COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Desarrollo y aplicación de técnicas para la interpretabilidad de la programación estocástica

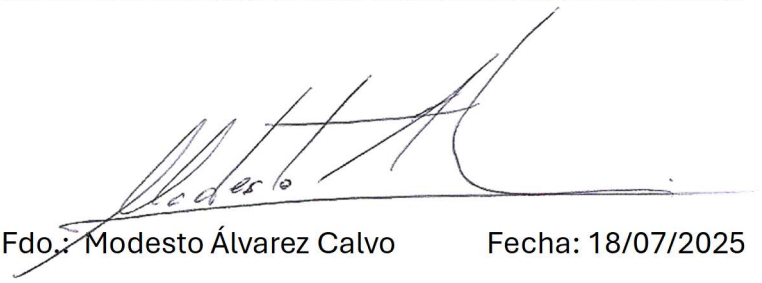
Autor: Modesto Álvarez Clavo

Directora: Dra. Sara Lumbreras Sancho

Madrid

Julio de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título "Desarrollo y aplicación de técnicas para la interpretabilidad de la programación estocástica" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.



Fdo.: Modesto Álvarez Calvo Fecha: 18/07/2025

Autorizada la entrega del proyecto

LA DIRECTORA DEL PROYECTO

Fdo.: Sara Lumbreras Sancho Fecha: 18/07/2025

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

TRABAJO FIN DE GRADO

Desarrollo y aplicación de técnicas para la interpretabilidad de la programación estocástica

Autor: Modesto Álvarez Calvo

Directora: Dra. Sara Lumbreras Sancho

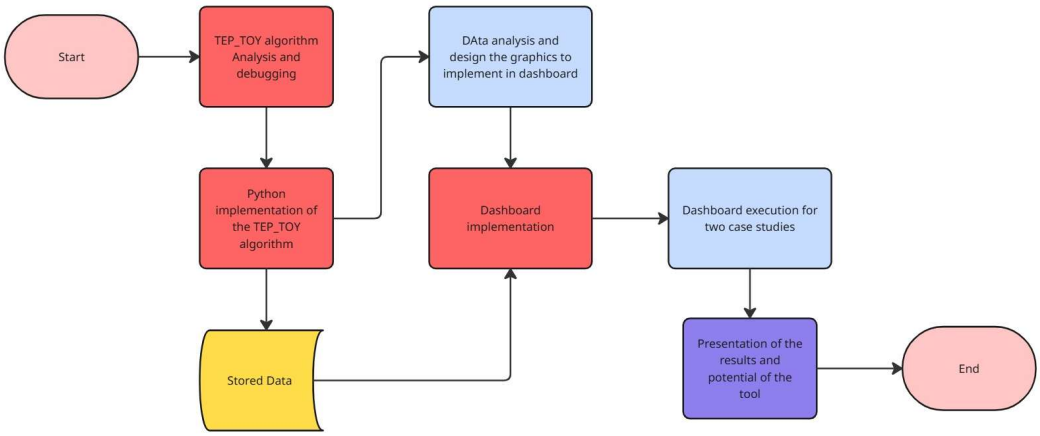
Desarrollo y aplicación de técnicas para la interpretabilidad de la programación estocástica

Autor: Modesto Álvarez Calvo
Director: Dr. Sara Lumbreras Sancho
Entidad colaboradora: ICAI - Universidad Pontificia Comillas, Madrid

Resumen del proyecto

Este trabajo propone una metodología multicriterio y una herramienta de visualización en Python desentrañar y presentar gráficamente la complejidad de la estocástica, mejorando su interpretabilidad y la toma de decisiones.

Palabras clave—Programación Estocástica, Interpretabilidad, Graficabilidad, Conflicto.



Introducción

La novedad de este trabajo reside en mejorar la interpretabilidad de sus resultados y no en la formulación de modelos estocásticos. En el presente trabajose ha aplicado a la planificación energética, donde las decisiones a tomar dependen de complejos procesos matemáticos que deben hacerse comprensibles para agentes técnicos y no técnicos.

Aunque los modelos estocásticos permiten obtener soluciones óptimas bajo incertidumbre, su adopción por parte de los responsables de las decisiones se ve limitada por la complejidad y opacidad de los resultados obtenidos a través de estos métodos.

Para salvar esta brecha, el proyecto propone una metodología que combina programación estocástica con teoría de decisión multicriterio, en una primera etapa, para posteriormente desarrollar una herramienta de visualización gráfica de conflictos entre combinaciones de escenarios y escenarios individuales. Se introduce el concepto de “conflicto” entre soluciones y escenarios, analizándolo cuantitativa y cualitativamente como base para interpretar de forma intuitiva lo que sucede al aplicar una solución calculada para un set de escenarios a cada escenario en particular. Preguntas como si se deben cumplir todas las restricciones de todos los escenarios o si se aceptan violaciones parciales podrán ser abordadas a partir de los gráficos

obtenidos, presentando de forma visual y asequible la información sobre cómo influyen los distintos escenarios en la solución final. Al identificar y visualizar explícitamente los conflictos entre escenarios, se pretende dotar al usuario de herramientas para comprender mejor el comportamiento del modelo.

Definición del proyecto

El punto de partida de este trabajo es un algoritmo desarrollado por la Dra. Sara Lumbreras Sancho, denominado **"TEPTOY"**. El proyecto se ha basado en adaptar y extender dicho algoritmo para cumplir con los objetivos planteados en este TFG.

El modelo se estructura en una resolución en espiral: se resuelve primero el modelo estocástico completo, luego cada escenario por separado, y finalmente combinaciones de dos, tres y cuatro escenarios. Para cada solución, se simulan de nuevo todos los escenarios fijando las inversiones obtenidas, y se calcula el coste comparado con el óptimo específico del escenario. Los conflictos se cuantifican mediante una métrica por unidad (PU), permitiendo la comparación entre combinaciones.

