COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Modelado de prioridades en la selección de alimentadores para el deslastre de carga por baja frecuencia

Autor: Alejandro Arroyo Gil

Director: Mohammad Rajabdorri

Co-Director: Oswaldo Arenas Crespo

Madrid

Julio de 2026

MODELADO DE PRIORIDADES EN LA SELECCIÓN DE ALIMENTADORES PARA EL DESLASTRE DE CARGA POR BAJA FRECUENCIA

Autor: Arroyo Gil, Alejandro
Directores: Rajabdorri, Mohammad y Arenas Crespo, Oswaldo
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este trabajo aborda la selección de un rango de valores para los coeficientes de la formulación de un problema multiobjetivo resuelto mediante el método de la suma ponderada. El problema busca minimizar el desvío del deslastre respecto a un objetivo en los esquemas de deslastre de carga por baja frecuencia.

Palabras clave: Optimización multiobjetivo, suma ponderada, deslastre de carga por baja frecuencia, MILP, selección de alimentadores

1. Introducción

La estabilidad de frecuencia es una condición esencial para operar un sistema eléctrico con seguridad. Cuando una perturbación grave desequilibra generación y demanda, los esquemas de deslastre de carga por baja frecuencia (EDCF) desconectan consumo de forma controlada para evitar el colapso de la red. Son la última línea de defensa. La transición energética ha desplazado el modelo de grandes centrales síncronas hacia uno con alta penetración de generación renovable y distribuida, lo que introduce una variabilidad considerable en los flujos de potencia de los alimentadores seleccionados para operar en el EDCF degradando el rendimiento del esquema EDCF tradicional (basado en relés de baja frecuencia). El resultado es una brecha entre el diseño del plan de deslastre y el deslastre obtenido durante la operación del esquema. Esta brecha compromete el margen de seguridad justo en el momento más crítico.

2. Definición del proyecto

El proyecto se desarrolla en el contexto de una selección de alimentadores evaluando los seis escalones acumulados de un esquema EDCF típico conforme a la regulación europea. La selección se plantea como un problema de optimización multiobjetivo y se resuelve mediante el método de la suma ponderada, cuyos pesos son los coeficientes de coste CP, CC y CO.

El objetivo central del trabajo . Es hallar un rango de valores que produzcan un funcionamiento coherente del esquema. Para llegar a él se fijan tres metas. La primera es revisar la literatura de optimización multiobjetivo aplicada al problema, la segunda es implementar y evaluar el enfoque de suma ponderada, y la tercera es estudiar la sensibilidad de cada coeficiente hasta obtener una recomendación concreta de parametrización.

Todas las pruebas mantienen el parámetro de entrada XS = 0, es decir, un esquema sin alimentadores preasignados como referencia.

3. Descripción del modelo/sistema/herramienta

El problema se formula como un modelo lineal entero mixto (MILP). La función objetivo es:

$$\min_{s} \frac{CP}{|\mathcal{I}|}\sum_{i} F_i \cdot x_i + \frac{CC}{|\mathcal{I}|}\sum_{i}(n_i^{\uparrow} + n_i^{\downarrow}) + \frac{CO}{LS \cdot |\mathcal{T}|}\sum_{t}(s_t^{\uparrow} + s_t^{\downarrow}) + 100$$

A partir de aquí se definen los siguientes términos.

- $T1 = \frac{1}{|\mathcal{I}|}\sum_i F_i \cdot x_i$
- $T2 = \frac{1}{|\mathcal{I}|}\sum_i (n_i^{\uparrow} + n_i^{\downarrow})$
- $T3 = \frac{1}{LS \cdot |\mathcal{T}|}\sum_t (s_t^{\uparrow} + s_t^{\downarrow})$

Cada coeficiente multiplica a un término. CP multiplica a T1, el cual penaliza la selección de alimentadores ponderada por su importancia F_i. CC multiplica a T2 que cuenta los cambios de asignación. CO multiplica a T3, que es el término encargado de medir el desvío del deslastre respecto al objetivo del escalón. La variable binaria x_i indica si el alimentador se desconecta, las holguras $s_i^{\uparrow}, s_i^{\downarrow}$ recogen el desvío por exceso y por defecto, y las variables $n_i^{\uparrow}, n_i^{\downarrow}$ cuantifican los cambios frente a la referencia XS.

Los alimentadores se agrupan en categorías por criticidad, de mayor a menor prioridad de desconexión: Industrial, Rural, Doméstico, Público y N/A. El modelo parte de datos históricos reales y anonimizados de distribución en España, con potencia horaria por

alimentador y potencia total del sistema. La formulación se implementa en Python con Pyomo y se resuelve con Gurobi.

El código, recogido en el Anexo I, se usa para recorrer de forma sistemática los casos definidos por los coeficientes, lo que da soporte a una metodología iterativa donde cada ejecución sirve para decidir el ajuste siguiente.

4. Resultados

El estudio arranca de un caso base con CP = CC = 0, donde el optimizador solo minimiza el desvío. Este caso fija el mínimo de T3 y el máximo de T1, y sirve de cota para el resto. Como lo importante es la relación entre coeficientes, no su valor absoluto, se fijó CO = 1 para usar como referencia.

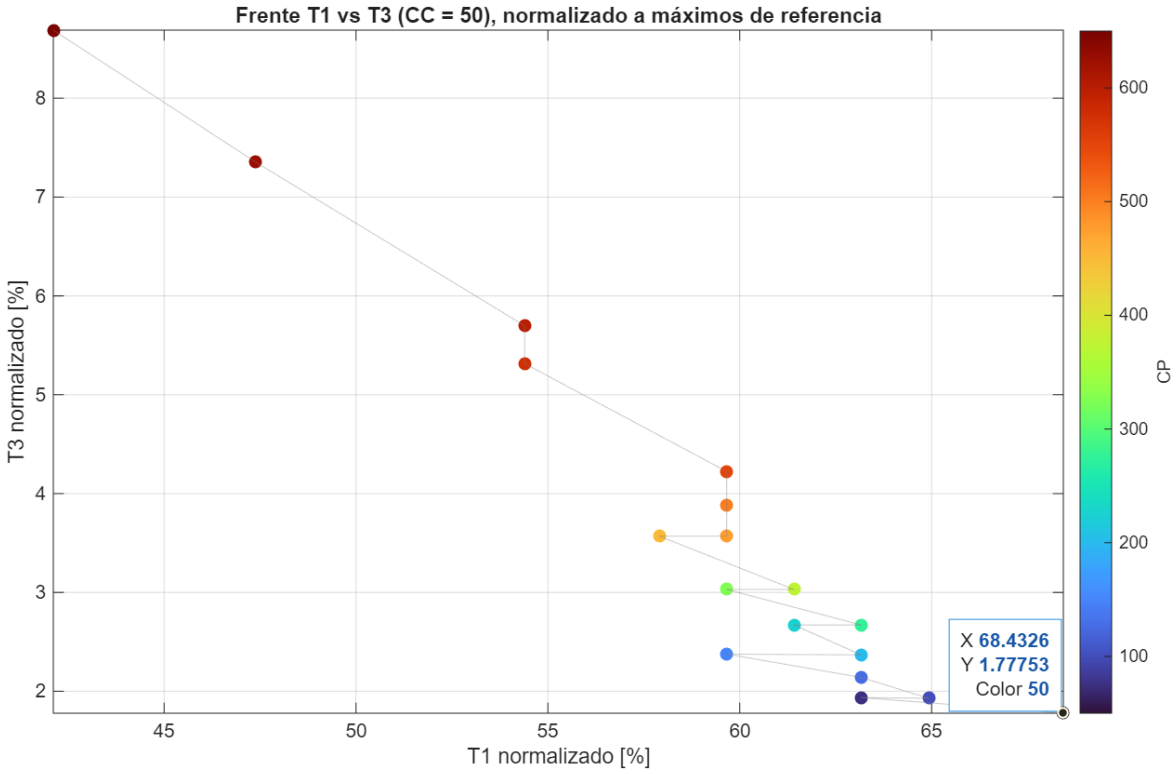
A partir de ahí se hizo un análisis univariante en escala logarítmica, variando un coeficiente y dejando los otros fijos. Los casos de tipo A recorren CP y los de tipo B recorren CC. En ambos, la zona de quiebre donde el coeficiente empieza a alterar la solución cae entre 10 y 100. Estos casos también mostraron que CP empuja el desvío con mucha más fuerza que CC. Basta comparar los extremos: con CP = 1000 el T3 se dispara, mientras que con CC en ese mismo valor apenas se mueve.

El análisis bivariante confirmó la coherencia del modelo, T1 disminuye al crecer CP y T2 al crecer CC. Con XS = 0 el término de cambios de asignación se vuelve redundante, ya que T2 se reduce a contar alimentadores y T1 domina esa misma decisión con más información.

El foco pasó entonces al compromiso entre T1 y T3. La primera aproximación se hizo sobre los casos del barrido bivariante, representando T1 frente a T3 sin normalizar y descartando los valores con T3 > 50 para leer mejor la nube. Ahí se estimó que la zona de funcionamiento razonable rondaba T1 = 0,14 y T3 = 15.

Para fijar un criterio objetivo se normalizó T3 con su punto nadir aproximado, el máximo desvío respecto al deslastre objetivo, obtenido al forzar toda la prioridad hacia T1 y T2. Ese máximo resultó ser 614,29. El mínimo normalizado quedó en 0,9556%. En base a la experiencia y tolerancia del distribuidor se fijó un umbral de aceptación, el cual se situó medio punto por encima, en 1,4556%.

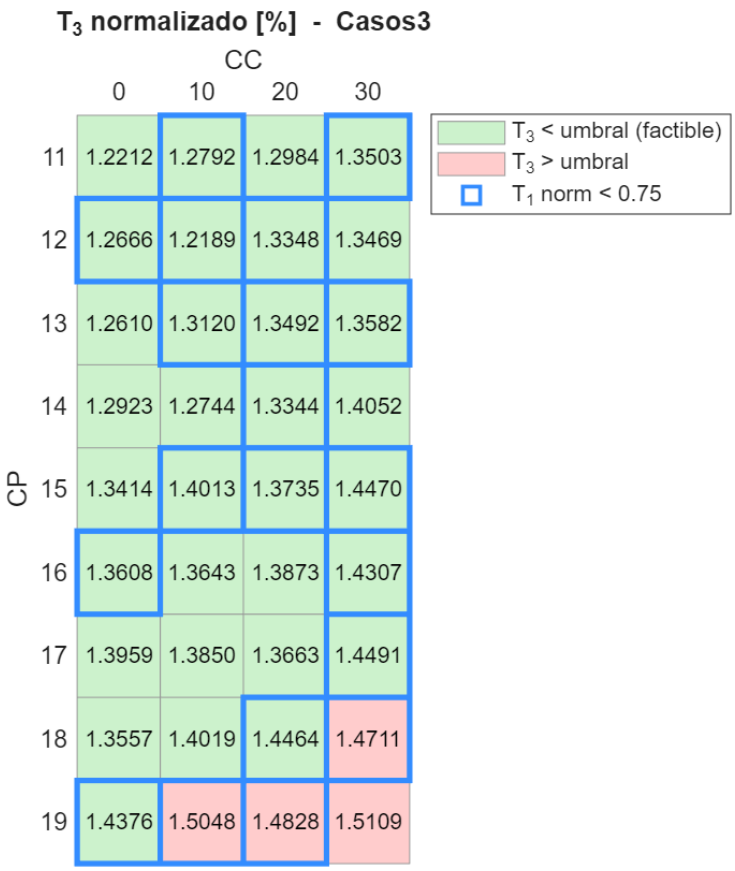
Al llevar los casos ya probados a escala normalizada se vio que ni el punto más bajo disponible, con CC = 50 y CP = 50, bajaba del umbral.



El punto de trabajo no estaba entre esos casos. Los coeficientes probados eran demasiado altos.

Se lanzó entonces una segunda tanda de casos, con CC de 0 a 50 en pasos de 10 y CP de 10 a 100 en pasos de 10. Esto dio sesenta casos en total. La mayoría seguía por encima del umbral, pero apareció una señal clara: la reducción de T1 era mayor con CC = 0, y para casi todos los valores de CC el T3 rebasaba el umbral en cuanto CP ≥ 20. Se demostró que CP debía estar por debajo de 20.

Eso acotó la búsqueda a un barrido fino, con CP entre 11 y 19 y CC en 0, 10, 20 y 30. En ese rango T1 se comporta de forma errática, sin patrón reconocible, mientras que T3 crece de forma ligeramente ordenada con CP. Por eso el umbral se decidió atendiendo solo a T3. La columna de CC = 0 fue la única factible en todo su recorrido y reunió además los valores más bajos, con CP = 19 todavía por debajo del umbral en 1,4376%.



5. Conclusiones

El recorrido completo, del caso base a los barridos finos, permite cerrar el intervalo de trabajo. Se recomienda, con CO = 1, CC = 0 y CP por debajo de 17.

En la columna de CC = 0 todos los casos siguen siendo factibles hasta CP = 19, de modo que el corte en CP < 17 actúa como margen de seguridad frente al umbral, no como límite de factibilidad. CO se fijó en 1, como se ha mencionado anteriormente, porque lo relevante es la relación entre coeficientes y no su valor absoluto. Anular CC responde a que con XS = 0 no hay asignación de referencia previa, el coste por cambio pierde su significado y CP ya penaliza la selección ponderada por importancia. Con esto quedan cubiertos los objetivos iniciales.

Como línea de trabajo futura, resultaría interesante implementar otras formulaciones de escalarización, en particular la técnica ε-constraint.

6. Referencias

[1] K. Miettinen, Nonlinear Multiobjective Optimization, Boston: Kluwer Academic Publishers, 1998.

[2] P. Halffmann, L. Schäfer, K. Dächert, K. Klamroth y S. Ruzika, «Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey,» Journal of Multi-Criteria Decision Analysis, pp. 341-363, 2022.

[3] R. T. Marler y J. S. Arora, «The weighted sum method for multi-objective optimization: New insights,» Structural and Multidisciplinary Optimization, vol. 41, nº 6, pp. 853-862, 2010.

[4] J. R. Figueira, C. Fonseca, P. Halffmann, K. Klamroth, L. Paquete, S. Ruzika, B. Schulze y M. Stiglmayr, «Easy to say they are Hard, but Hard to see they are Easy— Towards a Categorization of Tractable Multiobjective Combinatorial Optimization Problems,» Journal of Multi-Criteria Decision Analysis, vol. 24, pp. 82-98, 2017.

[5] F. Y. Edgeworth, «Mathematical psychics,» Mind, vol. 6, nº 24, pp. 581-583, 1881.

[6] V. Pareto, Cours d'Économie Politique, Lausanne: F. Rouge, 1896.

[7] M. Ehrgott, «A discussion of scalarization techniques for multiple objective integer programming,» Annals of Operations Research, vol. 147, nº 1, pp. 343-360, 2006.

[8] A. M. Geoffrion, «Proper Efficiency and the Theory of Vector Maximization,» Journal of Mathematical Analysis and Applications, vol. 22, nº 3, pp. 618-630, 1968.

ÍNDICE DE LA MEMORIA

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

La estabilidad de frecuencia es una condición esencial para la operación segura de un sistema eléctrico. Cuando una perturbación grave desequilibra la generación y la demanda, los esquemas de deslastre de carga por baja frecuencia desconectan parte del consumo de forma controlada para evitar el colapso de la red. Son la última línea de defensa. La eficacia de estos esquemas depende de elegir bien qué cargas son desconectadas. Esa elección se ha complicado o nunca llegó a ser efectiva del todo. La integración de energías renovables y de generación distribuida ha aumentado la variabilidad y la incertidumbre en los flujos de carga de los alimentadores asignado al Esquema de Deslastre por Baj Frecuencia (EDCF). Esto ha producido una bajada en el rendimiento de los métodos de selección tradicionales.

Este trabajo aborda ese problema y propone mejorar la selección de los alimentadores a deslastrar para que el deslastre real se ajuste mejor al objetivo de diseño.

