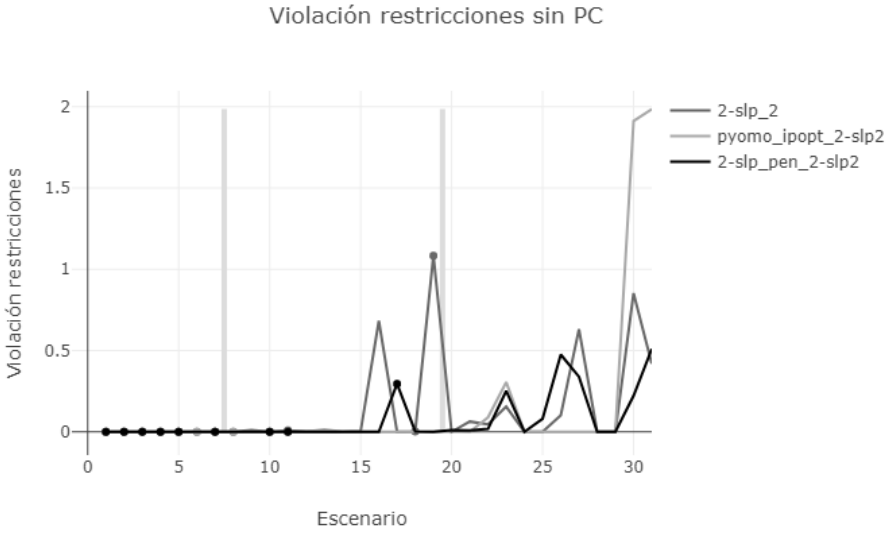


Figura 4.40: Violación de las restricciones excepto las de propagación de calidad.

de ejecución menor al solver Ipopt lanzado a través de Pyomo.

4.4. Conclusiones.

Como acabamos de comprobar a lo largo de este capítulo, el hecho de añadirle al modelo básico las restricciones necesarias para asegurar que el gas llegue a los puntos de demanda con la suficiente

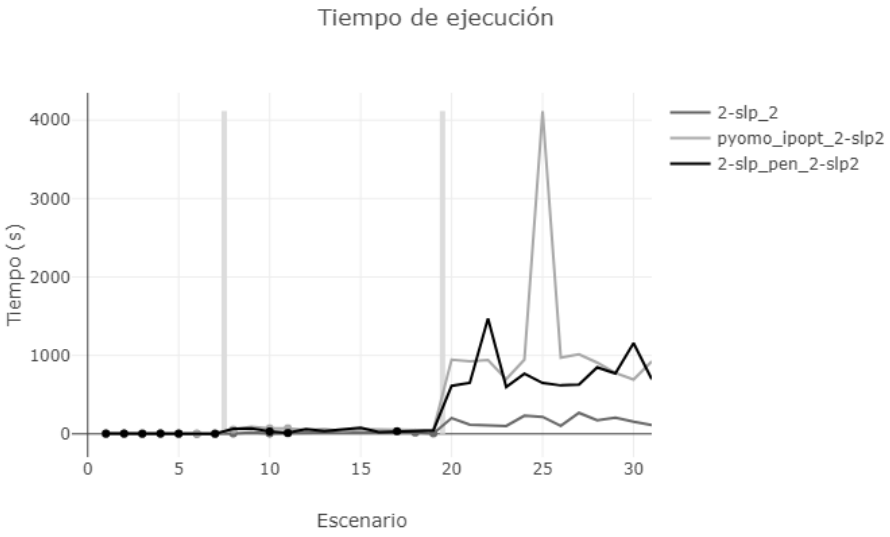


Figura 4.41: Tiempo de ejecución.

cantidad de energía, supone una gran complejidad al problema. De esta forma, vemos que con los distintos métodos y las distintas configuraciones lanzadas con cada uno de ellos en la mayoría de casos no se encuentran soluciones factibles para los escenarios de la red española, y en el caso de la red gallega lo hacen únicamente en un número reducido de casos.

Además, vimos que el método para el que más escenarios conseguíamos una solución factible para el modelo completo era el método 3 a través del algoritmo 2-SLP con la segunda configuración, donde directamente se resolvía el modelo básico con propagación de calidad y el flujo en energía era calculado en cada iteración del algoritmo a partir del poder calorífico. Asimismo, esta configuración tenía un buen rendimiento computacional, ya que los tiempos de ejecución eran menores que para el resto de configuraciones de ese método y para las mejores del resto de métodos comparables.

Por otro lado, habíamos observado que en cuanto a la violación de restricciones y coste en las estaciones de compresión en general también se obtenían mejores resultados para aquellas configuraciones que resolvían los distintos modelos a través del algoritmo 2-SLP.

De esta forma tenemos que el algoritmo creado por el equipo de GANESO™ es competitivo con grandes solvers de optimización local como son Ipopt y Knitro, obteniendo además mejores resultados para nuestros problemas, tanto a nivel computacional como de cantidad y calidad de las soluciones.

Apéndice A

Escenarios de prueba.

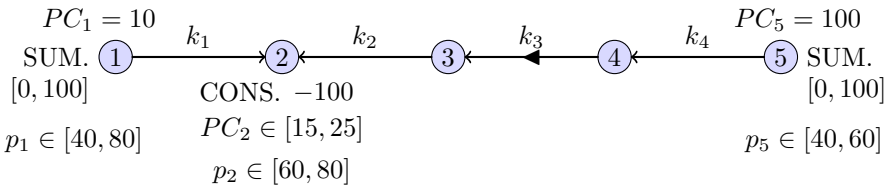


Figura A.1: Grafo y datos correspondientes al escenario 1.