Descripción de la herramienta

Con el objetivo de mejorar la visualización e interpretabilidad, se reimplementó el algoritmo en Python, aprovechando su flexibilidad y capacidades gráficas. Esta transición no estuvo exenta de dificultad, pero permitió una estructura de código más eficiente y comprensible, cuyos resultados podrán ser utilizados de forma inmediata por la aplicación gráficas.

La implementación en Python replica la lógica central del modelo GAMS, pero calcula y evalúa únicamente las combinaciones de escenarios necesarias, eliminando redundancias. Cada combinación se define mediante una distribución de probabilidades, se resuelve, y se generan los resultados necesarios para la evaluación de conflictos.

El valor central de este TFG reside en la herramienta de visualización interactiva desarrollada con la biblioteca gráfica de Python, Streamlit, que da lugar a un panel de control llamado Dashboard.py. Esta interfaz permite a usuarios no expertos explorar la existencia y magnitud de los conflictos entre soluciones individuales y combinadas de forma intuitiva.

La herramienta realiza análisis tanto binarios como cuantitativos, destacando qué escenarios generan más conflictos y cómo se comportan distintas combinaciones en términos de coste y robustez. Se muestran costes óptimos y esperados junto con los indicadores de conflicto, facilitando decisiones informadas y transparentes.

Resultados

La principal aportación del trabajo es un marco metodológico y computacional para visualizar conflictos entre escenarios en modelos de optimización estocástica. Para demostrar su utilidad, se aplicó la herramienta a dos casos de estudio con la misma red, pero diferentes parámetros.

Caso 1: Configuración original. Se observa que el escenario s1 genera la mayoría de los conflictos, mientras que s2–s3 y s4–s5 forman dos grupos diferenciados. Los mapas de calor y gráficos de barras muestran tanto el número como la intensidad de los conflictos. Un gráfico que representa la "evolución del conflicto" permite visualizar cómo ciertas combinaciones reducen más conflictos que incluso la solución estocástica.

Caso 2: Se modifican valores de potencia en algunos nodos y el valor de la penalización por demanda no satisfecha. Aparecen conflictos mucho más marcados. El escenario s1 sigue siendo el más problemático, mientras que s5 se vuelve el más benévolo. Aunque la combinación s3–s4 minimiza el número de conflictos, su intensidad PU es mayor.

Es importante remarcar, que los resultados de estos análisis no son el resultado más valioso del TFG, sino la herramienta desarrollada en sí misma. Si bien, la aplicación a la expansión de red que supone la aplicación concreta de este trabajo se ha resuelto según lo ya explicado.

En ambos casos, el panel permite identificar escenarios clave, evaluar combinaciones robustas y analizar compensaciones de forma interactiva. El usuario puede modificar parámetros, ejecutar el modelo y visualizar los resultados, acercando la programación estocástica a la toma de decisiones práctica.

Conclusiones

Lo que comenzó como una herramienta gráfica sencilla ha derivado en el desarrollo de un auténtico sistema de apoyo a la decisión para interpretar modelos estocásticos en redes eléctricas. La traducción del algoritmo desde GAMS a Python, la integración de teoría multicriterio y la creación de un panel interactivo superan ampliamente las expectativas académicas iniciales.

Pese a no contar con una base sólida en programación, el proyecto ha logrado construir una herramienta flexible, transparente y potente que aproxima la modelización matemática a la toma de decisiones reales; permite a usuarios no expertos explorar conflictos, identificar escenarios relevantes y mejorar la planificación de redes energéticas bajo incertidumbre. La novedad de este trabajo consiste en hacer interpretables los modelos estocásticos.

Referencias

[1] GAMS Development Corporation, *General Algebraic Modeling System (GAMS) - User Guide*. Washington, DC, USA: GAMS Dev. Corp., 2023.

[2] A. Ramos and S. Cerisola, *Stochastic Optimization Lecture Notes*, Instituto de Investigación Tecnológica (IIT), Universidad Pontificia Comillas, Madrid, Spain, 2024. [Online]. Available: www.iit.comillas.edu

[3] A. Ramos, P. Sánchez, J. M. Ferrer, and S. Wogrin, *Mathematical Models of Specific Optimization Techniques*, IIT, Universidad Pontificia Comillas, 2024.

[4] A. Ramos, P. Sánchez, and S. Wogrin, *Mixed Integer Linear Modeling*. IIT, Universidad Pontificia Comillas, 2024.

[5] M. Álvarez Calvo, Apuntes de Investigación operativa, Universidad Pontificia Comillas, 2024.

[6] S. Lumbreras Sancho, "Resolución progresiva de conflictos inter-escenario", algoritmo no publicado, Universidad Pontificia Comillas, Madrid, Spain, 2025.

[7] Python Software Foundation, *Python Language Reference*, version 3.10. [Online]. Available: https://www.python.org/

[8] Streamlit Inc., *Streamlit Documentation*, 2025. [Online]. Available: https://docs.streamlit.io

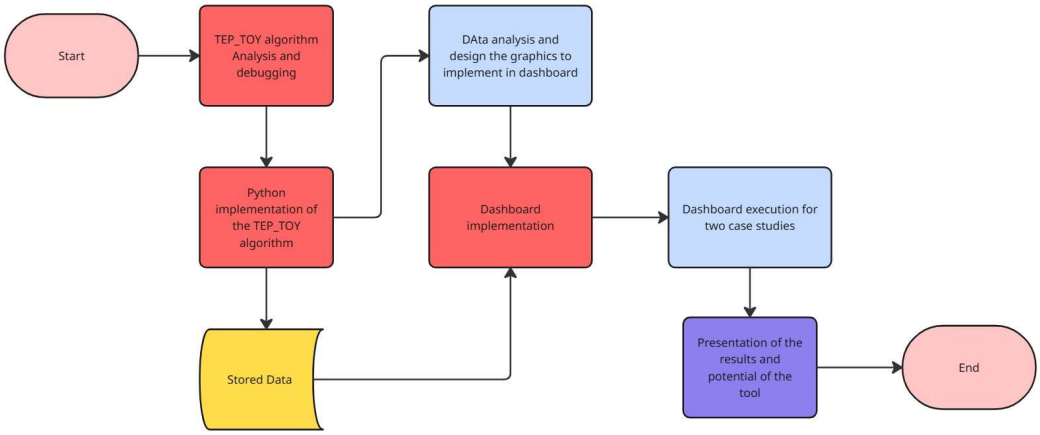
Development and Application of Techniques for the Interpretability of Stochastic Programming

Author: Modesto Álvarez Calvo
Supervisor: Dr. Sara Lumbreras Sancho
Collaborating entity: ICAI - Universidad Pontificia Comillas, Madrid

Abstract

This paper proposes a multicriteria-based methodology and Python dashboard to visualize scenario conflicts in stochastic energy planning, improving interpretability and decision-making.