La energía eléctrica sostiene prácticamente toda la actividad de las sociedades modernas. Desde el suministro doméstico hasta la operación de hospitales, sistemas de transporte e infraestructuras digitales. Su disponibilidad continua se da por supuesta y la sociedad es completamente dependiente de ella. Esa continuidad descansa sobre un equilibrio físico permanente entre la potencia generada y la consumida, que se refleja en la frecuencia del sistema. Mantener esa frecuencia dentro de un margen determinado es la condición que permite que el conjunto de la red opere de forma síncrona y estable.

Sostener ese equilibrio es cada vez más exigente. La transición energética ha desplazado el modelo tradicional, basado en grandes centrales síncronas, hacia uno con una elevada penetración de generación renovable y distribuida. Estas fuentes aportan beneficios ambientales evidentes, pero introducen una variabilidad e incertidumbre considerables en los flujos de potencia. La red se vuelve más limpia y, al mismo tiempo, más difícil de

controlar. Las herramientas de protección diseñadas para un sistema más predecible empiezan a mostrar sus límites y deficiencias.

Cuando ese equilibrio se rompe de forma brusca, las consecuencias son inmediatas y costosas. El ejemplo más reciente tuvo lugar en España en 2025. Un apagón generalizado detiene la actividad económica y compromete servicios críticos como hospitales, transporte o comunicaciones. Para evitar ese extremo, los sistemas eléctricos cuentan con los EDCF, que desconectan consumo de manera controlada como último recurso ante una caída de frecuencia. Perturbaciones recientes han evidenciado que la carga realmente deslastrada puede alejarse de la prevista en el diseño, lo que reduce el margen de seguridad justo en el momento más crítico.

Buena parte de ese desajuste se origina en cómo se seleccionan los alimentadores que componen cada escalón del esquema. Decidir qué cargas desconectar, y en qué orden, determina si el deslastre real cumple su objetivo. En un sistema con generación distribuida y perfiles de demanda volátiles, los procedimientos manuales habituales tienen dificultades para capturar esa dinámica. Mejorar esta selección refuerza la resiliencia de la red y hace más segura la integración de renovables, con un efecto directo sobre la fiabilidad del suministro del que depende la actividad económica y social.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Python

El desarrollo del trabajo se ha apoyado en Python, un lenguaje de programación de propósito general muy extendido en la comunidad científica por su versatilidad y por la amplitud de su ecosistema de bibliotecas. Sobre él se articulan el resto de las herramientas empleadas.

Pyomo

Pyomo es una librería de Python orientada al modelado de problemas de optimización. Permite formular el problema directamente en código, con una sintaxis muy próxima a la notación algebraica habitual en optimización matemática. Así, los conjuntos, las variables, las restricciones y la función objetivo se definen de forma legible y cercana a su expresión teórica, lo que facilita traducir el modelo de selección de alimentadores a una implementación ejecutable sin perder de vista su formulación original. Pyomo actúa como capa de modelado y delega la resolución en solvers externos, con los que se conecta a través de una interfaz común. Admite un amplio conjunto de ellos, entre ellos Gurobi.

Gurobi

Un solver es el componente encargado de resolver el problema matemático: a partir del modelo formulado, busca los valores de las variables que cumplen todas las restricciones y optimizan la función objetivo. Gurobi es uno de los solvers comerciales de referencia para problemas de programación lineal y lineal entera mixta. Destaca por su rendimiento y por la rapidez con que alcanza soluciones óptimas en problemas de gran tamaño. Se utiliza ampliamente en la industria y en la investigación a nivel mundial. Dispone de licencias académicas las cuales han permitido su uso en un proyecto como este.

pandas, NumPy, Matplotlib y seaborn.

El tratamiento de los datos y el análisis de los resultados se han apoyado en varias bibliotecas complementarias. pandas y NumPy se han empleado para cargar, limpiar y manipular las series de datos históricos de potencia. Matplotlib y seaborn se han utilizado para representar gráficamente los resultados. Esto facilita interpretar el comportamiento del modelo y comparar los distintos casos estudiados.

Excel y MATLAB

De forma complementaria, se han utilizado Excel y MATLAB en el análisis de los resultados y en la elaboración de tablas y gráficas, aprovechando su comodidad para organizar datos y generar representaciones visuales de apoyo.

Claude Code.

Por último, se ha empleado Claude Code como herramienta de asistencia en el desarrollo. Su uso se ha limitado a apoyar la modificación del código en Python durante la prueba de los distintos casos, lo que agilizó los cambios necesarios para evaluar cada configuración.

Capítulo 3. ESTADO DE LA CUESTIÓN

Introducción a los problemas multiobjetivo

La optimización multiobjetivo se ocupa de problemas en los que se pretende optimizar de forma simultánea más de una función objetivo [1]. A diferencia del caso de un solo objetivo, donde la noción de óptimo es única y existe un valor de referencia claro, aquí intervienen varias funciones que representan criterios distintos y que el decisor quiere minimizar (o maximizar) a la vez [2].

Un problema de este tipo, con $k \geq 2$ objetivos, puede expresarse de forma compacta como:

$$\min f(x) = (f_1(x), f_2(x), \dots, f_k(x))$$

$$x \in X$$

donde cada f_i recoge uno de los criterios considerados y X es el conjunto de soluciones admisibles o región factible [1], [2].

X que aparece en la formulación anterior (1) se denomina región factible y está formado por todos los puntos del espacio de decisión que satisfacen las restricciones del problema [1]. Para un problema con restricciones lineales, la región factible adopta la forma $X = \{x : Ax \leq b\}$, donde A y b recogen los coeficientes y los términos independientes de las restricciones [2]. Cualquier solución que cumpla estas condiciones se considera admisible, y la búsqueda del óptimo se restringe a este conjunto.

Si los objetivos no estuvieran en conflicto, existiría una solución capaz de alcanzar el mínimo de todas las funciones a la vez, y el problema se reduciría en la práctica a resolver k problemas por separado. Esa situación es excepcional. Lo habitual es que la mejora de un criterio solo se consiga a costa de empeorar otro, por lo que no existe una solución que sea óptima para todos los objetivos de manera conjunta [1]. El problema abordado en este

trabajo presenta este conflicto. En el problema presentado en este trabajo existen tres funciones objetivo que buscan minimizar distintos objetivos. La primera minimiza el número de alimentadores seleccionados, condicionado por un grado de importancia asignado a cada uno. La segunda minimiza los cambios de alimentadores. La tercera minimiza la variación del deslastre de carga con respecto a un objetivo. Según la lógica del funcionamiento de la red, para minimizar el tercer objetivo se requiere la mayor libertad de cambio y asignación de alimentadores, lo que entra en conflicto con los dos primeros objetivos.

A esta dificultad se suma que las funciones objetivo suelen estar expresadas en unidades diferentes y con órdenes de magnitud que no son comparables. Son las funciones inconmensurables [1]. Es el caso del problema de este trabajo. La presencia de objetivos en conflicto, junto con la heterogeneidad de las unidades, impide ordenar las soluciones mediante una comparación directa y obliga a recurrir a métodos que transformen el problema vectorial en una o varias formulaciones escalares tratables [3], [4].

Dado que no existe una solución que minimice todas las funciones objetivo a la vez, la noción de óptimo debe redefinirse. En optimización multiobjetivo se recurre al concepto de dominancia, cuyo origen se remonta a los trabajos de Edgeworth [5] y Pareto [6]. La dominancia permite comparar soluciones entre sí sin necesidad de agregar los objetivos en un único valor.

Considérese un problema de minimización, como el que se aborda en este trabajo, con k objetivos. Dadas dos soluciones factibles $x^1, x^2 \in X$ con imágenes $f(x^1)$y $f(x^2)$, se dice que x^1domina a x^2 si su imagen es al menos tan buena en todos los objetivos y estrictamente mejor en al menos uno:

$$f_i(x^1) \leq f_i(x^2) \forall i \in \{1, \dots, k\}, f_j(x^1) < f_j(x^2) \text{ para algún } j$$

Una solución factible se denomina eficiente, o Pareto-óptima, cuando no existe ninguna otra solución factible que la domine [1], [2]. Su imagen en el espacio de objetivos recibe el nombre de imagen no dominada.

Conviene distinguir dos espacios. En el espacio de decisión, el conjunto de todas las soluciones eficientes se denota por X_E. En el espacio de objetivos, el conjunto de sus imágenes se denomina conjunto no dominado, o frente de Pareto, y se denota por Y_N [1], [2]. Cada punto de este frente representa un compromiso entre los criterios. Esto es, que no es posible mejorar uno de los objetivos sin empeorar al menos otro.

Junto a la eficiencia se define una condición más débil. Una solución es débilmente eficiente si no existe otra solución factible que la mejore de forma estricta en todos los objetivos simultáneamente [1], [2]. Toda solución eficiente es débilmente eficiente, aunque al revés no se cumple. Esta distinción resulta relevante al analizar los métodos de resolución, ya que algunos de ellos solo garantizan soluciones débilmente eficientes [7].

La consecuencia práctica es que la resolución de un problema multiobjetivo no proporciona una única solución, sino un conjunto de soluciones de compromiso [1]. Elegir una entre ellas exige incorporar información adicional sobre las preferencias del decisor, o bien transformar el problema vectorial en una formulación escalar que pueda resolverse con técnicas convencionales [7], [3].

Para caracterizar el frente de Pareto resulta útil acotar el rango de valores que toman las funciones objetivo sobre el conjunto no dominado. Esa acotación se establece mediante dos puntos de referencia, el punto ideal y el punto nadir [1].

El punto ideal y^I se obtiene minimizando cada función objetivo por separado sobre la región factible. Su componente i-ésima se define como:

$$y_i^I := \min\{f_i(x) : x \in X\}$$

El punto ideal representa, por tanto, el mejor valor alcanzable para cada criterio considerado de forma aislada [1], [2]. Salvo en el caso excepcional de objetivos no conflictivos, el punto ideal no es alcanzable por ninguna solución factible, ya que reuniría en un único punto los óptimos individuales de criterios que entran en conflicto. Funciona como una cota inferior del frente de Pareto.

El punto nadir y^N recoge, por el contrario, los peores valores que toma cada objetivo dentro del conjunto de soluciones eficientes. Su componente i-ésima se define sobre el conjunto no dominado:

$$y_i^N := \max\{f_i(x) : x \in Y_N\}$$

El punto nadir acota superiormente el rango de los objetivos a lo largo del frente de Pareto [1], [2]. Su cálculo no es inmediato. Para dos objetivos coincide con el máximo de cada componente entre las imágenes lexicográficamente óptimas, pero a partir de tres objetivos no se conoce ningún algoritmo eficiente para determinarlo de forma exacta [2].

Ambos puntos delimitan el rango de cada función objetivo sobre el frente, lo que ofrece una referencia para normalizar criterios expresados en unidades distintas [1], [2]. Esta normalización es necesaria cuando los objetivos no son directamente comparables [3]. Esta situación se da en el problema abordado en este trabajo.

La resolución de un problema multiobjetivo es, en general, más costosa que la de su análogo de un solo objetivo. Esta dificultad tiene dos orígenes distintos que conviene separar.

El primero se refiere al tamaño del conjunto solución. El número de puntos no dominados puede crecer de forma exponencial con el tamaño de la instancia del problema. Cuando esto ocurre, se dice que el problema es intratable [4]. Aun cuando cada solución pudiera calcularse de forma eficiente, el mero hecho de enumerar un frente de Pareto de tamaño exponencial impediría resolver el problema en tiempo polinómico.

El segundo origen tiene que ver con la naturaleza de las soluciones eficientes. No todas se obtienen con el mismo esfuerzo. Una solución eficiente se denomina soportada cuando puede alcanzarse minimizando una suma ponderada de los objetivos, es decir, cuando su imagen se sitúa en la frontera de la envolvente convexa del conjunto no dominado. El resto se denominan soluciones no soportadas [4], [7]. El cálculo de estas últimas resulta considerablemente más exigente [4]. En consecuencia, un problema combinatorio multiobjetivo solo admite resolución en tiempo polinómico si reúne dos condiciones: que su número de puntos no dominados esté acotado polinómicamente y que sus soluciones no soportadas, en caso de existir, puedan calcularse de forma eficiente [4].

A estas dificultades se añade la presencia de variables enteras. La formulación abordada en este trabajo es un problema lineal entero mixto (MILP). La existencia de variables discretas complica el proceso de resolución respecto al caso puramente continuo, ya que aparecen puntos no dominados no soportados y la estructura del frente deja de ser convexa [2]. Este es uno de los motivos por los que los algoritmos para problemas multiobjetivo con variables enteras han recibido una atención creciente en la investigación reciente [2].

El conjunto de estas observaciones justifica el recurso a métodos que transformen el problema vectorial en una o varias formulaciones escalares, resolubles con técnicas de optimización de un solo objetivo convencionales. Estos métodos de escalarización constituyen el objeto de la revisión que se presenta en la siguiente sección [2], [7].

Métodos de resolución

La revisión de Halffmann et al. [2] recopila los enfoques algorítmicos disponibles para problemas de optimización lineal multiobjetivo con variables enteras y enteras mixtas. Su alcance se limita a los métodos exactos, esto es, aquellos que obtienen el conjunto completo de imágenes no dominadas, y deja fuera las metaheurísticas, los algoritmos evolutivos y los métodos de aproximación o interactivos [2]. El trabajo cataloga más de cien artículos publicados hasta 2021 y se plantea como una referencia del estado del arte en esta categoría de problemas. Para organizar tal volumen de literatura, los algoritmos se

dividen primero según el tipo de problema, entero puro o entero mixto, y a continuación en subgrupos definidos por el principio de resolución que comparten [2].

Antes de presentar los algoritmos, la revisión introduce las técnicas de escalarización más habituales, que constituyen la base de buena parte de ellos [2]. Una escalarización transforma el problema vectorial en uno de un solo objetivo que se resuelve de forma repetida con distintos parámetros para recuperar las soluciones eficientes [7]. Las principales son el método de suma ponderada (weighted sum), el método ε-constraint, el método híbrido, el método de Benson, el método de Tchebycheff ponderado y el método de Pascoletti-Serafini [2]. Entre ellas, el método de suma ponderada es el de planteamiento más sencillo: minimiza una combinación lineal de los objetivos y, cuando los pesos son estrictamente positivos, garantiza una solución eficiente del problema original [2], [8]. Este es el método adoptado en el presente trabajo.

Para el caso entero puro (MOILP), la revisión identifica nueve familias de algoritmos [2]. Las técnicas de ramificación y acotación (branch-and-bound y branch-and-cut) figuran entre las primeras en aparecer y siguen siendo un área activa. Los métodos basados en la escalarización ε-constraint constituyen la familia más citada, debido a su simplicidad y velocidad. Los métodos de descomposición del espacio de objetivos (image space decomposition) han ganado protagonismo recientemente, al explotar propiedades geométricas para descartar regiones del espacio de imágenes. El resto de familias agrupa los métodos basados en normas, los métodos en dos fases, la programación dinámica, la programación disyuntiva, los métodos algebraicos y un grupo final de aportaciones diversas [2].

Para el caso entero mixto (MOMILP), la literatura es más escasa y reciente, ya que el primer algoritmo no se propuso hasta 1998 [2]. La presencia de variables continuas junto a las discretas obliga a tratar con partes no dominadas de distinta dimensión, lo que complica la comparación de soluciones. La revisión agrupa estos algoritmos en tres familias: los de ramificación y acotación, los que operan en el espacio de objetivos y los métodos híbridos, que combinan recursos de los dos anteriores [2].

La revisión de Halffmann et al. [2] introduce seis técnicas de escalarización que sirven de base a la mayoría de los algoritmos exactos. Todas transforman el problema vectorial en uno monoobjetivo, pero difieren en las garantías que ofrecen sobre las soluciones obtenidas y en el esfuerzo computacional que requieren. La Tabla 1 resume sus características.