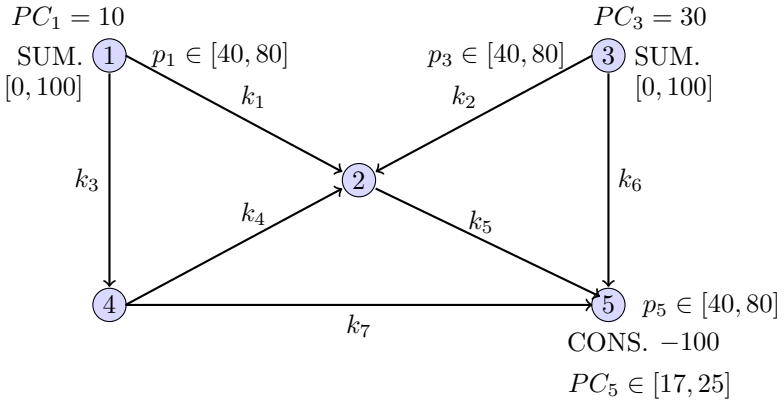


Figura A.2: Grafo y datos correspondientes al escenario 2.

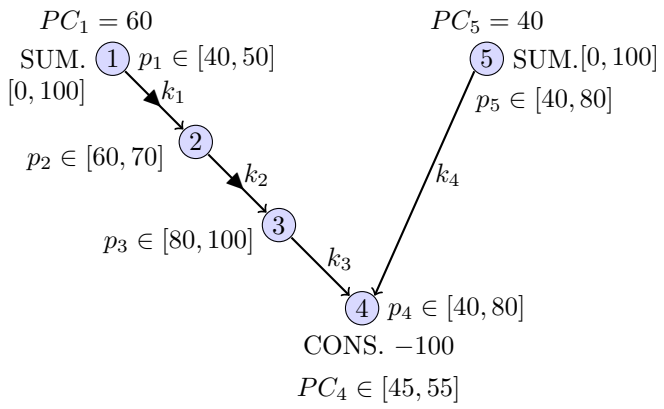


Figura A.3: Grafo y datos correspondientes al escenario 3.

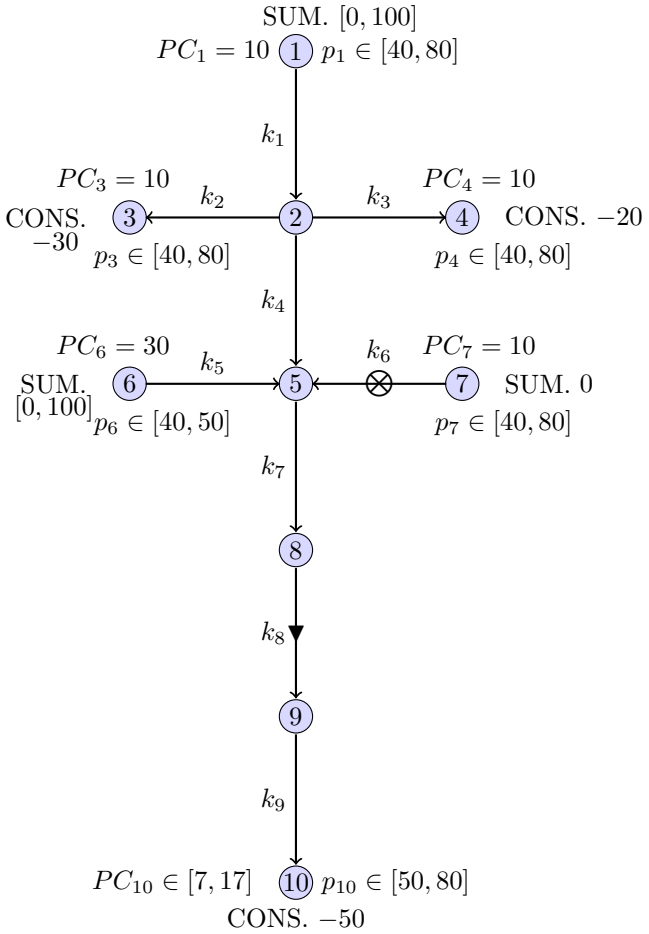


Figura A.4: Grafo y datos correspondientes al escenario 4.

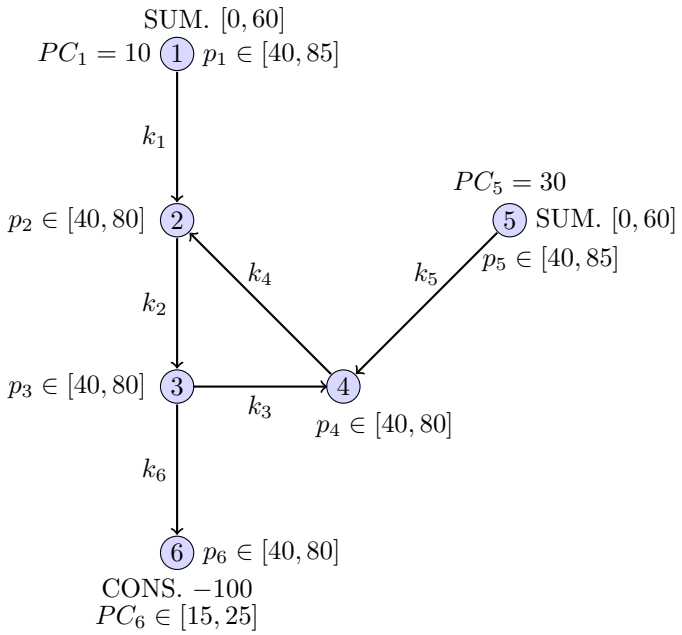


Figura A.5: Grafo y datos correspondientes al escenario 5.

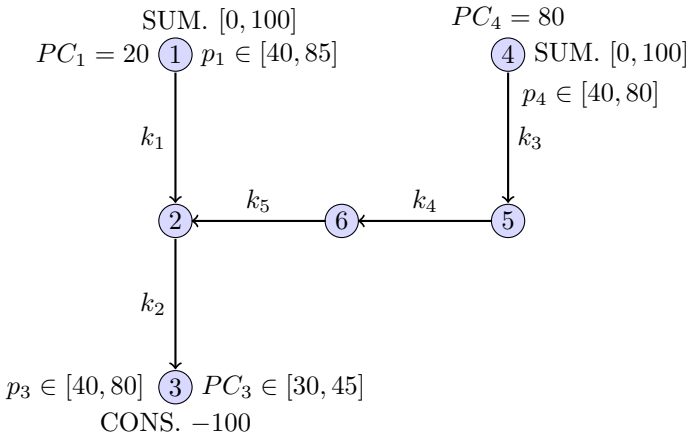


Figura A.6: Grafo y datos correspondientes al escenario 6.