Keywords—Stochastic Programming, Interpretability, Conflict, Visualization.



Introduction

The novelty of this work lies not in the formulation of stochastic models, but in enhancing the interpretability of their results, particularly important in electric network planning, where decisions must be understood by both technical and non-technical stakeholders. While stochastic models can offer optimal solutions under uncertainty, their complexity often hinders their adoption by decision-makers.

To bridge this gap, the project proposes a methodology that combines stochastic programming with multicriteria decision-making theory and develops a tool for graphical visualization of scenario conflicts. The concept of “conflict” between solutions and scenarios is introduced and analyzed quantitatively and qualitatively, offering an intuitive framework for interpreting deviations between scenario-specific and aggregated solutions. Questions such as whether all scenario constraints must be satisfied or whether partial violations are acceptable are placed at the center of the analysis.

This approach addresses a key limitation of standard stochastic models: the lack of insight into how individual scenarios influence the final decision. By explicitly identifying and visualizing scenario-based conflicts, the methodology aims to provide decision-makers with tools to better understand and trust the models' outputs.

Project Definition

The foundation of this work is an algorithm developed by Dr. Sara Lumbreras Sancho, titled **"TEPTOY"**. The project began by adapting and extending this algorithm to meet the specific objectives of this work.

The model follows a spiral resolution structure: it first solves the full stochastic model, then optimizes each individual scenario, and finally evaluates pairs, triplets, and quartets of scenarios. For each solution, fixed investment decisions are re-simulated across all scenarios, and the resulting costs are compared to the scenario-specific optimum. Conflicts are computed using a per-unit (PU) metric, enabling comparisons across different combinations.

In total, the original GAMS model evaluated 43 scenario combinations and generated 94 conflict metrics. These results form the basis for the visualization tool described in the next section.

Tool Description

To enhance visualization and interpretability, the GAMS-based algorithm was reimplemented in Python, leveraging its flexibility and visualization capabilities. This transition was challenging, but it led to a more streamlined and efficient codebase.

The Python implementation replicates the core logic of the GAMS model while computing only the 31 unique scenario combinations, avoiding redundancies. Each combination is defined by a probability distribution and solved to generate the required outputs for conflict evaluation.

The innovation lies in the visualization component developed using Python's Streamlit library, resulting in an interactive dashboard named Dashboard.py. This dashboard enables users — especially non-experts— to explore the existence and magnitude of conflicts between scenario-based and semi-stochastic solutions in an intuitive format.

The tool does both binary and quantitative analyses, providing insights into which scenarios contribute most to conflict and how different combinations affect cost and resilience. Optimal and expected costs are displayed alongside conflict indicators, supporting informed trade-offs in decision-making.

Results

The main contribution of this project is a methodological and computational framework for visualizing inter-scenario conflicts in stochastic optimization. To showcase its capabilities, two test cases were conducted using the same network but with different parameters and penalty values.

Case 1 uses the original configuration. Results show that scenario s1 generates the most frequent conflicts, while scenarios s2–s3 and s4–s5 form two distinguishable groups. Heatmaps and bar charts help illustrate the number and intensity of conflicts per scenario. A graph tracking the "evolution of conflict"—plotting the number of conflicts versus solution complexity and cost—highlights how some combinations reduce conflict more effectively than even the stochastic solution.

Case 2 modifies power values at nodes and the penalty for not met demand. Here, the contrast between conflicts is sharper: they are either very high or nearly zero. Scenario s1 remains problematic, while s5 becomes the least conflicting. Although the s3-s4 solution minimizes the number of conflicts, its intensity (PU) is higher.

It is important to insist that the results of these analyses are not the main result of this thesis, but the graphic tool itself.

Nevertheless, in both cases, the dashboard enables decision-makers to identify key scenarios, evaluate robust combinations, and explore trade-offs interactively. Users can modify parameters, re-run optimizations, and visualize the results, making stochastic modeling more accessible and actionable.

Conclusions

Originally aimed at producing simple graphical representations, this project evolved into the development of a full-fledged decision support tool for interpreting stochastic optimization in energy networks. The translation of the algorithm from GAMS to Python, the integration of multicriteria decision theory, and the construction of an interactive dashboard were significant achievements that exceeded the expected academic scope.

Despite the author's limited programming background, the project succeeded in creating a flexible, transparent, and powerful tool that bridges the gap between complex mathematical modeling and practical decision-making; it enables non-expert users to explore scenario conflicts, identify influential parameters, and improve network design under uncertainty. It also encourages further exploration, such as extending conflict analysis to the full solution space, applying clustering or dimensionality reduction techniques, and studying the dynamic behavior of stochastic systems. This project's contribution consists of a novel framework for making stochastic optimization interpretable.

References

[1] GAMS Development Corporation, *General Algebraic Modeling System (GAMS) - User Guide*. Washington, DC, USA: GAMS Dev. Corp., 2023.