Método	***Descripción***	***Ventajas***	***Desventajas***
Suma ponderada (weighted sum) [2], [7], [8]	Minimiza una combinación lineal de los objetivos mediante un vector de pesos λ. El conjunto factible no se modifica.	Planteamiento sencillo. Con $\lambda > 0$ la solución óptima es eficiente [8]. Conserva la misma complejidad y esfuerzo de cálculo que la versión monoobjetivo del problema [7].	Solo obtiene soluciones soportadas. Las soluciones eficientes situadas en el interior de la envolvente convexa no pueden alcanzarse, lo que deja inaccesible parte del frente cuando este no es convexo [7].
ε-constraint [2], [7]	Minimiza uno de los objetivos y convierte los demás en restricciones con cotas superiores ε.	No agrega los objetivos. Con una elección adecuada de ε puede encontrar todas las soluciones eficientes, incluidas las no soportadas [7].	La solución óptima es solo débilmente eficiente salvo que sea única [2], [7]. Las cotas son restricciones de tipo presupuestario, por lo que la escalarización suele ser NP-hard [7].
Híbrido (hybrid) [2]	Combina los dos anteriores: minimiza una suma ponderada sujeta a	Con $\lambda > 0$ garantiza eficiencia y es capaz de generar todas las soluciones eficientes	Hereda las restricciones tipo presupuestario del ε-constraint, con su coste computacional. Mayor

	restricciones sobre todos los objetivos.	[2].	número de parámetros que ajustar.
Benson [2]	Parte de una solución dada y determina si es eficiente; si no lo es, devuelve una solución eficiente.	Verifica la eficiencia de una solución y produce una eficiente a partir de una que no lo es [2].	Orientado a comprobar o mejorar soluciones individuales más que a generar directamente el frente completo.
Tchebycheff ponderado (weighted Tchebycheff) [2]	Minimiza la máxima desviación ponderada respecto al punto ideal. Es un método de punto de referencia.	A diferencia de la suma ponderada, alcanza soluciones no soportadas, también con frentes no convexos. Las variantes aumentada o lexicográfica devuelven soluciones eficientes [2].	En su forma básica la solución es solo débilmente eficiente [2]. Requiere el punto ideal y el ajuste de los parámetros de la variante aumentada.
Pascoletti-Serafini [2]	Método general de punto de referencia definido por un punto ay una dirección r. Engloba varios de los anteriores como casos particulares.	Formulación muy general. La unicidad de la solución óptima asegura la eficiencia [2].	En general devuelve soluciones débilmente eficientes [2]. Exige fijar tanto el punto de referencia como la dirección.

Tabla 1. Comparación de los distintos métodos de escalarización

De la comparación se desprende una tensión recurrente. Los métodos que garantizan la obtención del frente completo, incluidas las soluciones no soportadas, lo hacen a costa de introducir restricciones que elevan la dificultad de la escalarización [7]. La suma ponderada se sitúa en el extremo opuesto. Esta renuncia a capturar las soluciones no soportadas a cambio de conservar la complejidad del problema de un solo objetivo original [7].

Método de la suma ponderada

El método de la suma ponderada transforma el problema multiobjetivo en uno monoobjetivo minimizando una combinación lineal de las funciones objetivo, ponderadas por un vector de pesos λ:

$$\min_{x \in X} \sum_{i=1}^{k} \lambda_i \, f_i(x)$$

El conjunto factible X no se modifica, lo que distingue a esta técnica de las que introducen restricciones adicionales [2], [7]. El método fue introducido por Zadeh y es una de las escalarizaciones más antiguas y utilizadas [2], [3].

Los pesos suelen interpretarse como una medida de la importancia relativa de cada objetivo [3]. Esta interpretación, sin embargo, exige una precisión que con frecuencia se pasa por alto. La solución Pareto-óptima que devuelve el método no depende solo de los pesos, sino también de la magnitud relativa de las funciones objetivo. Geométricamente, el vector de pesos define el gradiente de la función agregada, y la solución es el punto del frente de Pareto donde el contorno de menor valor de esa función lo toca. Para un conjunto fijo de pesos, escalar una función objetivo desplaza ese punto y, por tanto, cambia la solución obtenida [3]. La consecuencia práctica es que el valor de un peso solo es significativo en relación con los demás pesos y con el rango de valores de su función objetivo asociada [3].

De lo anterior se deduce que, cuando las funciones objetivo presentan rangos o unidades dispares, fijar los pesos según la importancia de cada criterio obliga a normalizar previamente las funciones para que tengan rangos comparables [3]. Si no se hace, los pesos acaban recogiendo las diferencias de magnitud en lugar de la importancia relativa pretendida. Esta situación se presenta en el problema abordado en este trabajo, donde los términos CP, CC y CO miden magnitudes de distinta naturaleza.

La propiedad teórica que sostiene el uso del método procede de Geoffrion [8]. Si todos los pesos son estrictamente positivos, la solución óptima de la suma ponderada es una solución eficiente del problema original. Con pesos no negativos, la garantía se debilita y la solución es solo débilmente eficiente [2]. Esta condición ofrece un criterio sencillo para asegurar que la solución obtenida pertenece al frente de Pareto.

El método presenta dos limitaciones bien documentadas cuando se emplea para representar el frente de Pareto completo. La primera es que solo obtiene soluciones soportadas: las soluciones eficientes situadas en porciones no convexas del frente no pueden alcanzarse con ninguna combinación de pesos [7], [3]. La segunda es que una variación uniforme de los pesos no produce una distribución uniforme de puntos sobre el frente, por lo que las soluciones tienden a agruparse [3]. A un nivel más fundamental, la suma ponderada constituye únicamente una aproximación lineal de la función de preferencia del decisor, de modo que la solución puede no reflejar las preferencias iniciales por mucho que se ajusten los pesos [3]. Por estos motivos, Marler y Arora recomiendan considerar métodos alternativos cuando es crítico representar de forma completa el frente de Pareto en problemas no convexos [3].

La adopción del método de la suma ponderada en este trabajo responde a dos motivos.

El primero es su sencillez. La suma ponderada es la más directa de las técnicas de escalarización, minimiza una combinación lineal de los objetivos sin alterar el conjunto factible ni introducir variables o restricciones adicionales [2]. Esta sencillez tiene una consecuencia computacional relevante, ya que el problema escalarizado conserva la misma complejidad y exige el mismo esfuerzo de resolución que la versión de un solo objetivo del

problema original [7]. Un planteamiento simple reduce el coste de implementación y de cálculo, y no implica por ello una solución de menor calidad. De hecho, el método sigue siendo uno de los más empleados en la práctica precisamente por su facilidad de uso [3]. A esta ventaja se añade la garantía de eficiencia: con pesos estrictamente positivos, la solución obtenida pertenece al frente de Pareto [8].

El segundo motivo procede de la naturaleza del problema abordado. La principal limitación de la suma ponderada, su incapacidad para capturar el frente de Pareto completo cuando este no es convexo, solo resulta determinante cuando el objetivo es representar el conjunto entero de soluciones de compromiso, es decir, en una articulación de preferencias a posteriori [3]. No es este el caso. En el problema considerado no se pretende generar el frente completo, sino obtener una única solución que refleje las prioridades del decisor, lo que corresponde a una articulación de preferencias a priori, en la cual el decisor busca un punto que satisfaga sus requisitos predefinidos en lugar de explorar toda la frontera [3]. Esta diferencia es la que hace que la limitación señalada no afecte al planteamiento.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

El valor de este proyecto parte de un hecho medible. Existe una brecha entre el deslastre de carga que un esquema EDCF tiene diseñado y el que realmente ejecuta durante una perturbación, y esa brecha tiene un precio. La desviación entre el deslastre real y el objetivo compromete la estabilidad de frecuencia del sistema de potencia. Un deslastre insuficiente puede desestabilizar la red. Un método que reduzca ese desvío mejora directamente un indicador que el operador del sistema valora y por el que está dispuesto a pagar.

El segundo argumento es defensivo. Un apagón generalizado interrumpe el suministro, pero su impacto va mucho más allá. Supone pérdidas directas por la energía no vendida, posibles penalizaciones regulatorias y un elevado coste de oportunidad para las empresas de generación y distribución. Reforzar la protección frente a caídas de frecuencia reduce la probabilidad y el alcance de estos escenarios. La inversión en un sistema de deslastre más robusto se comporta como un seguro: un gasto contenido que evita pérdidas mucho mayores.

A esto se suma una barrera de adopción muy baja. La propuesta se sustenta íntegramente en software (Python y el solver Gurobi) y en la parametrización de relés de baja frecuencia. El modelo es además escalable a redes de distinto tamaño, lo que permite aplicarlo a sistemas pequeños o a redes complejas sin rehacer la metodología. La relación entre coste de implantación y beneficio esperado es favorable, y se vuelve más atractiva a medida que avanza la integración de generación renovable y distribuida, que es justo el escenario donde los métodos tradicionales pierden eficacia.

4.2 OBJETIVOS

Los objetivos de este trabajo son:

- Revisar la literatura sobre problemas de optimización multiobjetivo (especialmente MILP).
- Identificar y justificar los términos que deben incluirse en la función objetivo (y su interpretación práctica).
- Implementar y evaluar el enfoque de suma ponderada (weighted sum), ajustando pesos/multiplicadores y estudiando la sensibilidad de los resultados.
- Analizar los resultados obtenidos y extraer conclusiones/recomendaciones sobre cómo parametrizar la función objetivo.

4.3 METODOLOGÍA

La metodología seguida en este trabajo es de carácter iterativo y se apoya en datos reales. El punto de partida es un conjunto de datos históricos anonimizados, correspondientes a una serie de alimentadores de la Comunidad de Madrid. Sobre esos datos se formula la función objetivo del modelo y se asigna un valor inicial a los coeficientes de coste CP, CC y CO. El problema resultante se resuelve con el optimizador Gurobi, que devuelve la selección de alimentadores y el deslastre asociado para cada caso evaluado.

A partir de ahí el proceso se vuelve cíclico. Cada ejecución genera una serie de casos en los que se observa el deslastre obtenido en el sexto escalón del esquema, que es el objeto de estudio. Estos resultados se analizan para valorar en qué medida la solución se ajusta al comportamiento esperado de un sistema EDCF. Según ese análisis se eligen nuevos valores de CP, CC y CO y se repite la ejecución.

El ciclo se reitera hasta acotar un rango de coeficientes apropiado. La meta no es un único valor, sino un intervalo de CP, CC y CO que resulte viable y produzca un funcionamiento

normal y coherente del esquema de deslastre. Ese rango es el que se discute en los capítulos posteriores.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

5.1 ANÁLISIS DEL SISTEMA

El problema que aborda este trabajo es la selección de alimentadores en los esquemas de deslastre de carga por baja frecuencia (EDCF). Estos esquemas reparten el deslastre total en una secuencia de escalones de frecuencia, y a cada escalón se le asigna un conjunto de alimentadores cuya desconexión debe aportar una potencia determinada. El estudio se centra en el último escalón, el sexto, que cierra la secuencia y debe alcanzar el deslastre acumulado de diseño.

La magnitud principal que se busca ajustar es la desviación entre el deslastre realmente ejecutado y el objetivo de diseño del escalón. Un deslastre por defecto deja la frecuencia sin la corrección necesaria. Uno por exceso desconecta más consumo del previsto y puede provocar sobrefrecuencia. Junto a este objetivo, la selección atiende a dos criterios adicionales. El primero es priorizar qué tipo de alimentador se desconecta (industrial, servicios públicos, rural o doméstico), para proteger los consumos más sensibles. El segundo es minimizar el número de cambios en la asignación, porque incluir o mover un alimentador dentro del esquema tiene un coste operativo: obliga a configurar la asociación entre el alimentador y el relé de frecuencia en la red de distribución, y a desconectar ese consumo cuando actúe el escalón. Estos cambios se miden respecto a una asignación de referencia, que en todo este trabajo se fija en XS = 0; es decir, se parte de un esquema sin alimentadores preasignados.

El punto de partida son datos históricos reales y anonimizados en sistemas de distribución de España. Para cada alimentador se conoce su potencia a lo largo del tiempo, con resolución horaria, junto con la potencia total del sistema en cada instante. Estos datos reflejan la variabilidad real de la demanda durante el año y son la base sobre la que se construye y se valida el modelo. Tradicionalmente, esta selección se ha hecho de forma

manual y sobre unos pocos escenarios de punta o valle, un enfoque que no recoge esa variabilidad y tiende a desviarse del objetivo en el resto de las condiciones.

De este planteamiento se derivan los requisitos del sistema. El principal es aproximar el deslastre objetivo del sexto escalón minimizando la desviación, tanto por exceso como por defecto. A esto se añade la prioridad por tipo de consumo, que reserva los alimentadores más sensibles para los escalones en los que es admisible desconectarlos, y la penalización de las reasignaciones innecesarias frente a la configuración de referencia. Por último, la solución debe ser computacionalmente viable y escalable a redes de distinto tamaño, de modo que pueda aplicarse sobre los datos reales de las comunidades estudiadas.

5.2 DISEÑO

Este apartado introduce el modelo matemático diseñado para solucionar el problema. Se explicarán sus variables, parámetros y funcionamiento general.

La función objetivo es la siguiente:

$$\min_{s} \frac{CP}{|\mathcal{I}|}\sum_{i} F_i \cdot x_i + \frac{CC}{|\mathcal{I}|}\sum_{i}(n_i^{\uparrow} + n_i^{\downarrow}) + \frac{CO}{LS \cdot |\mathcal{T}|}\sum_{t}(s_t^{\uparrow} + s_t^{\downarrow}) + 100 \qquad (1)$$

Restricciones:

$$n_i^{\uparrow} + n_i^{\downarrow} = x_i - XS_i \quad \forall i \in \mathcal{I} \quad (2)$$

$$n_i^{\uparrow} + n_i^{\downarrow} \leq 1 \quad \forall i \in \mathcal{I} \quad (3)$$

$$x_i + XS_i \leq 1 \quad \forall i \in \mathcal{I} \quad (4)$$

Índices:

- $i = \{1, \ldots, |\mathcal{I}|\}$: set de alimentadores
- $t = \{1, \ldots, |\mathcal{T}|\}$: set de estampas de tiempo evaluadas

Parámetros:

- CP: Costo de asignación de alimentadores
- F_i: Matriz de pesos de los alimentadores
- CC : Costo por cambio de asignación de alimentadores
- CO : Costo operativo
- $|\mathcal{T}|$: Número de estampas de tiempo evaluadas
- LS : Requerimiento de deslastre de carga del escalón evaluado

Variables:

- x_i : Variable binaria {0,1} utilizada para asignar alimentadores. Su valor será 1 si el alimentador i es elegido. Será 0 si dicho alimentador no es elegido.
- $s_t^{\uparrow}, s_t^{\downarrow}$: Variable de holgura; cuantifica el desvío del deslastre elegido respecto al deslastre objetivo.
- $n_i^{\uparrow}, n_i^{\downarrow}$: Variable que cuantifica el cambio en la elección de un alimentador.

$E(s_t^{\downarrow})$ es el valor esperado o valor medio de la desviación negativa respecto del deslastre objetivo. $E(s_t^{\uparrow})$ es lo mismo, pero la desviación positiva. Estos valores se mostrarán como un valor porcentual del objetivo.

Uno de los principales objetivos a la hora de obtener los coeficientes es minimizar la suma de las variables de holgura. Esta se presenta como $E(|s_t^{\uparrow}| + |s_t^{\downarrow}|)$.

Los alimentadores sujetos a una posible desconexión se dividen en 4 categorías distintas. A cada alimentador se le asigna un grado de criticidad distinto. A mayor importancia, se priorizará que ese alimentador no se desconecte durante el proceso de deslastre. Las categorías son las siguientes, ordenadas de mayor a menor prioridad:

1. Servicios Públicos (S.S.P.P).
2. Doméstico
3. Rural

4. Industrial

Existen, además, alimentadores sin tipo de consumo asignado, identificados como N/A, que quedan fuera de esta clasificación de prioridad.

En este capítulo se presentan tres términos distintos. Estos términos, denominados T1, T2 y T3, son aquello que multiplica a cada uno de los coeficientes de la función objetivo. Por tanto, tendrán la siguiente forma:

- $T1 = \frac{1}{|\mathcal{I}|}\sum_i F_i \cdot x_i$ (5)
- $T2 = \frac{1}{|\mathcal{I}|}\sum_i (n_i^{\uparrow} + n_i^{\downarrow})$ (6)
- $T3 = \frac{1}{LS \cdot |\mathcal{T}|}\sum_t (s_t^{\uparrow} + s_t^{\downarrow})$ (7)

5.3 IMPLEMENTACIÓN

La función objetivo y el conjunto de datos descritos se han implementado en Python. El código carga los registros históricos de potencia de cada alimentador y de la potencia total del sistema, y separa dos conjuntos: los escenarios representativos obtenidos por agregación ponderada de series temporales, que actúan como entrada del optimizador, y un conjunto de validación reservado para comprobar el comportamiento del modelo sobre datos no empleados en el ajuste. A partir de estos datos se construye la matriz de potencias por alimentador que alimenta la función objetivo.