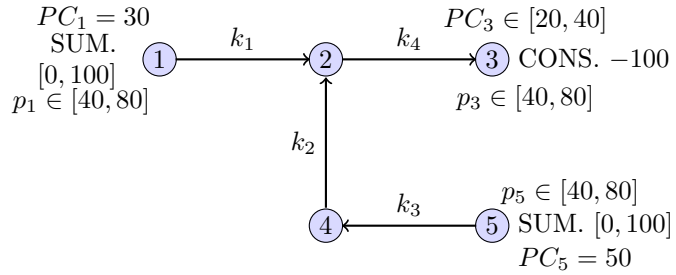


Figura A.7: Grafo y datos correspondientes al escenario 7.

Apéndice B

Scripts utilizados.

B.1. Generación de escenarios:

```
# Generacion de escenarios según el tipo de consumo (bajo, medio, alto)

print('Numero de escenario a generar? \n')
n_esc=input()

esc='escenario'+n_esc+'.xml'
print('El escenario a generar es:\n'+esc)

path=r'C:\Users\Andrea\repositorios\ganesoAMPLoptimization\test\Escenarios\PC\Escenario'+n_esc+
   r'\\'+esc

import xml.etree.ElementTree as ET
tree = ET.parse(path)
root = tree.getroot()

print('-----------------------------------------------')

for child in root:
   print('pagina:',child.attrib.get('{urn:schemas-microsoft-com:office:spreadsheet}Name'))
   if child.attrib.get('{urn:schemas-microsoft-com:office:spreadsheet}Name')=='nodes':
      pag_nodos=child

print('-----------------------------------------------')
nodes=ET.SubElement(root[5],'{urn:schemas-microsoft-com:office:spreadsheet}Worksheet',
   {'{urn:schemas-microsoft-com:office:spreadsheet}Name': 'nodes'})
nodestree=ET.ElementTree(pag_nodos)

for child in nodestree.getroot():
   if child.tag=='{urn:schemas-microsoft-com:office:spreadsheet}Table':
      tabla=child

ncol=tabla.attrib['{urn:schemas-microsoft-com:office:spreadsheet}ExpandedColumnCount']
nrow=tabla.attrib['{urn:schemas-microsoft-com:office:spreadsheet}ExpandedRowCount']

print('-----------------------------------------------')

table=ET.ElementTree(tabla)
```

```
datos=[]

for child in table.getroot():
   if child.tag=='{urn:schemas-microsoft-com:office:spreadsheet}Row':
      subtree=ET.ElementTree(child)
      for subchild in subtree.getroot():
         for subelem in subchild:
            if subelem.tag=='{urn:schemas-microsoft-com:office:spreadsheet}Data':
               datos.append(subelem.text)

print('---------------------------------------------')

import pandas as pd
import numpy as np

data=pd.DataFrame(np.array(datos[2*int(ncol):]).reshape(int(nrow)-2,int(ncol)),columns=
   datos[int(ncol):2*int(ncol)])
print(datos[int(ncol):2*int(ncol)])


print('---------------------------------------------')

type=pd.to_numeric(data['N'],downcast='integer')
cini=pd.to_numeric(data['cini'],downcast='float')
cmax=pd.to_numeric(data['cmax'],downcast='float')

n=len(data['i'])
sum_max=0
cons_total=0

for node in range(n):
   if type[node]==1 or type[node]==5:
      sum_max+=cmax[node]
   elif type[node]==2:
      cons_total+=cini[node]

print('Tipo consumo? (s:baixo, m:medio, l:alto)')
cons=input()

if cons=='s':
cte=0.5
elif cons=='m':
cte=0.6
elif cons=='l':
cte=0.75
else:
print('Tipo de consumo no válido')
exit()

l=cte*sum_max/cons_total

for node in range(n):
   cini_new=(l*cini[node])
   if type[node]==2:
      cini[node]=-cini_new
      data['cmin'][node]=cini[node]
```

```
      data['cmax'][node]=cini[node]
print(cini)
data['cini']=cini
datos[2*int(ncol):]=np.array(data).reshape(-1)

print('-----------------------------------------------')
#Modificacion del cini en el xml

for child in root:
   if child.attrib.get('{urn:schemas-microsoft-com:office:spreadsheet}Name')=='nodes':
      nodestree=ET.ElementTree(child)
      for child in nodestree.getroot():
      if child.tag=='{urn:schemas-microsoft-com:office:spreadsheet}Table':
         table=ET.ElementTree(child)
         i=0
         for child in table.getroot():
            if child.tag=='{urn:schemas-microsoft-com:office:spreadsheet}Row':
               subtree=ET.ElementTree(child)
               for subchild in subtree.getroot():
                  for subelem in subchild.iter(tag=
                  '{urn:schemas-microsoft-com:office:spreadsheet}Data'):
                     subelem.text=str(datos[i])
                     i+=1
sal='escenario'+n_esc+'_'+cons+'.xml'
tree.write(sal, "UTF-8",xml_declaration=True)
```