[2] A. Ramos and S. Cerisola, *Stochastic Optimization Lecture Notes*, Instituto de Investigación Tecnológica (IIT), Universidad Pontificia Comillas, Madrid, Spain, 2024. [Online]. Available: www.iit.comillas.edu

[3] A. Ramos, P. Sánchez, J. M. Ferrer, and S. Wogrin, *Mathematical Models of Specific Optimization Techniques*, IIT, Universidad Pontificia Comillas, 2024.

[4] A. Ramos, P. Sánchez, and S. Wogrin, *Mixed Integer Linear Modeling*. IIT, Universidad Pontificia Comillas, 2024.

[5] M. Álvarez Calvo, *Class Notes on Multicriteria Decision Theory in Operations Research*, Universidad Pontificia Comillas, 2024.

[6] S. Lumbreras Sancho, "Resolución progresiva de conflictos inter-escenario", unpublished algorithm, Universidad Pontificia Comillas, Madrid, Spain, 2025.

[7] Python Software Foundation, *Python Language Reference*, version 3.10. [Online]. Available: https://www.python.org/

[8] Streamlit Inc., *Streamlit Documentation*, 2025. [Online]. Available: https://docs.streamlit.io

Contenido

Agradecimientos

Quiero agradecer al Beato (y próximamente Santo) Carlo Acutis por responder a mis ruegos y acudir en mi ayuda cuando no había forma de que el código en Python funcionara después de días y días de revisiones, pruebas; cuando la desesperación invitaba incluso a abandonar este TFG y buscar otro para el año que viene. Gracias.

También quiero agradecer por su accesibilidad y comprensión a mi directora, la Dra. Sara Lumbreras Sancho que me ha dado la oportunidad de colaborar con ella en la intuición genial que se convirtió en el algoritmo que ha dado origen a este trabajo; y por la confianza que depositó en mí para la elaboración de este TFG. Espero no haberla decepcionado.

Por último, y en este caso sí un poco menos, quiero agradecer al Dr. Andrés Ramos que me devolviera el gusto por la ciencia estadística, tan decididamente arrebatado antes de nacer por los docentes anteriores y las circunstancias.

I. Introducción

Cuando se afrontó la idea de este trabajo, aún no había tenido lugar el apagón del 28 de abril que tuvo sin energía eléctrica a toda la península ibérica. Sin embargo, tanto este TFG como las prácticas curriculares desarrolladas por el autor, han tocado esta eventualidad de una forma central. En las prácticas se abordó el estudio de cómo dar a una central fotovoltaica una inercia (presente de forma natural en las plantas de generación con máquinas rotativas); y en el TFG cómo tener en cuenta la incertidumbre de la generación de renovables en el diseño de expansión de una red eléctrica.

Desde la aparición en 1955 con los trabajos de Dantzig [1] y Beale [2] de la programación estocástica como extensión de la programación lineal en entornos en los que la incertidumbre juega un papel central, su aporte a la optimización ha sido remarcable. A pesar de la complejidad matemática de los métodos de resolución de este tipo de problemas, ofrecen una oportunidad de abordar problemas de optimización estudiando los distintos escenarios con sus respectivas incertidumbres. La variabilidad y el riesgo que suponen sistemas complejos tienen así una solución abordable, siempre desde la incertidumbre que conllevan los distintos posibles escenarios. Con esta herramienta, se puede enfrentar problemas donde los datos son desconocidos, incompletos o imprecisos. La programación estocástica ofrece soluciones robustas y adaptativas en estas circunstancias.

Problemas de este tipo, Se pueden encontrar en el mundo financiero, la logística o la misma inteligencia artificial, ahora tan en auge. El ámbito de aplicación que abordará este trabajo será el de las redes de transporte de energía en entornos de generación de energías renovables. El reto en este ámbito es doble por la doble incertidumbre: en la demanda de energía y en la generación de este tipo de energías.

Este trabajo, como se ha apuntado, pretende avanzar en una línea de investigación que aún no ha sido explorada. Ciertamente, la programación estocástica es un campo muy estudiado. Los métodos de resolución en esta área de la optimización siguen siendo objeto de estudio debido a los avances en capacidad de cálculo que ofrecen las nuevas tecnologías informáticas. Sin embargo, lo que este trabajo pretende abordar es el reto de ofrecer una interpretabilidad de la solución que ofrece la optimización estocástica a los problemas de redes de distribución de energía eléctrica, sin limitarse a ellos. Efectivamente, una vez se alcance una solución al problema planteado en este trabajo, podrá ser de utilidad en cualquier campo de la estocástica.

Este Trabajo Fin de Grado, enfrenta el reto ante un vacío que la programación estocástica deja abierto y que puede resultar en una herramienta de enorme utilidad para que los decisores puedan tener una mayor comprensión de la complejidad de la solución que este tipo de optimización ofrece, así como de sus potenciales adaptaciones. De este modo, se ofrecerá a los tomadores de decisiones una capacidad antes desconocida para alterar los parámetros del problema y de la solución y así alcanzar mejores decisiones.

Sin embargo, la solución ofrecida por estos métodos de programación y optimización, si bien resuelve el problema teniendo en cuenta los distintos escenarios posibles, resulta demasiado oscura (podría decirse que incluso críptica) para los tomadores de decisión. El problema viene de no saber cómo están influyendo cada uno de los escenarios en la solución final ¿Cuál es el escenario que está forzando una solución mayor, en lo que a dimensionamiento de la red se refiere? ¿Sería asumible una solución que tuviera menos en cuenta el escenario más desfavorable ya que sólo sucedería en un tiempo transitorio breve?

La propuesta que este trabajo de investigación pretende ofrecer, intentará aplicar a la programación estocástica, las aportaciones de la teoría de la decisión "multicriteria" para poder hacer más accesible a los tomadores de decisiones la comprensión y la interpretabilidad de los resultados de la programación estocástica de una forma gráfica de modo que la comprensión de este tipo de métodos sea intuitiva y pueda llegar a ser casi inmediata.