El modelo se formula con Pyomo. Cada índice, parámetro, variable y restricción de la sección anterior se traduce a una expresión equivalente en código, lo que permite mantener la formulación matemática y su implementación próximas entre sí. Una vez construido el modelo para un escalón concreto, con sus coeficientes CP, CC y CO y su requerimiento de deslastre, el problema se entrega a Gurobi, que resuelve el problema lineal entero mixto y devuelve la selección de alimentadores y el deslastre asociado.

El código está organizado para repetir este proceso de forma sistemática. Recorre los distintos casos definidos por los valores de los coeficientes y, dentro de cada caso, resuelve el modelo para el escalón en estudio. Esta estructura es la que da soporte a la metodología iterativa: cada ejecución produce los resultados con los que se evalúa el efecto de los coeficientes y se decide cómo ajustarlos en la siguiente vuelta. Este código se muestra en el Anexo I.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este apartado se expondrán los resultados obtenidos en las pruebas hechas para encontrar los coeficientes óptimos.

6.1 CASO BASE

En este apartado se presenta el caso base, el cual servirá como referencia para el resto de los casos. Se busca minimizar el impacto de los coeficientes en la función objetivo, con el fin de evaluar de manera objetiva el impacto de los coeficientes de ponderación en la optimización.

El caso descrito se ajusta a los siguientes criterios técnicos:

1. $CP = CC = 0$. Se anulan los costos de asignación de cada alimentador y los costos por cambio de asignación de alimentador.
2. $CO = 1$. Al multiplicar a la suma de las variables de holgura respecto del requerimiento de deslastre de carga, se anula su efecto igualando el costo operativo a 1.

Valores esperados	*Porcentaje respecto al objetivo*
$E(s_t^{\downarrow})$	-1,27%
$E(s_t^{\uparrow})$	1%
$E(\lvert s_t^{\uparrow}\rvert + \lvert s_t^{\downarrow}\rvert)$	2,27%
T1	0,2298
T2	0,0726
T3	5,8702

Tabla 2. Resultados del Caso Base

6.2 *ANÁLISIS UNIVARIANTE*

El análisis univariante es el punto de partida de las pruebas. Los coeficientes de la función objetivo pueden tomar cualquier valor entre cero e infinito, por lo que conviene empezar por una exploración amplia antes de afinar rangos concretos. Para cubrir varios órdenes de magnitud con pocos casos se recorre cada coeficiente en escala logarítmica. Así se identifica a partir de qué valor empieza a influir en los resultados de la función objetivo y de qué forma lo hace.

El estudio analiza el efecto de cada coeficiente por separado. Se varía un único coeficiente a lo largo de ese rango mientras los otros dos se mantienen fijos, de modo que cualquier cambio observado pueda atribuirse al coeficiente modificado. Se definen dos grupos de casos. Los de tipo A recorren distintos valores de CP con CC anulado, y los de tipo B hacen lo propio con CC y anulando CP. En ambos, CO se mantiene en su valor del caso base. El objetivo es localizar, para cada coeficiente, el rango en el que su variación empieza a alterar la selección de alimentadores y la desviación del deslastre, lo que más adelante se denomina zona de quiebre.

En primer lugar, se analiza el comportamiento del sistema ante la variación del coeficiente CP.

	CP	CC	CO
Caso A1	0,1	0	1
Caso A2	1	0	1
Caso A3	10	0	1
Caso A4	100	0	1
Caso A5	1000	0	1

Tabla 3. Coeficientes para los casos A

	Caso A1	*Caso A2*	*Caso A3*	*Caso A4*	*Caso A5*
$E(s_t^{\downarrow})$	-1,27%	-1,27%	-1,27%	-1,14%	-2,1%
$E(s_t^{\uparrow})$	1%	1%	1%	1,78%	3,4%
$E(\lvert s_t^{\uparrow}\rvert + \lvert s_t^{\downarrow}\rvert)$	2,27%	2,27%	2,27%	2,93%	5,5%
T1	0,1855	0,2419	0,1613	0,1411	0,0685
T2	0,0645	0,0766	0,0605	0,0565	0,0363
T3	5,85	5,9524	7,5315	12,3282	266,977

Tabla 4. Resultados de los casos A

La Tabla 3 reúne los cinco casos de tipo A. Aquí el coeficiente CP va creciendo de diez en diez, desde 0,1 hasta 1000, mientras CC se mantiene en cero y CO en uno. Cinco casos bastan para barrer cuatro órdenes de magnitud, que es precisamente lo que interesa para ver dónde empieza a influir la variación del coeficiente y empieza a notarse en la solución.

Los resultados aparecen en la Tabla 4, y cuentan una historia clara. En los tres primeros casos el sistema apenas se inmuta. La desviación esperada, recogida en E(|s↑|+|s↓|), se queda fija en 2,27%, y con ella el término T3. CP puede multiplicarse por cien entre A1 y A3 sin que el deslastre se aparte de su objetivo. Es a partir de A4 cuando la cosa cambia de verdad: la desviación sube a 2,93%, y en A5 se dispara hasta el 5,5%. T3 acompaña y crece con fuerza, hasta multiplicarse por más de cuarenta entre los extremos del barrido (los casos A1 y A5).

Se puede observar en la Tabla 4 que en los primeros tres casos aumenta T3 sin que cambie el valor esperado. Esto es porque el valor esperado sí que cambia, pero poco, y no se refleja cuando se redondea al segundo decimal. Si se observa el caso A4, donde sí que cambia el valor esperado, este cambia un 0,2% y el T3 cambia de 7,5 a 12,4, 5 puntos más, que es casi un 100% del valor de T3 anterior.

Lo bueno es que la zona de quiebre, ese punto en el que CP deja de ser inofensivo y empieza a influir en los resultados, cae entre 10 y 100 que es justo dentro del rango que se

había elegido. El barrido captura la transición que se quería estudiar, ni se queda corto ni se pasa de largo.

Los resultados encajan con lo que CP representa en la función objetivo (1). Cuanto mayor es, más le pesa al optimizador desconectar alimentadores, hasta el punto de renunciar a algunos que hacían falta para cumplir el objetivo de reducir el desvío con respecto a la función objetivo. Esa renuncia es la que se paga en forma de mayor desviación.

A continuación, se procede a analizar el coeficiente CC.

	CP_i	CC	CO
Caso B1	0	0,1	1
Caso B2	0	1	1
Caso B3	0	10	1
Caso B4	0	100	1
Caso B5	0	1000	1

Tabla 5. Coeficientes para los casos B

	Caso B1	*Caso B2*	*Caso B3*	*Caso B4*	*Caso B5*
$E(s_t^{\downarrow})$	-1,27%	-1,27%	-1,27%	-1,39%	-1,72%
$E(s_t^{\uparrow})$	1%	1%	1%	1,17%	1,5%
$E(\lvert s_t^{\uparrow}\rvert + \lvert s_t^{\downarrow}\rvert)$	2,27%	2,27%	2,27%	2,56%	3,22%
T1	0,2863	0,2218	0,1895	0,1532	0,1371
T2	0,0887	0,0766	0,0605	0,0605	0,0524
T3	5,8639	5,9213	6,0932	8,9026	19,8863

Tabla 6. Resultados de los casos B

En la Tabla 6 se puede observar un patrón similar al comentado previamente sobre la Tabla 4. La zona de quiebre vuelve a situarse aproximadamente entre los mismos valores, entre

10 y 100. La diferencia está en la intensidad del efecto. El crecimiento de T3 al aumentar CC es bastante más suave que el que se produce al variar CP. Basta comparar los casos extremos. En A5, con CP en 1000, T3 alcanza 266,977, mientras que en B5, con CC en ese mismo valor, se queda en 19,8863. CC empuja los resultados en la misma dirección que CP, pero con mucha menos fuerza.

6.3 ANÁLISIS BIVARIANTE

Conocidas las zonas de quiebre de ambos coeficientes se hace un análisis bivariante con el fin de evaluar la sensibilidad de cada coeficiente. Para ello se variaron los coeficientes CC y CP desde 50 hasta 1000 en intervalos de 25. Se hicieron un total de 1521 casos. Con el fin de comprobar la coherencia de los resultados se llevaron a cabo dos análisis.

El primero se centró en comparar la evolución del término T1 a medida que aumentaba el coeficiente CP para un mismo CC. Cada una de las líneas de la Figura 1 es una barrida para un valor de CC.

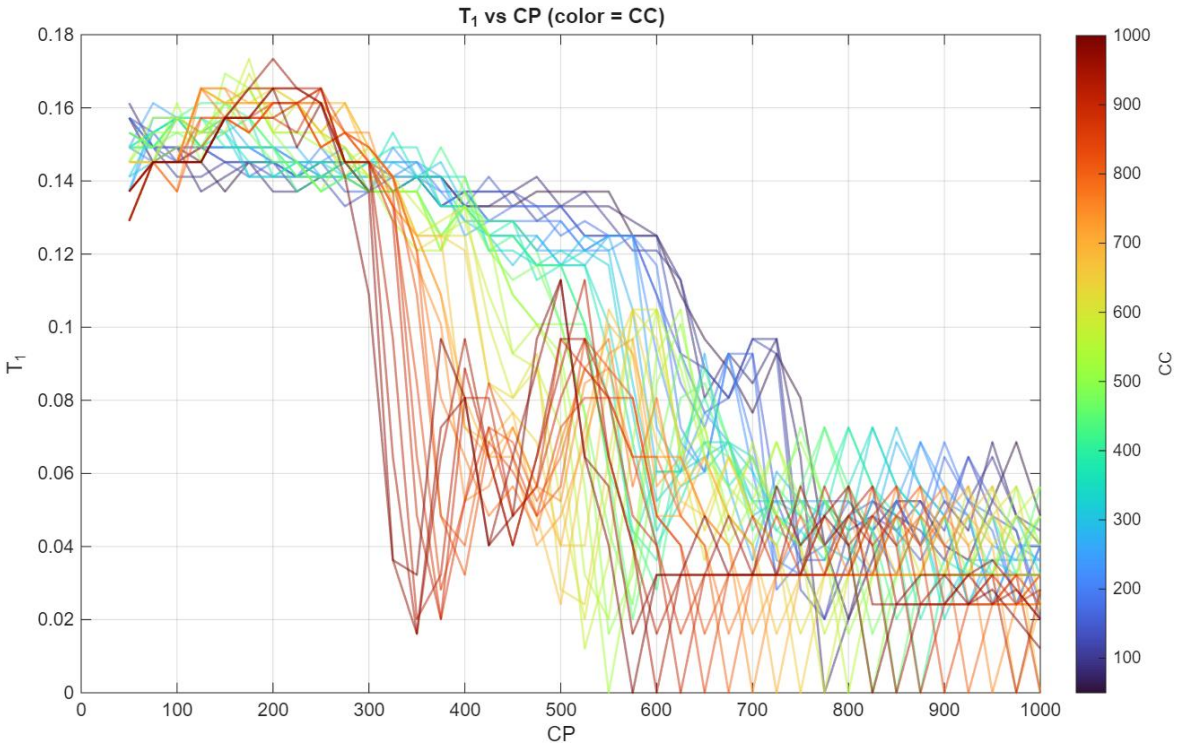


Figura 1. Gráfica T1 vs CP

Se puede observar en la Figura 1 que, al aumentar CP, el término T1 tiende a disminuir. Esto es coherente con la función objetivo. Si el valor de CP incrementa, el término T1, que lo multiplica, gana mayor importancia y el optimizador prioriza su minimización. Incluso, para algunos valores de CP, el término T1 llega a anularse.

Se hizo un segundo análisis. Esta vez comparando la evolución de T2 a medida que aumentaba el coeficiente CC para un mismo CP. La gráfica dibujada se refleja en la Figura 2, en la cual cada línea es una barrida para un valor de CP.

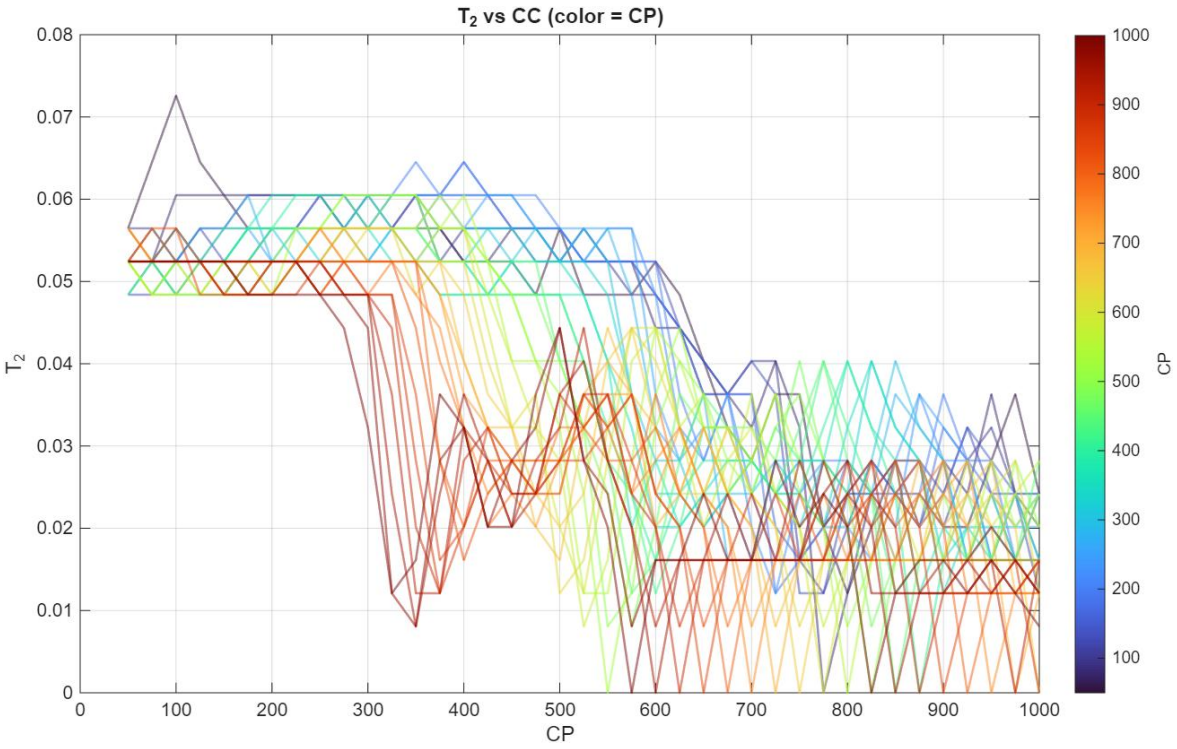


Figura 2. Gráfica T2 vs CC

Se observa que la gráfica de la Figura 2 es muy parecida a la de la Figura 1. Los valores de T2 tienden a disminuir cuando aumenta CC. Esto demuestra coherencia con el funcionamiento de la función objetivo.

6.4 DOMINANCIA DE T1

El comportamiento tiene una explicación directa en la formulación. Con $XS = 0$ ningún alimentador queda preasignado como referencia. La restricción (2), $n_i^{\uparrow} + n_i^{\downarrow} = x_i - XS_i$, pasa a ser $n_i^{\uparrow} + n_i^{\downarrow} = x_i$. En la función objetivo (1) el término T2 se reduce a lo siguiente:

$$T2 = \frac{1}{|\mathcal{I}|}\sum_i (n_i^{\uparrow} + n_i^{\downarrow}) \qquad (8)$$

Es decir, T2 no hace otra cosa que contar el número de alimentadores seleccionados. El término $T1 = \frac{1}{|\mathcal{I}|}\sum_i F_i \cdot x_i$ penaliza esa misma selección, pero ponderada por la importancia F_i asignada a cada alimentador según su tipo.