B.2. Generación de cotas de PC.

```
import sys

print('Numero de escenario a generar las cotas de PC? \n')
n_esc=input()

esc='escenario'+n_esc+'.xml'
print('El escenario a generar es:\n'+esc)
path=r'C:\Users\Andrea\repositorios\Bateria_escenarios\Escenario'+n_esc+r'\\'+esc

import os
comando=r'ganesoPy -e .\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+n_esc+r'.xml
-k opt_ampl --test -opt_options .\opciones_maxcomp.json -um 1'
print(comando)
os.system(comando)

import json

dat=r'C:\Users\Andrea\repositorios\Bateria_escenarios\Escenario'+n_esc+r'\test_temp_files\
 from_ganesoPy_to_ganesoKernelOpt.json'
pc=r'C:\Users\Andrea\repositorios\Bateria_escenarios\Escenario'+n_esc+r'\test_temp_files\
 cp_by_iteration.json'
with open(pc) as file:
   pc=json.load(file)
with open(dat) as file:
   data=json.load(file)

print('\n COTAS PARA PC AL 5%: \n')
```

```
for nodes in data["nodes"]["elements"]:
   if nodes['type']==1 or nodes['type']==5:
      pc["0"][str(nodes['id'])])
      print(nodes['id'],pc["0"][str(nodes['id'])],0,pc["0"][str(nodes['id'])] )
   elif nodes['type']==2:
      print(nodes['id'],0.95*pc["0"][str(nodes['id'])],0,1.05*pc["0"][str(nodes['id'])])

print('\n COTAS PARA PC AL 25%: \n')
for nodes in data["nodes"]["elements"]:
   if nodes['type']==1 or nodes['type']==5:
      print(nodes['id'],pc["0"][str(nodes['id'])],0,pc["0"][str(nodes['id'])] )
   elif nodes['type']==2:
      print(nodes['id'],0.75*pc["0"][str(nodes['id'])],0,1.25*pc["0"][str(nodes['id'])])
```

B.3. Ejecución método 1.

```
######################## script para ejecutar M1 ########################

import os
import json

opt=['opt_ampl','opt_pyomo','opt_slp']
solvers=["ipopt","knitro"]

with open('opciones.json', 'r+') as f:
   json_data = json.load(f)
   json_data["options"]["opt_pc_activation"] = False
   json_data["options"]["mu_abs_pc"] = 0
   json_data["options"]["op_range"] = 0
   for i in range(31):
      n_esc=str(i+1)
      if n_esc=='2' or n_esc=='5' or n_esc=='6' or n_esc=='7':
         json_data["options"]["obj_eecc_slp"] = 0
         json_data["options"]["penalty_slp"] = 0
      else:
         json_data["options"]["obj_eecc_slp"] = 1
         json_data["options"]["penalty_slp"] = 0.001
      if int(n_esc)<20:
         json_data["options"]["opt_valves_activation"] = False
         json_data["options"]["obj_difprespcv"] = 0
      else:
         json_data["options"]["opt_valves_activation"] = True
         json_data["options"]["obj_difprespcv"] = 0.0001
      for op in opt:
         if op!='opt_slp':
            for solver in solvers:
               json_data["solver"] = solver
               f.seek(0)
               f.write(json.dumps(json_data,indent=4))
               f.truncate()
               comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+
               r'\escenario'+n_esc+r'.xml -k '+op+r' --test -opt_options .\opciones.json -um 50'
               print(comando,solver)
               com=comando+r'>> salida_'+op+'_'+solver+'.txt'
               os.system(com)
```

```
            with open ('salida_'+op+'_'+solver+'.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
         else:
            json_data["options"]["linear_solver"] = "gurobi"
            json_data["options"]["max_it_slp"] = 100
            f.seek(0)
            f.write(json.dumps(json_data,indent=4))
            f.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+n_esc+
            r'.xml -k '+op+r' --test -opt_options .\opciones.json -um 50'
            print(comando)
            os.system(comando+'>> salida_slp.txt')
            with open ('salida_slp.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
```

B.4. Ejecución método 2.

```
######################## script para ejecutar M2 ########################

import os
import json

opt=['opt_ampl','opt_pyomo','opt_slp']
solvers=["ipopt","knitro"]

with open('opciones.json', 'r+') as f:
   json_data = json.load(f)
   json_data["options"]["mu_abs_pc"] = 100
   for i in range(31):
      n_esc=str(i+1)
      if n_esc=='2' or n_esc=='5' or n_esc=='6' or n_esc=='7':
         json_data["options"]["obj_eecc_slp"] = 0
         json_data["options"]["penalty_slp"] = 0
      else:
         json_data["options"]["obj_eecc_slp"] = 1
         json_data["options"]["penalty_slp"] = 0.001
      if int(n_esc)<20:
         json_data["options"]["opt_valves_activation"] = False
         json_data["options"]["obj_difprespcv"] = 0
      else:
         json_data["options"]["opt_valves_activation"] = True
         json_data["options"]["obj_difprespcv"] = 0.0001
      for op in opt:
         if op!='opt_slp':
            for solver in solvers:
            json_data["solver"] = solver
            f.seek(0)
            f.write(json.dumps(json_data,indent=4))
            f.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
            n_esc+r'.xml -k '+op+r' --test -opt_options .\opciones.json'
            print(comando,solver)
            com=comando+'>> salida_'+op+'_'+solver+'.txt'
            os.system(com)
```

```
            with open ('salida_'+op+'_'+solver+'.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
         else:
            json_data["options"]["linear_solver"] = "gurobi"
            json_data["options"]["max_it_slp"] = 100
            json_data["options"]["mu_abs_pc_slp"] = 100
            json_data["options"]["mu_abs_pc_cslp"] = 1e-2
            json_data["options"]["muqual_slp"] = 0.01
            json_data["options"]["mucarga_cslp"] = 100
            json_data["options"]["muqual_cslp"] = 1e-4
            json_data["options"]["muenbal_cslp"] = 100
            f.seek(0)
            f.write(json.dumps(json_data,indent=4))
            f.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
            n_esc+r'.xml -k '+op+r' --test -opt_options .\opciones.json'
            print(comando)
            os.system(comando+'>> salida_slp.txt')
            with open ('salida_slp.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
```