La optimización estocástica, una vez vencido el escollo de la resolución de problemas de enormes dimensiones, presenta un reto que este trabajo quiere arrostrar: Cómo influye en la solución final de la optimización estocástica cada posible escenario. Se introducirá para el propósito del trabajo el concepto de conflicto. Este se cuantificará y se cualificará en función de un umbral como ya se explicará más adelante. Dado que la solución estocástica, por su misma naturaleza, no satisface al 100% las necesidades de todos los escenarios, cuando una solución a un escenario o una solución semi-estocástica (se verá lo que esto significa) entra en conflicto con un escenario, cómo de importante es ofrecer una solución que satisfaga cada uno de ellos ¿Podría ser admisible una cierta transgresión en alguna de las condiciones que impone un escenario más restrictivo si se da sólo en el régimen dinámico?

La programación estocástica hasta hoy no ha afrontado este reto de desentrañar el peso de cada escenario posible, en la solución final. Este trabajo persigue llenar ese vacío para lo que estudiará los conflictos entre las soluciones de los distintos escenarios tomados de manera individual y en todas sus combinaciones posibles para sí poder ofrecer una comprensión más sencilla y visual de los resultados de esta programación.

El problema

Se parte de una red con tres nodos y unos datos de generación y de demanda. La generación puede tener dos fuentes: energías renovables y energías no renovables. Las energías no renovables se tratan como energías de reserva, pues la intención es que la satisfacción de la demanda provenga en la mayor medida posible de fuentes renovables.

También para la demanda deben contemplarse diferentes escenarios ya que, aunque con los históricos que se poseen pueden predecirse con bastante precisión los patrones de demanda, siempre que considerar una cierta aleatoriedad ya que los desequilibrios que se dan entre generación y demanda deben estar muy acotados para que las fluctuaciones de voltaje y de frecuencia pueden ser muy dañinas para la red y los sistemas que de ella dependen.

Así las cosas, se plantean cinco diferentes escenarios en función de la generación de energías renovables y la demanda. Se desea diseñar la red de transporte de energía eléctrica teniendo en cuenta los distintos escenarios con sus respectivas incertidumbres. Aquí es donde entra en juego la programación estocástica. La programación entera mixta (en adelante MIP por sus siglas en inglés) podría arrojar una solución óptima para un solo escenario. Pero si se desea tener en cuenta todos los escenarios considerados, conocida (o al menos estimada) la probabilidad con la que cada escenario podría darse es necesario implementar métodos estocásticos de programación para calcular la dimensión óptima de la red teniendo en cuenta la mencionada incertidumbre.

Para intentar desentrañar cómo está afectando cada escenario a la solución óptima estocástica, una vez resuelto el problema de optimización, se procede a integrar las técnicas de conflictos multicriteria para intentar contrastar la solución de cada escenario con los demás. Es decir, ¿qué problema se da si aplicamos la solución de un escenario a otro? ¿Cómo se puede medir esos posibles desajustes de una solución aplicada a un escenario para el que no fue diseñada?

Se desea ir unos pasos más allá y se propone calcular las diferentes soluciones “semiestocásticas” y actuar del mismo modo que con las soluciones individuales de cada escenario. Se llaman soluciones semiestocásticas a los resultados de la optimización estocástica, pero teniendo en cuenta varios escenarios sin que sean todos (que sería la solución estocástica completa. Así, se irán tomando pares de escenarios, ternas, y cuaternas y aplicando las técnicas de programación estocástica a estas combinaciones de escenarios. Las soluciones obtenidas de estas combinaciones también se contrastarán con cada uno de los escenarios para obtener los distintos conflictos y obtener información de lo que exige cada escenario a las distintas soluciones.

II. El Algoritmo

Se parte del algoritmo diseñado por la directora de este trabajo, Dra. Sara Lumbreras Sancho: "Resolución progresiva de conflictos inter-escenario", que se adjunta en el Anexo 1.A.

El primer esfuerzo fue para ajustar y depurar el algoritmo de modo que se alineara con los objetivos del presente trabajo.