Los dos términos actúan sobre el mismo vector de decisión y empujan a seleccionar menos alimentadores. La diferencia es que T1 lo hace con más información. Aplicando la definición de dominancia, cualquier solución preferida por T1 es al menos igual de buena en aquello que mide T2, el recuento de alimentadores, y estrictamente mejor en la importancia, que T2 no distingue. T2 no genera entonces soluciones no dominadas que T1 no alcance ya. En esta configuración T1 domina a T2 y el término de cambios queda redundante.

Los barridos confirman esta lectura. Al aumentar CP la selección se reorganiza y T3 crece con claridad, como en el caso A5. Al aumentar CC los tres términos apenas se mueven a lo largo de todo el eje, tal como se veía en B5. El peso de CC casi no altera la solución. La sensibilidad que interesa estudiar es, por tanto, cómo responde T3 a T1 y no a T2.

6.5 T1 VS T3

Aprovechando los datos de los casos realizados previamente, se decidió estudiar la evolución del término T3 con respecto a la del término T1 a medida que se varía CP manteniendo CC constante. Con tal de que CC no influya en los resultados, se eligió el valor de CC más pequeño disponible en los casos.

Se dibujó una primera gráfica con ambos términos sin normalizar.

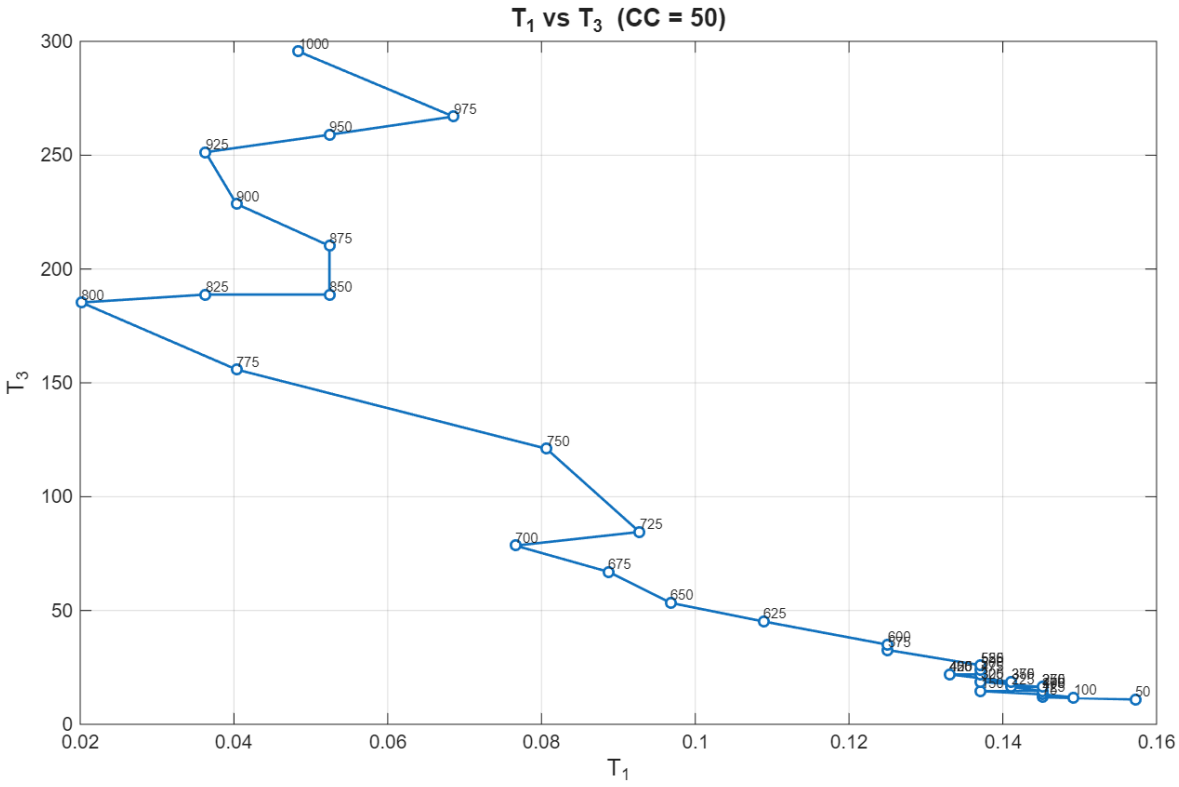


Figura 3. T1 vs T3 sin normalizar

La Figura 3 recoge el barrido de CP con CC = 50, donde cada punto está etiquetado con el valor de CP empleado. Lo que dibuja es el compromiso entre T1 y T3: al aumentar CP la solución reduce T1 a costa de incrementar T3. Como el fin último es minimizar T3, que cuantifica el desvío del deslastre respecto al objetivo, la zona de interés es la de T3 bajo. A partir de T3 ≈ 50 la curva se dobla con fuerza hacia arriba, de modo que reducciones cada vez menores de T1 exigen incrementos desproporcionados del desvío. Esos puntos se alejan del objetivo operativo y se descartan.

Con el fin de observar los casos relevantes, y viendo que para valores bajos de CP existe mucho ruido, se dibujó la misma gráfica para T3 < 50.

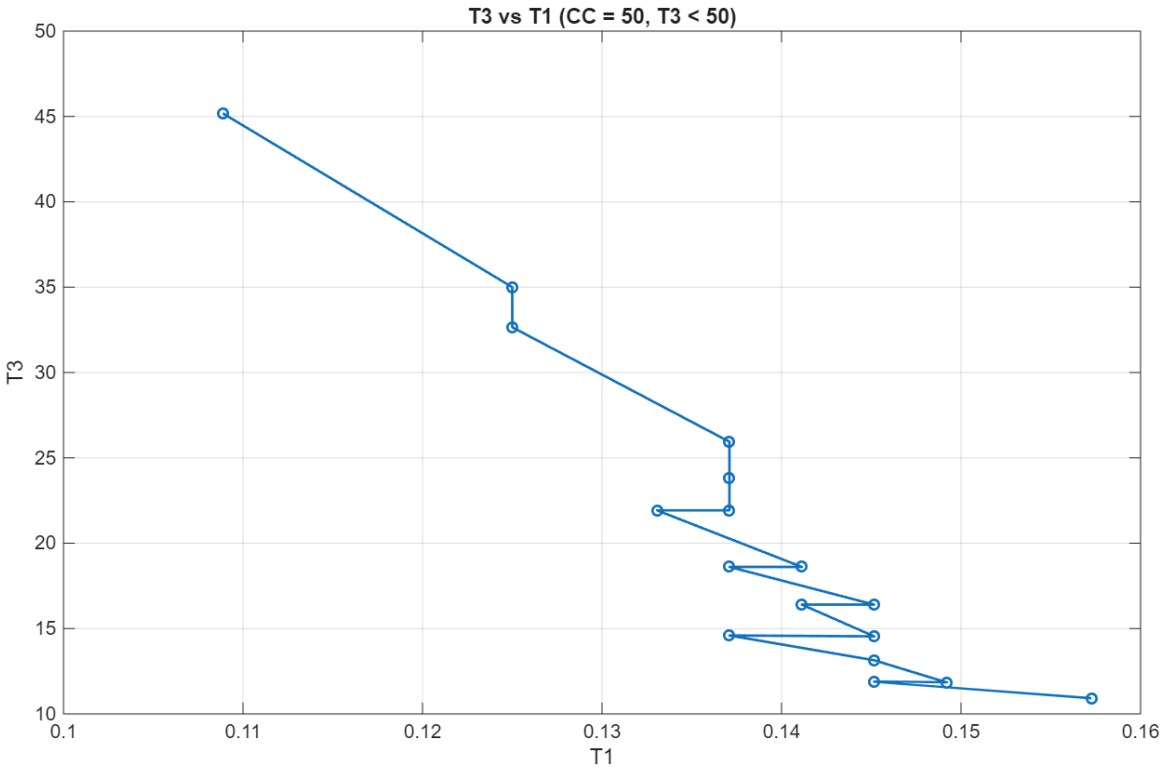


Figura 4. T1 vs T3 sin normalizar para T3 < 50

Como se ha mencionado anteriormente se busca minimizar T3 pero, a la vez, también T1. T1 cuantifica el número, agravado por la importancia, de los alimentadores seleccionados. El reducirlo permite que se desconecten el mínimo número de alimentadores y que los desconectados sean cuanto menos importantes. Se estima, en base a lo observado en la Figura 4, que esta zona óptima está alrededor de T1 = 0,14 y T3 = 15.

Se determinó que el valor máximo admisible de T3 fuese un 0,5% por encima del mínimo. Este valor se determinó en base a la experiencia y tolerancia permitida por el operador de distribución. Para ello fue necesario calcular el máximo, el punto nadir, del término T3 para normalizarlo.

El valor máximo de T1 y el mínimo de T3 (punto ideal) se da en el caso base. Al anular los coeficientes CP y CC, la función objetivo se centra únicamente en minimizar el término T3 sin darle importancia a lo que crezca T1 y T2.

El valor máximo de T3 se consiguió dándole a este un valor muy superior al de los demás. En este caso los valores tomados fueron:

```
CP=Vec_C_Prior*1000000 # vector de pesos (prioridad por tipología de consumo)
CC = 50  # Costo para el cambio en feeders
CO=1
```

Se obtuvo que el máximo de T3 era 614,2857.

La normalización de los coeficientes se hizo siguiendo las siguientes fórmulas:

$$T_1\ nor = \frac{\frac{1}{|\mathcal{I}|}\sum_i F_i \cdot x_i}{T_1\ max} \cdot 100 \qquad (9)$$

$$T_3\ nor = \frac{\frac{1}{LS \cdot |\mathcal{T}|}\sum_t (s_i^{\uparrow} + s_i^{\downarrow})}{T_3\ max} \cdot 100 \qquad (10)$$

Siguiendo las anteriores fórmulas (9) y (10), el mínimo valor de T3 normalizado en porcentaje sería el siguiente:

$$T_3 \min nor = \frac{5{,}8702}{614{,}2857} = 0{,}9556\% \qquad (11)$$

El valor de máximo de T3 admitido será 1,4556% con respecto a su punto nadir. A partir de este valor se considera que el sistema no minimiza lo suficiente el deslastre con respecto al objetivo.

Una vez normalizados los valores de los términos T1 y T3, se dibujó la gráfica normalizada en MATLAB, obviando los valores en los que T3 > 50 para ver los resultados de forma más clara.

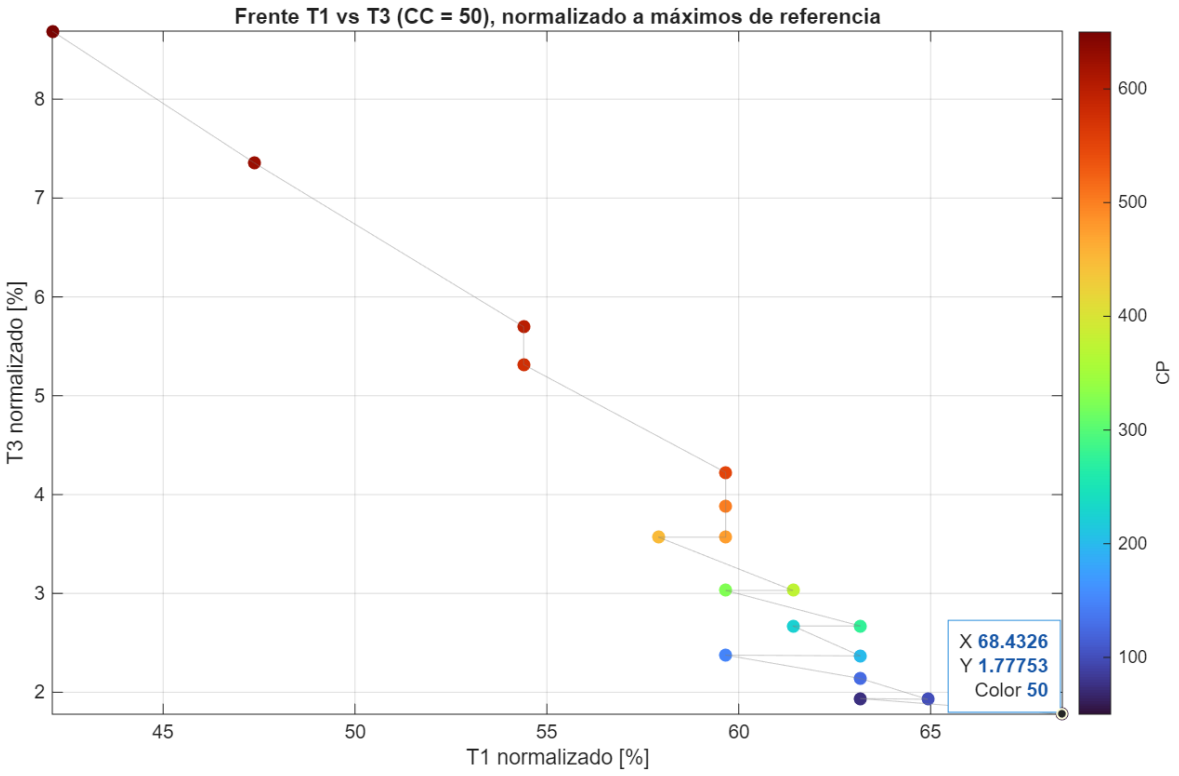


Figura 5. T1 vs T3 normalizados para T3 < 50

En la gráfica de la Figura 5 se ha resaltado el punto con el T3 más bajo. En este punto los coeficientes tienen los valores más pequeños de los casos probados, es decir, CC = 50 y CP = 50. Se ve que supera el umbral permitido de 0,5% sobre el T3 mínimo normalizado. Los coeficientes de este punto son excesivamente altos. El punto de funcionamiento buscado no se encuentra entre los casos probados.

Para encontrar la zona óptima de funcionamiento se hacen una serie de casos nuevos. En estos casos, como se había hecho previamente, se mantuvo CO = 1 mientras se variaba CC y CP. CC iba de 0 a 50 de 10 en 10, mientras que CP asumió valores de 10 a 100 de 10 en 10. Esto dio un total de 60 nuevos casos.

Se quiso volver a comparar los términos T1 y T3 normalizados para esta nueva tanda de casos.

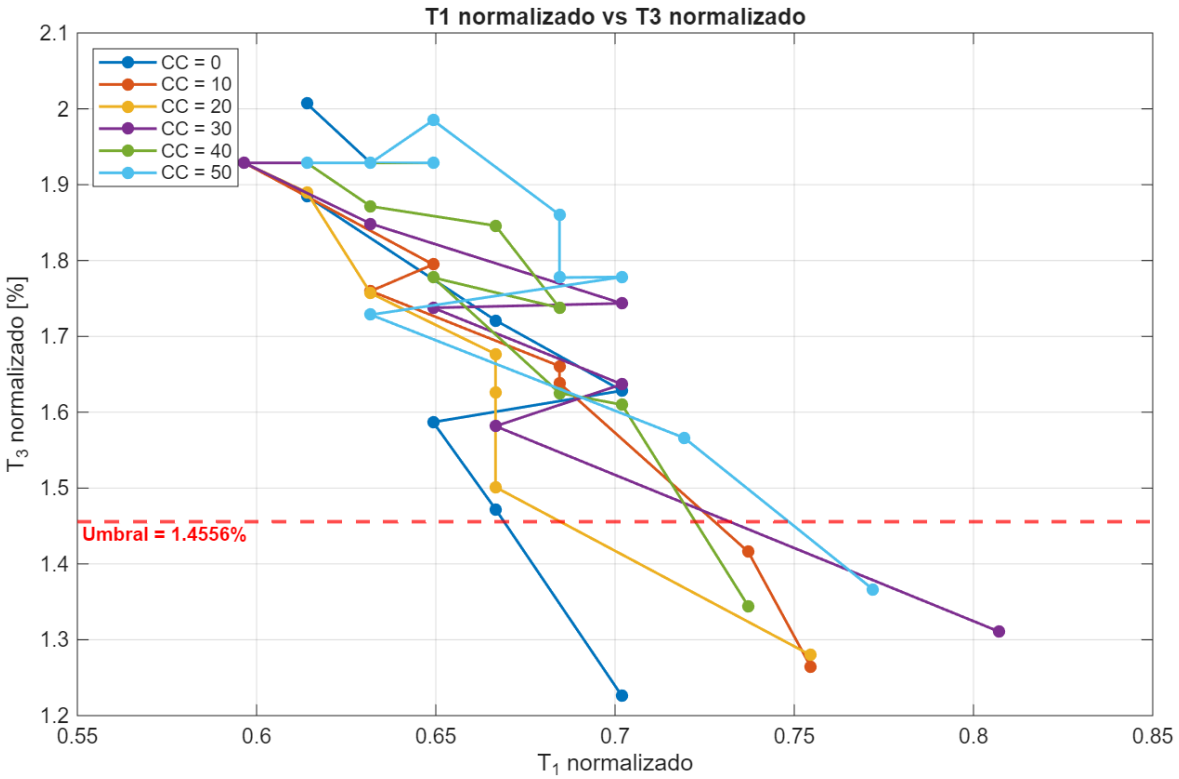


Figura 6. T1 vs T3 normalizado

Como se puede observar en la Figura 6 la gran mayoría de casos siguen estando por encima del umbral. El valor de CC que ofrece una reducción mayor del término T1 es 0. Caben destacar las líneas para las que CC = 20 y CC = 30. En estas dos líneas el cambio entre los primeros valores indica una fuerte reducción del valor del término T1. Para todos los valores de CC, a excepción de CC = 10, tenemos que T3 supera el umbral cuando CP ≥ 20.