B.5. Ejecución método 3.

```
######################## script para ejecutar M3 ########################

import os
import json

opt=['opt_ampl','opt_pyomo','opt_slp']
solvers=["ipopt","knitro"]

with open('opciones.json', 'r+') as f:
   json_data = json.load(f)
   json_data["options"]["mu_abs_pc"] = 0.01
   json_data["options"]["opt_energy_flow_conservation"] = 1
   json_data["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 0
   json_data["options"]["opt_update_mass_flows_slp"] = "gas"
   for i in range(31):
      n_esc=str(i+1)
      if n_esc=='2' or n_esc=='5' or n_esc=='6' or n_esc=='7':
         json_data["options"]["obj_eecc_slp"] = 0
         json_data["options"]["penalty_slp"] = 0
      else:
         json_data["options"]["obj_eecc_slp"] = 1
         json_data["options"]["penalty_slp"] = 0.001
      if int(n_esc)<20:
         json_data["options"]["opt_valves_activation"] = False
         json_data["options"]["obj_difprespcv"] = 0
      else:
         json_data["options"]["opt_valves_activation"] = True
         json_data["options"]["obj_difprespcv"] = 0.0001
      for op in opt:
         if op!='opt_slp':
         for solver in solvers:
```

```
            json_data["solver"] = solver
            f.seek(0)
            f.write(json.dumps(json_data,indent=4))
            f.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+n_esc+r'.xml
            -k '+op+r' --test -opt_options .\opciones.json'
            print(comando,solver)
            com=comando+'>> salida_'+op+'_'+solver+'.txt'
            os.system(com)
            with open ('salida_'+op+'_'+solver+'.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
      else:
         json_data["options"]["linear_solver"] = "gurobi"
         json_data["options"]["max_it_slp"] = 100
         json_data["options"]["mu_abs_pc_slp"] = 100
         json_data["options"]["mu_abs_pc_cslp"] = 1e-2
         json_data["options"]["muqual_slp"] = 0.01
         json_data["options"]["mucarga_cslp"] = 100
         json_data["options"]["muqual_cslp"] = 1e-4
         json_data["options"]["muenbal_cslp"] = 100
         f.seek(0)
         f.write(json.dumps(json_data,indent=4))
         f.truncate()
         comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+n_esc+r'.xml
          -k '+op+r' --test -opt_options .\opciones.json'
         print(comando,'slp 1')
         os.system(comando+'>> salida_slp1.txt')
         with open ('salida_slp1.txt', 'a') as sal:
            sal.write('\n Fin escenario '+n_esc+'\n')
         json_data["options"]["opt_energy_flow_conservation"] = 0
         json_data["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 1
         json_data["options"]["opt_update_mass_flows_slp"] = "cp"
         f.seek(0)
         f.write(json.dumps(json_data,indent=4))
         f.truncate()
         comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
         n_esc+r'.xml
         -k '+op+r' --test -opt_options .\opciones.json'
         print(comando,'slp 2')
         os.system(comando+'>> salida_slp2.txt')
         with open ('salida_slp2.txt', 'a') as sal:
            sal.write('\n Fin escenario '+n_esc+'\n')
```