Debido a que la programación estocástica está fuera del currículo para el Grado en Ingeniería de las Tecnologías Industriales, y que el nivel de conocimientos de programación en "General Algebraic Modeling System" (en adelante GAMS) necesarios para resolver los distintos modelos de programación estocástica exceden los recibidos en clase, el trabajo supuso un esfuerzo, dedicación y tiempo muy considerables, dados los estimados por la ETSII ICAI para el TFG. Sin embargo, el trabajo dio su fruto y además de los conocimientos de estocástica y programación en GAMS, se obtuvo el siguiente código que posteriormente se explicará paso a paso:

```
* ---------------------------------------------
* GAMS Model: Network Expansion Optimization with 3 Nodes and 5 Scenarios
* Objective: Minimize total investment and operational costs
* ---------------------------------------------
$onListing

Sets
    n   Nodes /n1, n2, n3/
    s   Scenarios /s1, s2, s3, s4, s5/
    t   Types /t1, t2/
    idx index solution /idx01*idx99/
;
alias (n,m), (s,ss,sss,ssss,sssss), (idx,idxx)
;
sets
    cand(n,m,t) Corridors between nodes n and m of type t
        /
        n1.n2.t1, n1.n2.t2,
        n1.n3.t1, n1.n3.t2,
        n2.n3.t1, n2.n3.t2,
        n2.n1.t1, n2.n1.t2,
        n3.n1.t1, n3.n1.t2,
        n3.n2.t1, n3.n2.t2
        /

    conflicts(s,ss)
    conflictsthree(s,ss,sss)
    conflictsfour(s,ss,sss,ssss)
    conflictsfive(s,ss,sss,ssss,sssss)

;

* Parameters Definition
Parameter
    D(n,s)          Demand at node n under scenario s
        /
```

```
    n1.s1 100, n1.s2 0,    n1.s3 100,  n1.s4 0,    n1.s5 100,
    n2.s1 0,   n2.s2 100,  n2.s3 100,  n2.s4 100,  n2.s5 100,
    n3.s1 0,   n3.s2 0,    n3.s3 0,    n3.s4 100,  n3.s5 100
    /,

R(n,s)          Renewable generation at node n under scenario s
    /
    n1.s1 150, n1.s2 150, n1.s3 150, n1.s4 150, n1.s5 150,
    n2.s1 0,   n2.s2 0,   n2.s3 0,   n2.s4 0,   n2.s5 0,
    n3.s1 0,   n3.s2 0,   n3.s3 0,   n3.s4 0,   n3.s5 0
    /,
B_cap(n,s)      Backup generation capacity at node n under scenario s
    /
    n1.s1 50, n1.s2 50, n1.s3 50, n1.s4 50, n1.s5 50,
    n2.s1 50, n2.s2 50, n2.s3 50, n2.s4 50, n2.s5 50,
    n3.s1 50, n3.s2 50, n3.s3 50, n3.s4 50, n3.s5 50
    /,
InvestCost(t)    Investment cost for corridor type t
    /
    t1 20000,
    t2 30000
    /,
Cap(t)           Capacity for corridor type t
    /
    t1 100,
    t2 200
    /,
 Penalty         Penalty cost for NSE
    /1000/
,
Price_B         Price per unit of backup generation (single scalar)
    /150/
,
InitialProb(s)         Scenario probability
    /s1 0.2, s2 0.2, s3 0.2, s4 0.2, s5 0.2/
,


Threshold         Threshold for conflict
    /0.20/

LastIdx   last index

Prob(s)   scenario probability to use


SaveProbsOpt(idx,s) probabilities in optimization
SaveOptFixed(idx) investment cost optimized
SaveOptInv(idx,n,m,t) investment variables
SaveOptVariable(idx) variable cost optimized
SaveOptTotal(idx) total cost optimized
SaveSimVariable(idx,s) variable cost simulated for each explored solution
SaveSimTotal(idx,s) total cost simulated
SaveSimExpectedVal(idx) expected value simulated
pAux
```

```
    Conflict(idx,s) conflict
    ConflictPU(idx,s) conflict in pu

    a
    b
    c
    cc
    ;

* Variables Definition
Positive Variables
    flow(n,m,s)      Power flow through corridor
    g_B(n,s)         Backup generation
    nse(n,s)         Non-Served Energy
    spill(n,s)       Spillage
    invCost
    opCost
;

Binary Variable
    invest(n,m,t)    Capacity invested in corridor
;

Variable
    totalCost;

* Equations Definition
Equations
    obj                   * Objective function
    powerBalance(n,s)     * Power balance at each node and scenario
    flowLimit(n,m,s)      * Flow limits based on invested corridor capacities
    backupLimit(n,s)      * Backup generation limits
    invCostdef            * investment cost definition
    opCostdef             * operation  cost definition
    nse_limit(n,s)        * Non-served energy limit
    spill_limit(n,s)      * Spillage limit
;

* Objective Function: Minimize Investment + Operational Costs
obj..
    totalCost =e=
        sum((n,m,t)$(cand(n,m,t)), invest(n,m,t) * InvestCost(t)) +
        sum((n,s), g_B(n,s) * price_B * Prob(s)) +
        sum((n,s), nse(n,s) * Penalty * Prob(s));

* Power Balance Constraint
powerBalance(n,s)..
    R(n,s) + g_B(n,s) + sum(m, flow(m,n,s)) -
    sum(m, flow(n,m,s)) =e= D(n,s) - nse(n,s) + spill(n,s);

* Flow Limits Constraint: Flow <= Sum of Invested Capacity across Types
flowLimit(n,m,s)..
    flow(n,m,s) =L= sum(t, invest(n,m,t) * Cap(t));

* Backup Generation Limit Constraint
backupLimit(n,s)..
```

```
    g_B(n,s) =L= B_cap(n,s);

* nse limit
nse_limit (n,s)..
    nse(n,s) =L= D(n,s);

* spillage limit
spill_limit(n,s)..
    spill(n,s) =L= R(n,s);

invCostdef..
    invCost =e= sum((n,m,t)$(cand(n,m,t)), Invest(n,m,t) * InvestCost(t));

opCostdef..
     opCost =e= sum((n,s), g_B(n,s) * price_B * Prob(s)) +
        sum((n,s), nse(n,s) * Penalty * Prob(s));

* Model Definition
Model NetworkExpansion /all/;

***********************************************************************
* INITIAL EXECUTION
***********************************************************************

* First test execution
Prob(s) = InitialProb(s);
conflicts(s,ss) = no;

flow.fx(n,n,s) = 0;
invest.fx(n,n,t) = 0;

Solve NetworkExpansion minimizing totalCost using MIP;
Display 'ejecución inicial'
Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l,
spill.l;


***********************************************************************
* CALCULATION PER INDIVIDUAL SCENARIO
***********************************************************************

* First we will calculate the initial solution again, then we will optimize
each scenario separately
SaveProbsOpt('idx01',s) = InitialProb(s);
SaveProbsOpt(idx,s)$[(ord(idx)>1)        and        (ord(idx)<=card(s)+1)]=
0+1$[ord(s)=ord(idx)-1];
LastIdx = card(s)+1;

**** THIS CAN BE AN INCLUDE, IT WILL ALWAYS BE THE SAME ******
loop(idx$(ord(idx)<=LastIdx),

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;
    invest.fx(n,n,t) = 0;
```

```
    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l,
spill.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display  invest.l,  flow.l,  g_B.l,  nse.l,  totalCost.l,  invCost.l,
opCost.l, spill.l;
        SaveSimVariable(idx,sss)= opCost.l;