Se llegó a la conclusión de que sería interesante estudiar los casos para valores de CP entre 11 y 19, incluidos, y CC siendo 0, 10, 20 y 30. Para ello se hicieron una serie de casos nuevos que comprendiesen estos valores de coeficientes.

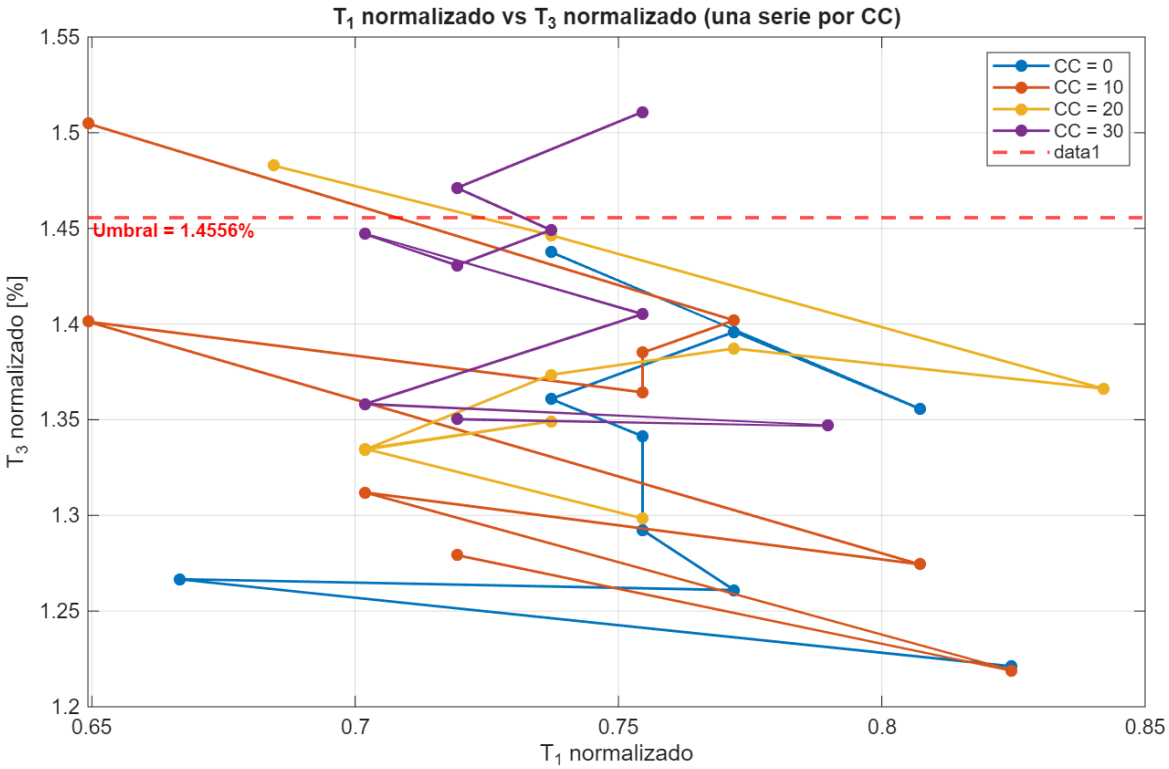


Figura 7. T1 vs T3 normalizado

La Figura 7 muestra que para estos datos el valor normalizado de T1 se comporta de manera prácticamente aleatoria. No sigue ningún patrón particular. Lo que sí se distingue es que el valor de T3 sí tiende a subir a medida que aumenta CP. Se decidió analizar únicamente los valores de T3 a la hora de decidir un umbral de coeficientes para este rango de casos.

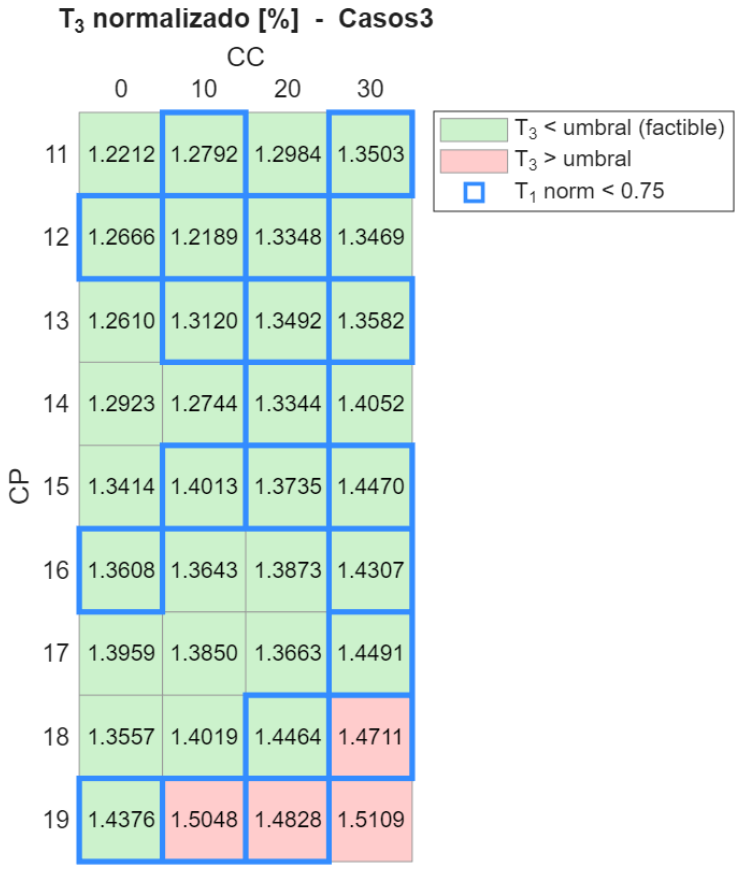


Figura 8. T3 normalizado

La figura recoge el T3 normalizado de los nueve valores de CP, de 11 a 19, frente a los cuatro de CC, 0, 10, 20 y 30. Las celdas verdes marcan los casos factibles, por debajo del umbral, y las rojas los que lo superan. El borde azul señala aquellos en los que T1 normalizado queda por debajo de 0,75.

La columna de CC = 0 es la única que se mantiene factible en todo su recorrido, y además reúne los valores más bajos de la matriz. Incluso en CP = 19 el T3 se queda en 1,4376, por debajo del umbral.

El T3 tiende a aumentar conforme sube CC dentro de cada fila y conforme sube CP dentro de cada columna, aunque sin un crecimiento estricto. Hay inversiones puntuales, como en CP = 12 y CP = 14, donde el valor baja al pasar de CC = 0 a CC = 10. Este comportamiento concuerda con el carácter ruidoso del rango. Los bordes azules, que

indican T1 normalizado por debajo de 0,75, aparecen dispersos por filas y columnas sin un patrón reconocible. Eso confirma que T1, para valores tan pequeños, se comporta de forma aleatoria y respalda la decisión de fijar el umbral atendiendo solo a T3.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

La meta de este trabajo era acotar un intervalo de CP y CC que produjera un funcionamiento coherente del esquema de deslastre, manteniendo XS = 0 en todas las pruebas. No se buscaba un valor único para cada coeficiente sino un rango viable. El caso base, con CP = CC = 0, sirvió como cota. Al anular ambos pesos el optimizador solo minimiza el desvío y desconecta más alimentadores de los necesarios, lo que fija el mínimo de T3 y el máximo de T1. Sobre esa base se definió un criterio de aceptación en base a la experiencia y tolerancia permitida por el operador de distribución. Una solución se consideró válida cuando su T3 normalizado no superaba el mínimo en más de medio punto porcentual, esto es, cuando quedaba por debajo del 1,4556% con respecto al máximo. Bajo ese criterio, el intervalo de valores de los coeficientes elegido es CC = 0 y CP por debajo de 17.

Como lo verdaderamente importante es la relación entre los coeficientes se fijó CO = 1 desde el principio. Este valor sirve de referencia para obtener CP y CC.

El primer límite que se pudo establecer fue el superior. En los casos de tipo A y B, al llevar los coeficientes hasta 1000, el término T3 se dispara. En A5 alcanza 266,977 y en B5 se queda en 19,8863, ambos muy por encima de cualquier deslastre razonable. Un peso de 1000 es desproporcionado, así que se tomó ese valor como tope de las pruebas posteriores, con la certeza de que el rango útil quedaría por debajo.

Tanto la Tabla 4 como la Tabla 6 mostraron que el barrido univariante había determinado que la variación empieza a notarse entre 10 y 100. El análisis bivariante arrancó por eso en 50, un punto intermedio de esa franja, recorriendo CC y CP de 50 a 1000. La gráfica normalizada de T1 frente a T3, mostrada en la Figura 5, reveló un problema. Incluso el caso más bajo del barrido, con CC = 50 y CP = 50, ya superaba el umbral. Los coeficientes seguían siendo demasiado altos y la zona de interés estaba más abajo. Quedaron descartados todos los valores seleccionados en las pruebas.

El umbral merece una explicación aparte, porque no sale del modelo. Se fijó según el criterio del operador de distribución. Se admitió como máximo un T3 que estuviera medio punto porcentual por encima del mínimo. Ese mínimo normalizado, obtenido en el caso base, resulta de dividir 5,8702 entre el máximo de T3, igual a 614,2857, y vale 0,9556%. Al sumarle el margen del 0,5% se obtiene el umbral del 1,4556%. Todo caso por encima de esa cifra se descarta al considerarse que el desvío con respecto al objetivo es excesivo.

Como los barridos de la primera tanda de casos lo superaban, se construyó una tanda nueva partiendo de la zona donde A y B mostraban el cambio, alrededor de 10. CP recorrió de 10 a 100 y CC de 0 a 50, ambos en pasos de 10. CC arrancó en cero porque interesaba el extremo en el que ese coeficiente no penaliza, y su tope se mantuvo en 50 por ser el valor mínimo de los barridos previos. La gráfica mostró una distancia grande entre los dos primeros valores de cada línea de CC, lo que situaba el cambio relevante en valores bajos de CP.

De ahí salió un último barrido fino, con CP entre 10 y 19. El foco aquí es muy estrecho y T1 normalizado se comporta de forma errática, sin patrón entre casos. T3 sí crece de forma ordenada conforme sube CP, así que el umbral final se decidió mirando solo ese término. El deslastre se mantiene dentro de lo admisible mientras CP no alcanza 17.

El recorrido completo, del caso base a los barridos finos, permite cerrar el intervalo de trabajo. Se recomienda CC = 0 y CP por debajo de 17. En la columna de CC = 0 todos los casos siguen siendo factibles hasta CP = 19, de modo que el corte en CP < 17 funciona como margen de seguridad más que como límite de factibilidad. Los valores CP = 18 y CP = 19 se excluyen del rango recomendado únicamente para disponer de un margen de seguridad, pese a que en la columna CC = 0 resulten factibles. Al aumentar CP el T3 se va acercando al umbral, y quedarse por debajo de 17 deja holgura frente a él.

Anular CC responde a una razón conceptual más que numérica. Con XS = 0 no hay asignación de referencia previa, de modo que el coste por cambio pierde su significado. CP

ya penaliza la selección de alimentadores ponderada por importancia, y mantener CC en cero evita duplicar esa penalización.

Con esto quedan cubiertos los objetivos iniciales. Se revisó la literatura de optimización multiobjetivo aplicada al problema, se identificaron y justificaron los términos de la función objetivo, se implementó el enfoque de suma ponderada y se estudió la sensibilidad de cada coeficiente hasta llegar a una recomendación concreta de parametrización.

En cuanto a trabajos futuros, se podría investigar e implementar otras formulaciones para resolver problemas de optimización multi-objetivo. Una muy prometedora puede ser la técnica ε-constraint.

BIBLIOGRAFÍA

[1] K. Miettinen, Nonlinear Multiobjective Optimization, Boston: Kluwer Academic Publishers, 1998.

[2] P. Halffmann, L. Schäfer, K. Dächert, K. Klamroth y S. Ruzika, «Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey,» *Journal of Multi-Criteria Decision Analysis,* vol. 29, nº 5-6, pp. 341-363, 2022.

[3] R. T. Marler y J. S. Arora, «The weighted sum method for multi-objective optimization: New insights,» *Structural and Multidisciplinary Optimization,* vol. 41, nº 6, pp. 853-862, 2010.

[4] J. R. Figueira, C. Fonseca, P. Halffmann, K. Klamroth, L. Paquete, S. Ruzika, B. Schulze y M. Stiglmayr, «Easy to say they are Hard, but Hard to see they are Easy—Towards a Categorization of Tractable Multiobjective Combinatorial Optimization Problems,» *Journal of Multi-Criteria Decision Analysis,* vol. 24, pp. 82-98, 2017.

[5] F. Y. Edgeworth, «Mathematical psychics,» *Mind,* vol. 6, nº 24, pp. 581-583, 1881.

[6] V. Pareto, Cours d'Économie Politique, Lausanne: F. Rouge, 1896.

[7] M. Ehrgott, «A discussion of scalarization techniques for multiple objective integer programming,» *Annals of Operations Research,* vol. 147, nº 1, pp. 343-360, 2006.

[8] A. M. Geoffrion, «Proper Efficiency and the Theory of Vector Maximization,» *Journal of Mathematical Analysis and Applications,* vol. 22, nº 3, pp. 618-630, 1968.