B.6. Ejecución método 4 opción 1.

```
######################## script para ejecutar M4 ########################

import os
import json

opt=['opt_pyomo','opt_slp']

with open('opciones.json', 'r+') as f:
   json_data = json.load(f)
```

```
json_data["solver"] = "ipopt"
json_data["options"]["max_it_slp"] = 100
json_data["options"]["linear_solver"] = "gurobi"
with open('opcionesM2.json', 'r+') as g:
   json_data1 = json.load(g)
   json_data1["options"]["mu_abs_pc_slp"] = 100
   json_data1["options"]["mu_abs_pc_cslp"] = 1e-2
   json_data1["options"]["muqual_slp"] = 0.01
   json_data1["options"]["mucarga_cslp"] = 100
   json_data1["options"]["muqual_cslp"] = 1e-4
   json_data1["solver"] = "ipopt"
   json_data1["options"]["max_it_slp"] = 100
   json_data1["options"]["linear_solver"] = "gurobi"
   with open('opcionesM3.json','r+') as h:
      json_data2 = json.load(h)
      json_data2["options"]["opt_pc_activation"] = 1
      json_data2["options"]["max_it_slp"] = 100
      json_data2["options"]["linear_solver"] = "gurobi"
      json_data2["options"]["mu_abs_pc_slp"] = 100
      json_data2["options"]["mu_abs_pc_cslp"] = 1e-2
      json_data2["options"]["muqual_slp"] = 0.01
      json_data2["options"]["mucarga_cslp"] = 100
      json_data2["options"]["muqual_cslp"] = 1e-4
      json_data2["options"]["muenbal_cslp"] = 100
      for i in range(31):
         json_data2["options"]["opt_energy_flow_conservation"] = 1
         json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 0
         json_data2["options"]["opt_update_mass_flows_slp"] = "gas"
         n_esc=str(i+1)
         if n_esc=='2' or n_esc=='5' or n_esc=='6' or n_esc=='7':
            json_data["options"]["obj_eecc_slp"] = 0
            json_data["options"]["penalty_slp"] = 0
            json_data1["options"]["obj_eecc_slp"] = 0
            json_data1["options"]["penalty_slp"] = 0
            json_data2["options"]["obj_eecc_slp"] = 0
            json_data2["options"]["penalty_slp"] = 0
         else:
            json_data["options"]["obj_eecc_slp"] = 1
            json_data["options"]["penalty_slp"] = 0.001
            json_data1["options"]["obj_eecc_slp"] = 1
            json_data1["options"]["penalty_slp"] = 0.001
            json_data2["options"]["obj_eecc_slp"] = 1
            json_data2["options"]["penalty_slp"] = 0.001
         if int(n_esc)<20:
            json_data["options"]["opt_valves_activation"] = False
            json_data["options"]["obj_difprespcv"] = 0
            json_data1["options"]["opt_valves_activation"] = False
            json_data1["options"]["obj_difprespcv"] = 0
            json_data2["options"]["opt_valves_activation"] = False
            json_data2["options"]["obj_difprespcv"] = 0
         else:
            json_data["options"]["opt_valves_activation"] = True
            json_data["options"]["obj_difprespcv"] = 0.0001
            json_data1["options"]["opt_valves_activation"] = True
            json_data1["options"]["obj_difprespcv"] = 0.0001
            json_data2["options"]["opt_valves_activation"] = True
```

```
   json_data2["options"]["obj_difprespcv"] = 0.0001
f.seek(0)
f.write(json.dumps(json_data,indent=4))
f.truncate()
g.seek(0)
g.write(json.dumps(json_data1,indent=4))
g.truncate()
h.seek(0)
h.write(json.dumps(json_data2,indent=4))
h.truncate()
for op in ['opt_slp']:#opt:
   if op!='opt_slp':
      comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+
      r'\escenario'+n_esc+r'.xml -k '+op+' '+op+r' opt_slp --test -opt_options
      .\opciones.json .\opcionesM2.json .\opcionesM3.json'
      print(comando)
      com=comando+'>> salida_'+op+'_ipopt_slp1.txt'
      os.system(com)
      with open ('salida_'+op+'_ipopt_slp1.txt', 'a') as sal:
         sal.write('\n Fin escenario '+n_esc+'\n')
      json_data2["options"]["opt_energy_flow_conservation"] = 0
      json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 1
      json_data2["options"]["opt_update_mass_flows_slp"] = "cp"
      h.seek(0)
      h.write(json.dumps(json_data2,indent=4))
      h.truncate()
      comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+
      r'\escenario'+n_esc+r'.xml -k '+op+' '+op+r' opt_slp --test -opt_options
      .\opciones.json .\opcionesM2.json .\opcionesM3.json'
      print(comando)
      os.system(comando+'>> salida_'+op+'_ipopt_slp2.txt')
      with open ('salida_'+op+'_ipopt_slp2.txt', 'a') as sal:
         sal.write('\n Fin escenario '+n_esc+'\n')
   else:
      json_data2["options"]["opt_energy_flow_conservation"] = 1
      json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 0
      json_data2["options"]["opt_update_mass_flows_slp"] = "gas"
      h.seek(0)
      h.write(json.dumps(json_data2,indent=4))
      h.truncate()
      comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+
      r'\escenario'+n_esc+r'.xml -k '+op+' '+op+r' opt_slp --test -opt_options
      .\opciones.json .\opcionesM2.json .\opcionesM3.json'
      print(comando)
      os.system(comando+'>> salida_'+op+'_slp1.txt')
      with open ('salida_'+op+'_slp1.txt', 'a') as sal:
         sal.write('\n Fin escenario '+n_esc+'\n')
      json_data2["options"]["opt_energy_flow_conservation"] = 0
      json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 1
      json_data2["options"]["opt_update_mass_flows_slp"] = "cp"
      h.seek(0)
      h.write(json.dumps(json_data2,indent=4))
      h.truncate()
      comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+
      r'\escenario'+n_esc+r'.xml -k '+op+' '+op+r' opt_slp --test -opt_options
      .\opciones.json .\opcionesM2.json .\opcionesM3.json'
```

```
                print(comando)
                os.system(comando+'>> salida_'+op+'_slp2.txt')
                with open ('salida_'+op+'_slp2.txt', 'a') as sal:
                   sal.write('\n Fin escenario '+n_esc+'\n')
```