        SaveSimTotal(idx,sss) = totalCost.l;

     );
     SaveSimExpectedVal(idx)     =     sum(sss,     InitialProb(sss)     *
SaveSimTotal(idx,sss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$(ord(idx)<=LastIdx),

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END

* Calculating conflict pairs
conflicts(s,ss) = no;
conflicts(s,ss)$[(ord(s)     <>     ord(ss))     and     [sum(idx$[ord(idx)=
ord(s)+1],ConflictPU(idx,ss))] >= Threshold] = yes;
display conflicts, LastIdx;


*****************************************************************
* CALCULATION OF SOLUTIONS WITH TWO SCENARIOS
*****************************************************************
```

```
* Creating new solutions to solve the conflicts between pairs of scenarios

loop(conflicts(s,ss),
    pAux = InitialProb(s)+InitialProb(ss);
    SaveProbsOpt(idx,s) $[ord(idx) = LastIdx+1] = InitialProb(s)/pAux;
    SaveProbsOpt(idx,ss)$[ord(idx) = LastIdx+1] = InitialProb(ss)/pAux;

    LastIdx = LastIdx+1;
);

* Evaluating them.
**** THIS CAN BE AN INCLUDE ******
loop(idx$[(ord(idx)> card(s)+1) and (ord(idx)<=LastIdx)],

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;
    invest.fx(n,n,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l,
spill.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display  invest.l,  flow.l,  g_B.l,  nse.l,  totalCost.l,  invCost.l,
opCost.l, spill.l;
        SaveSimVariable(idx,sss)= opCost.l;
        SaveSimTotal(idx,sss) = totalCost.l;

     );
     SaveSimExpectedVal(idx)     =     sum(sss,     InitialProb(sss)     *
SaveSimTotal(idx,sss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$[(ord(idx)> card(s)+1) and (ord(idx)<=LastIdx)],

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));
```

```
        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END
display ConflictPU;
conflictsthree(s,ss,sss) = no;
loop(idx$ [[ord(idx)> card(s)+1] and [ord(idx)<=LastIdx]],

    conflictsthree(s,ss,sss) $ [[ord(s) < ord(ss)] and [ord(sss) <> ord(s)]
and   [ord(sss)   <>   ord(ss)]   and   [SaveProbsOpt(idx,s)   >   0]   and
[SaveProbsOpt(idx,ss) > 0] and [ConflictPU(idx,sss) >= Threshold]] = yes;

);
display conflictsthree, LastIdx;


* CALCULATE FOR THE TRIPLETS
* Creating new solutions to save the conflicts
loop(conflictsthree(s,ss,sss),
    pAux = InitialProb(s)+InitialProb(ss)+InitialProb(sss);
    SaveProbsOpt(idx,s) $[ord(idx) = LastIdx+1] = InitialProb(s)/pAux;
    SaveProbsOpt(idx,ss)$[ord(idx) = LastIdx+1] = InitialProb(ss)/pAux;
    SaveProbsOpt(idx,sss)$[ord(idx) = LastIdx+1] = InitialProb(sss)/pAux;
    LastIdx = LastIdx+1;
);

* Now calculating

**** THIS CAN BE AN INCLUDE ******
loop(idx$[(ord(idx)> card(s)+1 + card(conflicts)) and (ord(idx)<=LastIdx)],

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;
    invest.fx(n,n,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l,
spill.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(ssss,
```

```
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(ssss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l,
opCost.l, spill.l;
        SaveSimVariable(idx,ssss)= opCost.l;
        SaveSimTotal(idx,ssss) = totalCost.l;

     );
     SaveSimExpectedVal(idx)     =     sum(ssss,     InitialProb(ssss)     *
SaveSimTotal(idx,ssss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$[(ord(idx)> card(s)+1 + card(conflicts)) and (ord(idx)<=LastIdx)],

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END
* OJO esTE LOOP LOS VALORES
display ConflictPU;
conflictsfour(s,ss,sss,ssss) = no;
loop(idx$ [[ord(idx)> card(s)+1 + card(conflicts)] and [ord(idx)<=LastIdx]],

    conflictsfour(s,ss,sss,ssss)$[[ord(ss)>ord(s)]   and   [ord(sss)>ord(ss)]
and [ord(ssss)<>ord(sss)] and [ord(ssss)<>ord(ss)] and [ord(ssss)<>ord(s)]
and     [SaveProbsOpt(idx,s)>0]     and     [SaveProbsOpt(idx,ss)>0]     and
[SaveProbsOpt(idx,sss)>0] and [ConflictPU(idx,ssss)>= Threshold]] = yes;

);
display conflictsfour, LastIdx;

*******************************************************************************
*
* CALCULATING FOR GROUPS OF FOUR
*******************************************************************************
*

* Creating new solutions to save the conflicts
loop(conflictsfour(s,ss,sss,ssss),
    pAux         =         InitialProb(s)+InitialProb(ss)+InitialProb(sss)+
InitialProb(ssss);
    SaveProbsOpt(idx,s) $[ord(idx) = LastIdx+1] = InitialProb(s)/pAux;
    SaveProbsOpt(idx,ss)$[ord(idx) = LastIdx+1] = InitialProb(ss)/pAux;
    SaveProbsOpt(idx,sss)$[ord(idx) = LastIdx+1] = InitialProb(sss)/pAux;
    SaveProbsOpt(idx,ssss)$[ord(idx) = LastIdx+1] = InitialProb(ssss)/pAux;
    LastIdx = LastIdx+1;
);
```

```
* Now calculating

**** THIS CAN BE AN INCLUDE ******
loop(idx$[(ord(idx)> card(s)+1 + card(conflicts) + card(conflictsthree)) and
(ord(idx)<=LastIdx)],

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;
    invest.fx(n,n,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l,
spill.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sssss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sssss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display  invest.l,  flow.l,  g_B.l,  nse.l,  totalCost.l,  invCost.l,
opCost.l, spill.l;
        SaveSimVariable(idx,sssss)= opCost.l;
        SaveSimTotal(idx,sssss) = totalCost.l;

     );
     SaveSimExpectedVal(idx)    =    sum(sssss,    InitialProb(sssss)    *
SaveSimTotal(idx,sssss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$[(ord(idx)> card(s)+1 + card(conflicts) + card(conflictsthree)) and
(ord(idx)<=LastIdx)],

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);
```

```
**** THIS CAN BE AN INCLUDE - END

conflictsfive(s,ss,sss,ssss,sssss) = no;
loop(idx$ [[ord(idx)> card(s)+1 + card(conflicts)+ card(conflictsthree)] and
[ord(idx)<=LastIdx]],

   conflictsfive(s,ss,sss,ssss,sssss)$[[ord(ss)>ord(s)]                  and
[ord(sss)>ord(ss)] and [ord(ssss)>ord(sss)] and [ord(ssss)<>ord(sssss)] and
[ord(sss)<>ord(sssss)] and [ord(ss)<>ord(sssss)] and [ord(s)<>ord(sssss)] and
[SaveProbsOpt(idx,s)>0]       and       [SaveProbsOpt(idx,ss)>0]       and
[SaveProbsOpt(idx,sss)>0]     and      [SaveProbsOpt(idx,ssss)>0]      and
[ConflictPU(idx,sssss)>= Threshold]] = yes;

);
display    ConflictPU,    SaveProbsOpt,    SaveSimTotal,    SaveSimVariable,
SaveSimExpectedVal, SaveOptInv, SaveOptTotal;
display conflicts, conflictsthree, conflictsfour, conflictsfive, LastIdx;
```