ANEXO I

```
import pandas as pd
import numpy as np
import pyomo.environ as pyo
#from pyomo.environ import quicksum
from pyomo.core.expr.numeric_expr import LinearExpression
import numpy as np
import pandas as pd
#from scipy.stats import norm
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import clear_output
import pickle
#import matplotlib.dates as mdatesrural
import json
import time
from datetime import date
from matplotlib.lines import Line2D



# Usar Computer Modern (por defecto en LaTeX)
plt.rcParams['font.family'] = 'serif'
plt.rcParams['mathtext.fontset'] = 'cm'  # 'cm' = Computer Modern
size_leter = 12 #usar 10, 11 o 12 según documento LaTex
#size_leter = 9 #usar 10, 11 o 12 según documento LaTex
plt.rcParams['font.size'] = size_leter  # Puedes
plt.rcParams['axes.titlesize'] = size_leter
plt.rcParams['axes.labelsize'] = size_leter
plt.rcParams['xtick.labelsize'] = size_leter
plt.rcParams['ytick.labelsize'] = size_leter
plt.rcParams['legend.fontsize'] = size_leter
plt.rcParams['figure.titlesize'] = size_leter

def Grap_MV_Time_P(start_date, end_date, df_tr,window_size, Name_Graph,
s_ylabel,i_show=1,i_zoom=0,y_limits=None,path=None):
    # Hacer una copia para no modificar el DataFrame original
    df_graph = df_tr.copy()

    # Establecer la columna TIMESTAMP como índice (si existe)
    if 'TIMESTAMP' in df_graph.columns:
        df_graph.set_index('TIMESTAMP', inplace=True)

    # Filtrar el DataFrame entre las fechas deseadas
    filtered_df = df_graph.loc[start_date:end_date]

    # Calcular la media móvil para cada serie de tiempo
    df_rolling = filtered_df.rolling(window=window_size).mean()
```

```
    # Crear la figura
    plt.figure(figsize=(15, 8))

    # Listar las columnas que NO sean lim_inf ni lim_sup
    other_columns = [col for col in df_rolling.columns if col not in ['lim_inf',
'lim_sup']]

    # Graficar las columnas "normales"
    for column in other_columns:
        plt.plot(df_rolling.index, df_rolling[column],
                 label=f'{column}', alpha=0.5)

    # Si existen lim_inf y lim_sup, rellenar la banda entre ambas
    if 'lim_inf' in df_rolling.columns and 'lim_sup' in df_rolling.columns:
        plt.fill_between(df_rolling.index,
                         df_rolling['lim_inf'],
                         df_rolling['lim_sup'],
                         color='gray', alpha=0.2, label='Quality boundaries')

    # Añadir título y etiquetas
    plt.title(Name_Graph)
    plt.xlabel('Estampa de tiempo')
    plt.ylabel(s_ylabel)

    # Establecer límites del eje y si se proporcionan
    if y_limits is not None:
        plt.ylim(y_limits)
    if i_show:
        plt.legend(loc='best')

def calc_E_M(x,desl,df_Ps,df_Pf,i_show=0):
    acum=lambda i,j: sum(x[i,jj] for jj in range(x.shape[1]) if jj<=j)
    x_acum = np.array([[acum(i,j) for j in range(x.shape[1])] for i in
range(x.shape[0])])
    desl_acu=np.array([])
    suma=0
    for v in desl:
        suma+=v
        desl_acu=np.append(desl_acu,suma)
    P_targ=(df_Ps@np.atleast_2d(desl_acu))
    UFLS=df_Pf@x_acum
    dev=UFLS-P_targ
    M_up=dev.mask(dev<0,0).abs()
    M_down=dev.mask(dev>0,0).abs()
    M_tot=M_up+M_down
    M_sum=M_tot.sum().sum()/1000
    #print(f'Total de energía fuera de los rangos de calidad = {M_sum:.3F} GWh')
    if i_show:
        print('M_down:\n',M_down.sum())
        print('M_up:\n',M_up.sum())
    return M_up,M_down,M_sum
```

```
def MILP_RISK(Ps, Pf, XS, XP, d_curr,d_acu, BigM, CP, CC, CO, epsilon=None,
i_mode=1, t_lim=300):

    # Pre-procesamiento de formas
    CP = np.atleast_1d(CP)
    Ps = np.atleast_1d(Ps)
    XS = np.atleast_1d(XS)
    XP = np.atleast_1d(XP)
    Pf = np.atleast_2d(Pf) # shape (NT, NF)
    NT, NF = Pf.shape

    if Ps.size == 1:
        Ps = np.full(NT, Ps[0])

    # Parámetros derivados
    k_cvar = 0
    if i_mode in (2, 4) and epsilon is not None:
        # Nota: k_cvar incluye 1/NT para promediar en la suma
        k_cvar = (1 / epsilon) * (1 / NT)

    # --- CREACIÓN DEL MODELO ---
    model = pyo.ConcreteModel()
    model.NT = pyo.RangeSet(0, NT - 1)
    model.NF = pyo.RangeSet(0, NF - 1)

    # --- VARIABLES (Sin índice j) ---
    model.x = pyo.Var(model.NF, domain=pyo.Binary)
    model.n_down = pyo.Var(model.NF, domain=pyo.NonNegativeReals)
    model.n_up = pyo.Var(model.NF, domain=pyo.NonNegativeReals)

    if i_mode in (1, 2):
        model.s_down = pyo.Var(model.NT, domain=pyo.NonNegativeReals)
        model.s_up = pyo.Var(model.NT, domain=pyo.NonNegativeReals)
    if i_mode == 2: # CVaR
        model.z = pyo.Var(model.NT, domain=pyo.NonNegativeReals)
        model.T = pyo.Var(domain=pyo.Reals) # T es escalar ahora
    if i_mode == 3: # Robust
        model.s_down = pyo.Var(domain=pyo.NonNegativeReals) # Escalar
        model.s_up = pyo.Var(domain=pyo.NonNegativeReals)   # Escalar
    if i_mode == 4: # Chance Constraint
        model.s_up = pyo.Var(domain=pyo.NonNegativeReals)
        model.y_down = pyo.Var(model.NT, domain=pyo.Binary)

    # --- PARÁMETROS ---
    model.XS = pyo.Param(
        model.NF,
        initialize=lambda m, i: XS[i]
    )
    model.XP = pyo.Param(
        model.NF,
        initialize=lambda m, i: XP[i]
    )
    # Porcentaje de potencia de cada feeder en cada tiempo t
```

```
    model.Pf_por = pyo.Param(
        model.NT,
        model.NF,
        initialize=lambda m, t, i: (Pf[t, i] / Ps[t]) * 100
    )

    model.CP = pyo.Param(
        model.NF,
        initialize=lambda m, i: CP[i]
    )

    # --- EXPRESIONES ---
    # Potencia total seleccionada en tiempo t
    model.sum_PF_x_expr = pyo.Expression(model.NT,
        initialize=lambda m, t: sum(m.Pf_por[t, i] * (m.x[i]+m.XP[i]) for i in
m.NF)
    )

    # --- FUNCIÓN OBJETIVO ---
    obj_expr = 0

    # Costos F.O. independientes de los escenarios operativos
    obj_expr += sum(model.CP[i] * model.x[i] for i in model.NF) / NF #Prioridad
de feeders
    obj_expr += CC*sum((model.n_up[i] + model.n_down[i]) for i in model.NF) / NF
#Cambios en feeders frente a XS

    if i_mode == 1: # Neutral Risk
        # Promedio de s_up y s_down en el tiempo
        Expec_loss = sum((model.s_up[t] + model.s_down[t]) for t in model.NT) /
(NT*d_curr)
        obj_expr += CO * Expec_loss + 100

    elif i_mode == 2: # CVaR
        # CVaR(s_down)
        term_cvar = model.T + k_cvar * sum(model.z[t] for t in model.NT)
        # Shedding esperado (solo up en modo 2 según tu código original, o
ambos?)
        # En tu código original sumabas s_up a la FO.
        term_expec_up = sum(model.s_up[t] for t in model.NT) / NT
        obj_expr += (CO/d_curr) * (term_cvar + term_expec_up)  + 100

    elif i_mode == 3: # Robust
        # s_up y s_down escalares (el peor caso)
        obj_expr += (CO/d_curr) * (model.s_up + model.s_down) + 100

    elif i_mode == 4: # Chance Constrained
        # Minimizar s_up (escalar)
        obj_expr += (CO/d_curr) * model.s_up + 100

    model.objetive = pyo.Objective(expr=obj_expr, sense=pyo.minimize)

    # --- RESTRICCIONES ---
```

```
    # Objetivo: La suma de feeders debe acercarse a 100*d_acu
    target = 100 * d_acu

    #Restricciones grales
    # Feeders activados y desactivados dependientes de la selección de xi
    model.changes =pyo.Constraint(model.NF,
        rule= lambda m,i:  m.n_up[i] - m.n_down[i] == m.x[i] - m.XS[i]
    )

    # Simultaneidad activados/desactivados
    model.one_hot_n =pyo.Constraint(model.NF,
        rule= lambda m,i:  m.n_up[i] + m.n_down[i] <= 1
    )
    # No repetir feeders por escalón
    model.one_hot_step =pyo.Constraint(model.NF,
        rule= lambda m,i:  m.x[i] + m.XP[i] <= 1
    )

    if i_mode in (1, 2):
        model.shed_constraint_inf = pyo.Constraint(model.NT,
            rule=lambda m, t: target - m.sum_PF_x_expr[t] <= m.s_down[t]
        )
        model.shed_constraint_sup = pyo.Constraint(model.NT,
            rule=lambda m, t: m.sum_PF_x_expr[t] - target <= m.s_up[t]
        )
    if i_mode == 2:
        # Restricción CVaR: z >= Loss - T
        # Loss se define como el déficit: s_down (lo que falta)
        model.pos_z = pyo.Constraint(model.NT,
            rule=lambda m, t: m.z[t] >= m.s_down[t] - m.T
        )
    if i_mode == 3:
        # En robusto, s_down y s_up son cotas para todo t
        model.shed_constraint_inf = pyo.Constraint(model.NT,
            rule=lambda m, t: target - m.sum_PF_x_expr[t] <= m.s_down
        )
        model.shed_constraint_sup = pyo.Constraint(model.NT,
            rule=lambda m, t: m.sum_PF_x_expr[t] - target <= m.s_up
        )
        model.robust_fix = pyo.Constraint(
             rule=lambda m: m.s_up >= m.s_down
        )
    if i_mode == 4:
        # Chance constraints
        model.shed_constraint_inf = pyo.Constraint(model.NT,
            rule=lambda m, t: target - m.sum_PF_x_expr[t] <= BigM * (1 -
m.y_down[t])
        )
        # Probabilidad de cumplimiento
        model.Risk_manage = pyo.Constraint(
            rule=lambda m: sum(m.y_down[t] for t in m.NT) >= NT * (1 - epsilon)
        )
        model.shed_constraint_sup = pyo.Constraint(model.NT,
```

```
            rule=lambda m, t: m.sum_PF_x_expr[t] - target <= m.s_up
        )
    return model

# Obtener rutas a los archivos
#Se extraen los datos originales de los archivos .csv
with open('Paths.txt', 'r') as archivo:
    lineas = archivo.readlines()
Path_0=lineas[0][:-1]
Path=lineas[1][:-1]
Path_cleaned_data=lineas[2]
Path_cleaned_data

# ==============DATOS SERIES TEMPORALES=========================
# Definir la CCAA para analizar
CCAA= 'MA' # 'MA', 'CL'

'''
# Rango de fechas para el cálculo de los modelos
# Training(FIT) data-set
start_date_fit='2023-01-01 00:00:00'
end_date_fit='2023-12-31 23:00:00'

# Pruebas (TEST) data-set
start_date_test='2024-01-01 00:00:00'
end_date_test='2024-11-01 00:00:00'

# OBTENER PARÁMETROS DE PF FEEDERS DESAGREGADOS
# Cargar el diccionario de feeders desagregados por CCAA
with open(Path_cleaned_data+"/dict_1h_CCAA_corrected_values.pkl", "rb") as f:
    dict_1h_CCAA_corrected = pickle.load(f)

# CARGA DE DATOS PF
df_import_Pf=dict_1h_CCAA_corrected[CCAA]
#Reemplazar nulos por ceros
df_import_Pf.iloc[:, 1:] = df_import_Pf.iloc[:, 1:].fillna(0)
df_import_Pf=df_import_Pf.set_index('TIMESTAMP')


# =+==+==+==+==+==+=CARGA DE DATOS PS=+==+==+==+==+==+==+==+=
# Cargar los datos históricos totalizados por CCAA
df_import_Ps =
pd.read_parquet(Path_cleaned_data+'/Totales_iDE_CCAA_corrected_values.parquet')
df_import_Ps = df_import_Ps[[CCAA]]

# Data sets por año
df_Pf_2023=df_import_Pf['2023-01-01 00:00:00':'2023-12-31 23:00:00'].copy()
df_Ps_2023=df_import_Ps['2023-01-01 00:00:00':'2023-12-31 23:00:00'].copy()
df_Pf_2024=df_import_Pf['2024-01-01 00:00:00':'2024-11-01 00:00:00'].copy()
df_Ps_2024=df_import_Ps['2024-01-01 00:00:00':'2024-11-01 00:00:00'].copy()


# CREAR TRAIN - TEST COMO EL 80% / 20% RESPECTIVAMENTE PARA EL 2023
```

```
df_comp=pd.concat([df_Pf_2023,df_Ps_2023],axis=1,ignore_index=False)
df_fit=df_comp.sample(frac=(0.8),random_state=42).sort_values(by='TIMESTAMP')
df_test = df_comp[~df_comp.index.isin(df_fit.index)]

df_Pf_fit=df_fit.iloc[:,:-1]
df_Ps_fit=df_fit[[CCAA]]
'''
df_test=pd.read_csv(f'df_test_{CCAA}.csv',sep=';')
df_Pf_test=df_test.iloc[:,:-1]
df_Ps_test=df_test[[CCAA]]


#  DATOS ENTRENAMIENTO Y PRUEBA
#Ps_fit=np.array(df_Ps_fit).reshape(-1)
#Pf_fit=np.array(df_Pf_fit)

Ps_test=np.array(df_Ps_test).reshape(-1)
Pf_test=np.array(df_Pf_test)
#=+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
#  DATOS REPRESENTATIVOS MEDIANTE CLUSTER PONDERADO
df_TCLUS = pd.read_csv(f'TSAw_{CCAA}_400.csv',sep=';')
df_Pf_TSAw = df_TCLUS.iloc[:, :-1].copy()
df_Ps_TSAw = df_TCLUS[[CCAA]].copy()
Pf_TSAw = np.array(df_Pf_TSAw)
Ps_TSAw = np.array(df_Ps_TSAw)
#=+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+

# ==============DATOS ESTRUCTURALES==========================================
df_struc_data = pd.read_parquet('structural_data.parquet')
df_struc_data = df_struc_data[df_struc_data['CCAA']==CCAA]
# Renombrar escalones
escalones=sorted(df_struc_data['ESCALON'].unique(),reverse=True)
d_escal=dict(zip(escalones,[np.nan,0,1,2,3,4,5]))
df_struc_data.loc[:,'ESCALON']=df_struc_data['ESCALON'].astype(object)
df_struc_data['ESCALON']=df_struc_data['ESCALON'].map(d_escal)
df_steps=df_struc_data[['PAIR_ID','ESCALON']].sort_values(by='PAIR_ID',
ascending=True).copy()

# EXPORTAR ARCHIVOS DESDE ORIGEN
#df_test.to_csv(f'df_test_{CCAA}.csv', sep=';', decimal='.', index=False)

dict_typ_codType=dict(zip(['Industrial','industrial','Rural','Domestico','SSPP',N
one],np.array(['i','i','r','d','p',None])))
dict_typ_cost=dict(zip(['i','r','d','p',None],np.array([1,3,4,5,2]))) #Definir
costo por tipología de consumo
dict_typ_rj=dict(zip(['i','r','d','p',None],np.array([0,0,2,3,0]))) #Definir
restrición tipología (a partir de que escalón puede ser elegido j>=. )

# Definir data set ID - Tipo de consumo
df_id_type=df_struc_data[['PAIR_ID','TIPO_CONSUMO']]
df_id_type=df_id_type[df_id_type['PAIR_ID'].isin(df_Pf_test.columns.astype(int))]
.sort_values(by='PAIR_ID') #Filtrar solo los feeders que tienen datos históricos
y ordenar
```

```
df_id_type.loc[:,'TIPO_CONSUMO']=df_id_type['TIPO_CONSUMO'].map(dict_typ_codType)
# df ID - Typología
df_id_type.index=df_id_type['PAIR_ID']
df_id_type.drop(columns=['PAIR_ID'],inplace=True)

# Crear vector de costos
df_id_type_cost=df_id_type.copy()
df_id_type_cost['TIPO_CONSUMO']=df_id_type_cost['TIPO_CONSUMO'].astype(object)
df_id_type_cost.loc[:,'TIPO_CONSUMO']=df_id_type['TIPO_CONSUMO'].map(dict_typ_cos
t)
df_id_type_cost.columns=['Cost']
Vec_C_Prior=np.array(df_id_type_cost)

# Crear vector de valor j permitido según tipología
df_id_typ_j=df_id_type.copy()
df_id_typ_j['TIPO_CONSUMO']=df_id_type['TIPO_CONSUMO'].map(dict_typ_rj)
df_id_typ_j.columns=['Constraint j>=']
vec_j_cons=np.array(df_id_typ_j)


# DEFINICIÓN PORCENTAJE DE DESLASTRE
with open("design_steps.pkl", "rb") as f:
    design_steps = pickle.load(f)
desl=np.array(design_steps[CCAA])

#SELECCIÓN DISEÑO DEL UFLS
mu_Ps=np.array(df_Ps_test.mean())[0]
P_targ=mu_Ps*desl.sum()
print(f'Potencia media objetivo a deslastrar = {P_targ} MW')

dict_RES={} # resultados para x
d_data={} #Performance de modelos
print('Deslastre por escalón',desl)
desl_acu=np.array([])
suma=0
for v in desl:
    suma+=v
    desl_acu=np.append(desl_acu,suma)
print('Deslastre acum por escalón',desl_acu)

#Cargas feeders actuales
f = np.array([int(n) for n in df_Pf_test.columns])
# Crear diccionario para acceso rápido
f_index = {val: idx for idx, val in enumerate(f)}

#Asegurarse que los feeder en datos estructurales y en la matriz Pf sean los
mismos
df_steps = df_steps[df_steps['PAIR_ID'].isin(f)]
#Asegurar que
df_steps=df_steps.sort_values(by='PAIR_ID', ascending=True)

NF = df_Pf_test.shape[1]
NST = len(desl)
```

```
x_iDE = np.zeros((NF, NST))

for j in range(NST):
    feeder_stp_j = df_steps[df_steps['ESCALON'] == j]['PAIR_ID'].values
    for f in feeder_stp_j:
        if f in f_index:
            x_iDE[f_index[f], j] = 1

for j in range(NST):
    dict_RES[f'UFLS_Actual_i_DE step-{j+1}']=x_iDE[:,j].reshape(NF,1)

d_data[f'UFLS_Actual_i_DE step-']=f'N/A'

# ========================================
#  MILP RISK EJECUCIÓN SECUENCIAL (BUCLE)
# ========================================
Pf_sec = Pf_TSAw.copy()
Ps_sec = Ps_TSAw.copy()
#Costos FO
i_cons_typ_j=1 #Usar bloqueo de feeders permitidos por escalón según su tipología
CP=Vec_C_Prior*0.1 # vector de pesos (prioridad por tipología de consumo)
CC = 0  # Costo para el cambio en feeders
CO=1

t_max = 1000
MipGap = 0.001
BigM = 10
epsilon = None
t_init = time.time()
Nom_caso = ''

results_T = []  # acumula T1, T2, T3 por valor de CP


# Restricciones externas
# Calculamos la desviación estándar por columnas (axis=0)
set_del_test = np.where(np.std(Pf_test, axis=0) == 0)[0]
set_del_sec = np.where(np.std(Pf_sec, axis=0) == 0)[0]
# Combinamos feeders a eliminar, y eliminamos duplicados con np.union1d
set_del_feeders = np.union1d(set_del_test, set_del_sec)

for i_model in [1]: #Modelos a ejecutar: 1: Neutral risk; 2:CVaR; 3:Robust;
4:[Chance_cons_inf]
    epsilon=None
    if i_model==1:
        Nom_caso='Caso D6 step-'
    if i_model==2:
        alfa=0.95
        epsilon=1-alfa #Nivel confianza en CVaR (1-e)
        Nom_caso=f'CVaR('+r'$\alpha$'+f'={(alfa)}) step-'
    if i_model==3:
        Nom_caso='Ro step-'
    if i_model==4:
```

```
        epsilon=0.05 #Riesgo o probabilidad de no cumplir la restricción
        Nom_caso='CCO'+r'($\epsilon$'+f'={epsilon}) step-'

    # Set de feeders seleccionados por escalón
    set_sel_feeders = np.array([])
    # Matrices de binarios x (Feeders , Escalones)
    XS = np.zeros((Pf_sec.shape[1], len(desl)))
    #XS = np.zeros((Pf_sec.shape[1], len(desl))).astype(int)
    XP = np.zeros((Pf_sec.shape[1])).astype(int)
    XF = np.zeros((Pf_sec.shape[1], len(desl))).astype(int)
    #x_final = np.zeros((Pf_sec.shape[1], len(desl)))
    L_gap = []

    # --- BUCLE PRINCIPAL PARA CADA ESCALÓN---
    for j in range(len(desl)):
        d_curr = desl[j]
        d_acum = desl_acu[j]
        print(f"\n>>> Optimizando: {Nom_caso[:-6]}; Escalón {j+1} (Target:
{d_curr*100:.2f}%, acum:  {d_acum*100:.2f}%)")
        # Encontrar feeder no permitidos para el escalón actual
        set_del_const=np.where(j<vec_j_cons)[0]

        # 1. Crear el modelo NUEVO para cada escalón
        m_risk = MILP_RISK(
            Ps_sec, Pf_sec, XS[:,j], XP, d_curr,d_acum, BigM, CP, CC, CO,
            epsilon,i_mode=i_model
        )

        # 2. Aplicar restricciones externas (Feeders ya usados o prohibidos)
        for i in m_risk.NF:
            if i in set_del_feeders:
                m_risk.x[i].fix(0)
            elif i in set_sel_feeders:
                m_risk.x[i].fix(0)
            elif i in set_del_const and i_cons_typ_j:
                m_risk.x[i].fix(0) #Bloquear feeders para el escalón actual

        # 3. Resolver
        solver = pyo.SolverFactory('gurobi')
        solver.options.update({
            'Method': 2,
            'Threads': 24,
            'TimeLimit': t_max,
            'MIPGap': MipGap
        })

        res = solver.solve(m_risk, tee=True)

        # 4. Guardar métricas y GAP
        if hasattr(res.problem, 'upper_bound') and hasattr(res.problem,
'lower_bound'):
            ub = res.problem.upper_bound
            lb = res.problem.lower_bound
```

```
            if ub is not None and lb is not None:
                gap = abs(ub - lb) / max(abs(ub), 1e-5)
                L_gap.append(gap)

        # 5. Extraer resultados y actualizar lista de seleccionados
        # Guardamos la columna j en nuestra matriz de resultados
        XF[:,j] = np.array([round(pyo.value(m_risk.x[i])) for i in
m_risk.NF]).astype(int)
        XP = XP + XF[:,j]
        set_sel_feeders=np.where(XP>0)[0]

        # 6. Opcional reestablecer bloqueo de feeres por tipología
        for i in m_risk.NF:
            if i in set_del_const:
                m_risk.x[i].unfix() #Desbloquear feeders para el escalón actual

    t_end = time.time()
    t_tot = t_end - t_init
    gap_avg = (sum(L_gap) / len(L_gap)) * 100 if L_gap else 0.0
    d_data[Nom_caso]=f't={t_tot:.5}s;gap={gap_avg:.3}%'
    print(f'\n=== FIN ===')
    print(f'Gap promedio: {gap_avg:.2f} %')
    print(f'Tiempo total: {t_tot:.4f} s')

    # Guardar en diccionario de resultados
    for j in range(len(desl)):
        dict_RES[Nom_caso + str(j+1)] = np.atleast_2d(XF[:, j]).T

    # T1, T2, T3 del último escalón
T1 = sum(pyo.value(Vec_C_Prior[i] * m_risk.x[i]) for i in m_risk.NF) /
Pf_sec.shape[1]
T2 = sum(pyo.value(m_risk.n_up[i] + m_risk.n_down[i]) for i in m_risk.NF) /
Pf_sec.shape[1]
T3 = sum(pyo.value(m_risk.s_up[t] + m_risk.s_down[t]) for t in m_risk.NT) /
(len(Ps_sec) * desl[-1])

print(T1)
print(T2)
print(T3)

#Número de cambios realizados en la selección de feeders
abs((XF-XS)).sum()

df_Pf_data = df_Pf_test.copy()
df_Ps_data = df_Ps_test.copy()

beta=0.95

# Casos a evaluar
cases=list(set([n[:-1] for n in dict_RES.keys()]))
display(cases)

# Crear gráfico de distribuciones de probabilidad
```

```
h=1 #Escoger escalón {1,2,3,4,5,6}
plt.figure(figsize=(7, 4))
#linea del centro
plt.axvline(0, color='gray', linestyle='-', linewidth=2,alpha=0.6)
Colors=["#36AE42","#EB5353","#D98E04","#04A2BE","#B504BE","#1F12A3","#808080","#7
20992"]

k=0 #¡ No coambiar
for n in cases:
    #if n in [cases[i] for i in [1, 2]]: # Escoger casos para graficar
        x=np.zeros(shape=(len(dict_RES[n+str(1)]),len(desl)))
        for j in range (len(desl)):
            for i in range(x.shape[0]):
                x[i,j]=dict_RES[n+str(j+1)][i][0]

s_up,s_down,s_sum=calc_E_M(x[:,:h],desl[:h],df_Ps_data,df_Pf_data,i_show=0)
        P_targ=(df_Ps_data@np.atleast_2d(desl_acu[:h]))
        s_por=((s_up - s_down)/P_targ)*100
        s_por_down=((s_down)/P_targ)*100
        s_por_up=((s_up)/P_targ)*100
        desv_w=((s_por*desl_acu[:h])/desl_acu[:h].sum()).sum(axis=1) #
        s_por_acum=s_por[h-1]
        # Calcular límites de percentiles
        L_LOW = -s_por_down.quantile(beta)[h-1]
        L_HIGH = s_por_up.quantile(beta)[h-1]

        #sns.kdeplot(desv_w,label=n[:-6]+f'\n {L_LOW:.3}%,
{L_HIGH:.3}%',color=Colors[k],alpha=1,kde=True)
        sns.histplot(data=s_por_acum, bins=100,
kde=True,color=Colors[k],edgecolor=Colors[k],
                     label=n[:-6],
                     #+f'\n{L_LOW:.3}%, {L_HIGH:.3}%',
                     alpha=0.15, stat='density'
                     )
        # Líneas de percentiles
        #plt.axvline(L_LOW, color=Colors[k], linestyle='--
',label=f'L_LOW={L_LOW:.3}%')
        #plt.axvline(L_HIGH, color=Colors[k], linestyle='-
.',label=f'L_HIGH={L_HIGH:.3}%')
        plt.axvline(L_LOW, color=Colors[k], linestyle='--')
        plt.axvline(L_HIGH, color=Colors[k], linestyle='-.')
        k+=1

#==============Leyenda adicional
custom_lines = [
    Line2D([0], [0], color="grey",alpha=0.8, linestyle='--',
label=r'$\mathcal{Q}$(' f'{(1-beta)*100:.3}%)'),
    Line2D([0], [0], color="grey",alpha=0.8, linestyle='-.',
label=r'$\mathcal{Q}$(' f'{(beta)*100:.3}%)'),
]
first_legend = plt.legend(handles=custom_lines, loc='upper left')
plt.gca().add_artist(first_legend)
#==============================
```

```
#plt.title('Graphical representation for L-LOW and L-HIGH')
#plt.xlabel('Deviation of UFLS from target in %')
plt.xlabel('')
plt.ylabel('Probability density')
plt.legend(loc='center left')
plt.xlim(-20, 20)  # Limitar eje x
#plt.ylim(0, 1)
#plt.grid(True)
plt.tight_layout()
# Exportar a PDF
#plt.savefig(f"out_Figure/sens_CP_deviat_{CCAA}_fit_stp{h}.pdf")
plt.show()
plt.close()

# Validar métricas de calidad KPIs
dict_res_cases=dict()
for n in cases:
    #if n in [cases[i] for i in [1]]: # Escoger casos a evaluar
    #-----------------------------------------
        x=np.zeros(shape=(len(dict_RES[n+str(1)]),len(desl)))
        for j in range (len(desl)):
            for i in range(x.shape[0]):
                x[i,j]=dict_RES[n+str(j+1)][i,0]

s_up,s_down,s_sum=calc_E_M(x[:,:h],desl[:h],df_Ps_data,df_Pf_data,i_show=0)
        P_targ=(df_Ps_data@np.atleast_2d(desl_acu[:h]))
        s_por=((s_up - s_down)/P_targ)*100
        s_down_por=((s_down)/P_targ)*100
        s_up_por=((s_up)/P_targ)*100
        s_por_std=(s_por-s_por.mean())/s_por.std()
        #q_real=s_por_std.quantile(1-epsilon)
        #print(f'Cuantil-std (1-epsilon)= q({1-epsilon})={q_real[h-1]}')
        Expec_up_v=s_up_por.mean().values[h-1]
        Expec_down_v=-s_down_por.mean().values[h-1]
        Expec=(s_up_por+s_down_por).mean().values[h-1]
        Var_down_b=-(s_down_por).quantile(beta).values[h-1]
        Var_up_b=(s_up_por).quantile(beta).values[h-1]
        Tail_down=s_down_por[s_down_por>=s_down_por.quantile(beta)]  #Cálculo
CVaR (-S_down + S_up)
        CVaR_down=-Tail_down.mean().values[h-1]
        Tail_up=s_up_por[s_up_por>=s_up_por.quantile(beta)]  #Cálculo CVaR (-
S_down + S_up)
        CVaR_up=Tail_up.mean().values[h-1]
        Extr_down=-s_down_por.max().values[h-1]
        Extr_up=s_up_por.max().values[h-1]

v_res=[Expec_down_v,Expec_up_v,Expec,Var_down_b,Var_up_b,CVaR_down,CVaR_up,Extr_d
own,Extr_up,Extr_up-Extr_down]
        v_res=np.round(v_res,2)
        dict_res_cases[n[:-6]]=v_res
    #-------------------------------------------------------------------------
---
```

```
All_res=pd.DataFrame(dict_res_cases,index=['E(s↓)','E(s↑)','E(|s↓|+|s↑|)','VaR(β)
(s↓)','VaR(β)(s↑)','CVaR(β)(s↓)','CVaR(β)(s↑)','Extr(s↓)','Extr(s↑)','Extr(|s↓|+|s
↑|)'])
print(f'Resultados para el escalón acumulado {h}, β={beta}')
indx=All_res.index.values
All_res.index=[n for n in indx]
display(All_res)

# Performance modelos
print('Performance summary of models:')
dict_res_cases=dict()
for n in cases:
    dict_res_cases[n[:-6]]=d_data[n]

pd.DataFrame(dict_res_cases,index=['Perform']).T
```