B.7. Ejecución método 4 opción 2.

```
############ script para ejecutar M4 con propagación de calidad penalizada ##############

import os
import json

opt=['opt_pyomo','opt_slp']

with open('opciones.json', 'r+') as f:
   json_data = json.load(f)
   json_data["solver"] = "ipopt"
   json_data["options"]["max_it_slp"] = 100
   json_data["options"]["linear_solver"] = "gurobi"
   with open('opcionesM2.json', 'r+') as g:
      json_data1 = json.load(g)
      json_data1["options"]["opt_pc_penalized"] = 0
      json_data1["options"]["mu_abs_pc_slp"] = 0.01
      json_data1["options"]["alpha_penalty_pc"] = 0
      json_data1["options"]["mu_abs_pc_cslp"] = 1e-2
      json_data1["options"]["muqual_slp"] = 0.01
      json_data1["options"]["mucarga_cslp"] = 100
      json_data1["options"]["muqual_cslp"] = 1e-4
      json_data1["solver"] = "ipopt"
      json_data1["options"]["max_it_slp"] = 100
      json_data1["options"]["linear_solver"] = "gurobi"
      with open('opcionesM2p.json', 'r+') as g2:
         json_data12 = json.load(g2)
         json_data12["options"]["opt_pc_penalized"] = 1
         json_data12["options"]["alpha_penalty_pc"] = 0.1
         json_data12["options"]["muqual_slp"] = 0.01
         json_data12["options"]["mucarga_cslp"] = 100
         json_data12["options"]["muqual_cslp"] = 1e-4
         json_data12["solver"] = "ipopt"
         json_data12["options"]["max_it_slp"] = 100
         json_data12["options"]["linear_solver"] = "gurobi"
         with open('opcionesM3.json','r+') as h:
            json_data2 = json.load(h)
            json_data2["options"]["opt_pc_activation"] = 1
            json_data2["options"]["max_it_slp"] = 100
            json_data2["options"]["linear_solver"] = "gurobi"
            json_data2["options"]["mu_abs_pc_slp"] = 100
            json_data2["options"]["mu_abs_pc_cslp"] = 1e-2
            json_data2["options"]["muqual_slp"] = 0.01
            json_data2["options"]["mucarga_cslp"] = 100
            json_data2["options"]["muqual_cslp"] = 1e-6
            json_data2["options"]["muenbal_cslp"] = 100
            for i in range(19,31):
               json_data2["options"]["opt_energy_flow_conservation"] = 1
```

```
json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 0
json_data2["options"]["opt_update_mass_flows_slp"] = "gas"
n_esc=str(i+1)
if n_esc=='2' or n_esc=='5' or n_esc=='6' or n_esc=='7':
   json_data["options"]["obj_eecc_slp"] = 0
   json_data["options"]["penalty_slp"] = 0
   json_data1["options"]["obj_eecc_slp"] = 0
   json_data1["options"]["penalty_slp"] = 0
   json_data2["options"]["obj_eecc_slp"] = 0
   json_data2["options"]["penalty_slp"] = 0
else:
   json_data["options"]["obj_eecc_slp"] = 1
   json_data["options"]["penalty_slp"] = 0.001
   json_data1["options"]["obj_eecc_slp"] = 1
   json_data1["options"]["penalty_slp"] = 0.001
   json_data2["options"]["obj_eecc_slp"] = 1
   json_data2["options"]["penalty_slp"] = 0.001
if int(n_esc)<20:
   json_data["options"]["opt_valves_activation"] = False
   json_data["options"]["obj_difprespcv"] = 0
   json_data1["options"]["opt_valves_activation"] = False
   json_data1["options"]["obj_difprespcv"] = 0
   json_data2["options"]["opt_valves_activation"] = False
   json_data2["options"]["obj_difprespcv"] = 0
else:
   json_data["options"]["opt_valves_activation"] = True
   json_data["options"]["obj_difprespcv"] = 0.0001
   json_data1["options"]["opt_valves_activation"] = True
   json_data1["options"]["obj_difprespcv"] = 0.0001
   json_data2["options"]["opt_valves_activation"] = True
   json_data2["options"]["obj_difprespcv"] = 0.0001
f.seek(0)
f.write(json.dumps(json_data,indent=4))
f.truncate()
g.seek(0)
g.write(json.dumps(json_data1,indent=4))
g.truncate()
g2.seek(0)
g2.write(json.dumps(json_data12,indent=4))
g2.truncate()
h.seek(0)
h.write(json.dumps(json_data2,indent=4))
h.truncate()
for op in opt:
   if op!='opt_slp':
      comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
      n_esc+r'.xml -k '+op+' '+op+' '+op+r' opt_slp --test -opt_options
      .\opciones.json .\opcionesM2p.json .\opcionesM2.json .\opcionesM3.json'
      print(comando)
      com=comando+'>> salida_pen_'+op+'_ipopt_slp1.txt'
      os.system(com)
      with open ('salida_pen_'+op+'_ipopt_slp1.txt', 'a') as sal:
         sal.write('\n Fin escenario '+n_esc+'\n')
      json_data2["options"]["opt_energy_flow_conservation"] = 0
      json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 1
      json_data2["options"]["opt_update_mass_flows_slp"] = "cp"
```

```
            h.seek(0)
            h.write(json.dumps(json_data2,indent=4))
            h.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
            n_esc+r'.xml -k '+op+' '+op+' '+op+r' opt_slp --test -opt_options
            .\opciones.json .\opcionesM2p.json .\opcionesM2.json .\opcionesM3.json'
            print(comando)
            os.system(comando+'>> salida_pen_'+op+'_ipopt_slp2.txt')
            with open ('salida_pen_'+op+'_ipopt_slp2.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
         else:
            json_data2["options"]["opt_energy_flow_conservation"] = 1
            json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 0
            json_data2["options"]["opt_update_mass_flows_slp"] = "gas"
            h.seek(0)
            h.write(json.dumps(json_data2,indent=4))
            h.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
            n_esc+r'.xml -k '+op+' '+op+' '+op+r' opt_slp --test -opt_options
            .\opciones.json .\opcionesM2p.json .\opcionesM2.json .\opcionesM3.json'
            print(comando)
            os.system(comando+'>> salida_pen_'+op+'_slp1.txt')
            with open ('salida_pen_'+op+'_slp1.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
            json_data2["options"]["opt_energy_flow_conservation"] = 0
            json_data2["options"]["opt_iteratively_update_mass_by_quality_flows_slp"] = 1
            json_data2["options"]["opt_update_mass_flows_slp"] = "cp"
            h.seek(0)
            h.write(json.dumps(json_data2,indent=4))
            h.truncate()
            comando=r'ganesoPy -e ..\Bateria_escenarios\Escenario'+n_esc+r'\escenario'+
            n_esc+r'.xml -k '+op+' '+op+' '+op+r' opt_slp --test -opt_options
            .\opciones.json .\opcionesM2p.json .\opcionesM2.json .\opcionesM3.json'
            print(comando)
            os.system(comando+'>> salida_pen_'+op+'_slp2.txt')
            with open ('salida_pen_'+op+'_slp2.txt', 'a') as sal:
               sal.write('\n Fin escenario '+n_esc+'\n')
```

B.8. Información necesaria para las gráficas.