ANEXO II. ALINEACIÓN CON LOS ODS

El trabajo se relaciona con la Agenda 2030 de las Naciones Unidas a través de cuatro Objetivos de Desarrollo Sostenible: el 7, el 9, el 11 y el 13. La conexión nace de su propio objeto, la estabilidad de frecuencia y el deslastre de carga en un sistema con presencia creciente de generación renovable.

ODS 7. Energía asequible y no contaminante

La contribución a un suministro fiable y sostenible se produce en dos planos. El primero es la fiabilidad. Ajustar el deslastre real al objetivo de diseño refuerza la última línea de defensa de la red y rebaja tanto la probabilidad como el alcance de los apagones. El segundo es la integración de generación renovable y distribuida. La parametrización de CP, CC y CO conserva la eficacia del esquema justo en las condiciones de alta variabilidad donde los métodos de selección tradicionales pierden rendimiento.

ODS 9. Industria, innovación e infraestructura

La red eléctrica de distribución es una infraestructura crítica, y sobre ella se aplica aquí una formulación MILP resuelta con el método de la suma ponderada. Ese enfoque reemplaza la práctica habitual, un ajuste manual basado en unos pocos escenarios de punta o valle, por un procedimiento sistemático que parte de datos históricos reales. Su coste de adopción es reducido, ya que descansa en software y en la parametrización de relés de baja frecuencia, y se adapta a redes de tamaños diversos, condiciones que facilitan su paso a la industria.

ODS 11. Ciudades y comunidades sostenibles

El apagón que sufrió España en 2025 mostró cómo un corte generalizado paraliza la economía y deja sin servicio a hospitales, transporte y comunicaciones. Frente a ese riesgo, el modelo ordena las desconexiones por tipo de consumo y antepone los servicios públicos y el suministro doméstico al rural y al industrial. Cuando el deslastre no puede evitarse, los

consumos más sensibles quedan protegidos, y con ello disminuye la exposición de la población a las perturbaciones graves del sistema.

ODS 13. Acción por el clima

La lucha contra el cambio climático pasa, en el sector eléctrico, por descarbonizar la generación. Un parque con mucha renovable aporta variabilidad e incertidumbre a los flujos de potencia y desgasta unas herramientas de protección pensadas para un sistema más previsible. Un esquema de deslastre reforzado retira un obstáculo operativo a esa transición: la red admite más generación limpia sin poner en riesgo la estabilidad de frecuencia. Es una vía indirecta, pero real, de apoyo a la acción climática.