En este caso, el script es el mismo para todos los métodos, lo único que varía es el nombre de los archivos a partir de los cuales saca la información. Por ello, mostraremos únicamente uno de ellos.

```
files=['salida_opt_ampl_ipopt.txt','salida_opt_ampl_knitro.txt',
'salida_opt_pyomo_ipopt.txt','salida_opt_pyomo_knitro.txt','salida_slp1.txt','salida_slp2.txt']

ncol=['dimgray','dimgray','darkgray','darkgray','black','black']
mark=['dash','dashdot','dash','dashdot','solid','dashdot']
lab=['ampl_ipopt','ampl_knitro','pyomo_ipopt','pyomo_knitro','2-slp_1','2-slp_2']

inicost=[]
with open('salidamaxcomp.txt','rt') as in_file:
   par=True
   while par:
      for line in in_file:
```

```
         if 'Total consumption in compressor stations' in line:
            n=line.find(':')
            s=line.find('(')
            v=line[n+1:s]
         if 'Fin escenario' in line:
            par=False
            inicost.append(float(v))
            v=0

n_cons=[]
with open('n_cons.txt','rt') as in_file:
   par=True
   while par:
      for line in in_file:
         if 'Number of constraints:' in line:
            n=line.find(':')
            v=line[n+1:]
         if 'Fin escenario' in line:
            par=False
            n_cons.append(int(v))

costt,timet,max_violt,conv,gct,solt,pct=[],[],[],[],[],[],[]
for file in files:
j=1
sol=[]
cost,time,max_viol,gc,pc=[],[],[],[],[]
with open (file, 'rt') as in_file:
   par=True
   while par:
      for line in in_file:
         if 'Obj gc' in line:
            n=line.find('|')
            s=line.find('|',n+1)
            n2=line.find('|',s+1)
            v=line[s+1:n2].strip(' ')
         if 'Obj difcom' in line:
            n=line.find('|')
            s=line.find('|',n+1)
            n2=line.find('|',s+1)
            v2=line[s+1:n2].strip(' ')
         if 'Obj difpcv' in line:
            n=line.find('|')
            s=line.find('|',n+1)
            n2=line.find('|',s+1)
            v3=line[s+1:n2].strip(' ')
         if 'Ploss' in line:
            n=line.find('|')
            s=line.find('|',n+1)
            n2=line.find('|',s+1)
            r1=line[s+1:n2].strip(' ')
         if 'CP_prop' in line:
            n=line.find('|')
            s=line.find('|',n+1)
            n2=line.find('|',s+1)
            r2=line[s+1:n2].strip(' ')
            if file=='salida_slp1.txt' and float(r2)<100:
```

```
                sol.append(j)
            if file=='salida_slp2.txt' and float(r2)<100:
                sol.append(j)
         if 'EN_bal' in line:
            n=line.find('|')
            s=line.find('|',n+1)
            n2=line.find('|',s+1)
            r3=line[s+1:n2].strip(' ')
         if 'Total time:' in line:
            n=line.find(':')
            s=line.find('seconds')
            t=line[n+1:s]
         if 'optimal solution' in line or 'Optimal Solution' in line:
            sol.append(j)
         if 'Fin escenario' in line:
            j+=1
            par=False
            c=float(v)+float(v2)+float(v3)
            r=float(r1)+float(r2)+float(r3)
            pc.append(float(r2))
            cost.append(float(c))
            gc.append(float(v))
            max_viol.append(float(r))
            time.append(float(t))
   pct.append(pc)
   solt.append(sol)
   costt.append(cost)
   gct.append(gc)
   max_violt.append(max_viol)
   timet.append(time)

for i in range(len(files)):
   for j in range(31):
      pct[i][j]=pct[i][j]/n_cons[j]
      max_violt[i][j]=max_violt[i][j]/n_cons[j]

bbar=['black', 'dimgray', 'darkgray']
names=['Escenarios simples', 'Red gallega', 'Red española']
convit=[]
convi1,convi2,convi3=[],[],[]
for i in range(len(files)):
   i1,i2,i3=0,0,0
   for j in range(len(solt[i])):
      if solt[i][j]<8:
         i1+=1
      elif solt[i][j]>=8 and solt[i][j]<20:
         i2+=1
      else:
         i3+=1
   convi1.append(i1)
   convi2.append(i2)
   convi3.append(i3)
convit.append(convi1)
convit.append(convi2)
convit.append(convi3)
```

Bibliografía

[1] Bazaraa MS, Sherali HD, Shetty CM (2006) Nonlinear Programming: Theory and Algorithms. Wiley-Interscience, New Jersey, pp 568-576.

[2] Byrd R H, Nocedal J, Waltz R A (2006) Knitro: An Integrated Package for Nonlinear Optimization. En: Di Pillo G, Roma M (eds) Large-Sacle Nonlinear Optimization. Springer, Boston, pp 35-59.

[3] Fourer R, Gay D, Kernighan B (2003) AMPL: A Modeling Language for Mathematical Programming. Thomson.

[4] Frøysa K, Lunde P (2005) Density and calorific value measurement in natural gas using ultrasonic flow meters. 23º International North Sea Flow Measurement Workshop Tønsberg, Norway, 18 - 21 October 2005.

[5] González Díaz J, Casares de Cal MA (2014) Programación Lineal y Entera (Grado en Matemáticas), Departamento de Estadística, Análisis Matemático y Optimización. Universidade de Santiago de Compostela.

[6] González Díaz J (2017) Programación Matemática (Máster en Técnicas Estadísticas), Departamento de Estadística, Análisis Matemático y Optimización. Universidade de Santiago de Compostela.

[7] González Rueda AM (2017) Gas transmission networks: optimization algorithms and cost allocation methodologies. Tesis, Universidade de Santiago de Compostela.

[8] González Rueda AM, González Díaz J, Fernández de Córdoba MP (2019) A twist on SLP algorithms for NLP and MINLP problems: an application to gas transmission networks. Optimization and Engineering 20:349-395.

[9] Hager T, Bentaleb A, Werhmann EA (2012) Simulation system with calorific value tracking for gas distribution grids with an incomplete measurement infraestructure. XX IMEKO World Congress. Metrology for Green Growth. September 9-14, 2012, Busan, Republic of Korea.

[10] Hart W, Laird C, Watson J-P, Woodruff D, Hackebeil G, Nicholson B (2017) Pyomo - Optimization Modeling in Python. Second Edition. Springer.

[11] Hiller B, Koch T, Schewe L, Schwarz R, Schweiger J (2018) A system to evaluate gas network capacities: Concepts and implementation. European Journal of Operational Research 270:797-808.

[12] Koch T, Hiller B, Pfetsch ME, Schewe L (Eds.) (2015) Evaluating Gas Network Capacities. Society for Industrial and Applied Mathematics, Mathematical Optimization Society, Philadelphia.

[13] Van der Hoeven T (2004) Math in Gas and the Art of Linearization. Energy Delta Institute.

[14] Wächter A, Vigerske S (2015) Introduction to Ipopt: A tutorial for downloading, installing and using Ipopt. https://projects.coin-or.org/Ipopt/export/2768/stable/3.11/Ipopt/doc/ documentation.pdf. Accedido 4 de Junio de 2019.