Explicación

Aunque el código de GAMS es punto de partida para este trabajo, es muy importante entender bien su funcionamiento para poder percibir el alcance de la presente investigación. La estructura del algoritmo, como se muestra en la Figura 1, es espiral. Vuelve con los mismos procedimientos subiendo de nivel para ir calculando las soluciones individuales, las semiestocásticas (por pares, ternas, cuaternas) y la estocástica. Y de esta manera poder calcular los posibles conflictos de las soluciones obtenidas con cada uno de los escenarios. Este cálculo, como se explicará más en detalle en su momento, se realizará aplicando las decisiones de inversión, líneas, etc. obtenidas al optimizar el coste para una combinación concreta de escenarios a cada uno de los escenarios de manera individual. De esta aplicación se obtendrá un nuevo valor para el coste y se comparará con el que era óptimo para ese escenario particular sin fijar las variables de decisión que era óptimas para otra combinación de escenarios. Pero, como ya se ha dicho, se explicará más adelante en el trabajo.

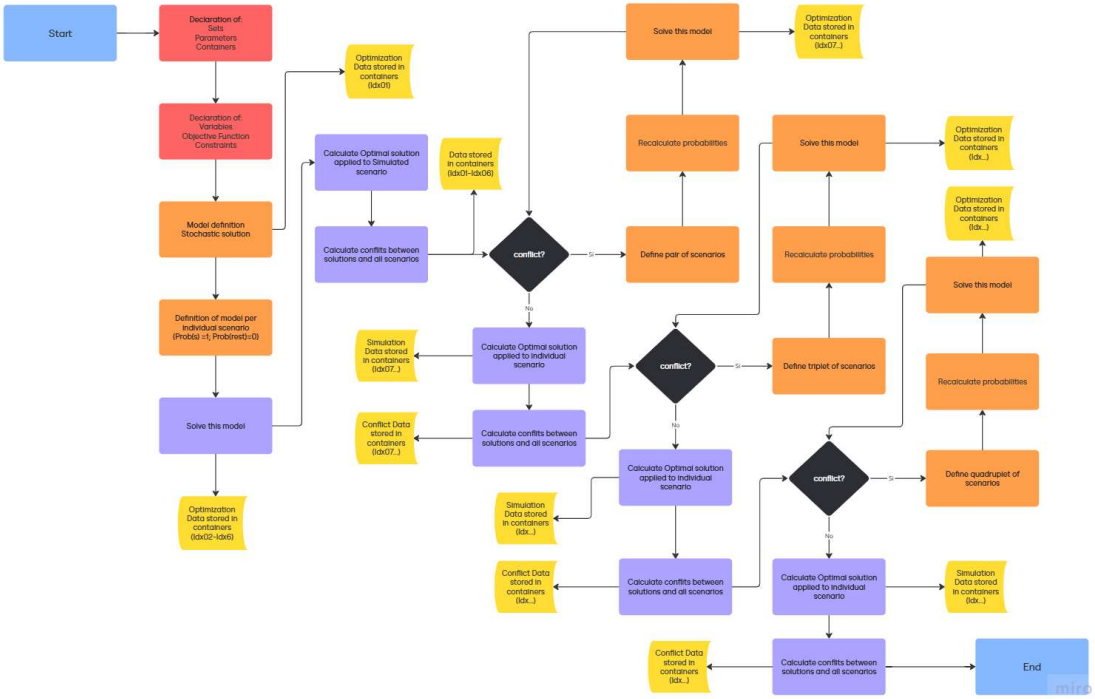


Figura 1: Flow diagram of the TEPTOY algorithm

A. Sets

Una parte muy importante de la programación en GAMS, es la declaración de los sets. En este caso, para el propósito que persigue este código, se declaran 9 sets, tres de los cuales tendrán alias: dos sets tendrán un alias y otro tendrá 4. Los alias se construyen para los casos en que sea necesario establecer relaciones de los sets consigo mismos sin que sean redundantes.

Los sets codificados corresponderán a:

- Nodos de la red (más un alias para las líneas de nodo a nodo)
- Escenarios posibles (más cuatro alias para las relaciones y conflictos entre escenarios)
- Tipos de líneas
- Soluciones de la programación estocástica (con un alias para los bucles que serán necesarios en el código)
- Enumerador de líneas posibles (de nodo a nodo y tipo). Aquí se añaden todas las opciones porque la variable Flow, que indica el flujo entre nodos se fuerza a que sea positiva, por lo que se debe definir el camino de n1 a n2 y viceversa.
- Enumerador de conflictos entre escenarios
- Enumerador de conflictos entre pareja de escenarios y escenarios
- Enumerador de conflictos entre terna de escenarios y escenarios
- Enumerador de conflictos entre cuaterna de escenarios y escenarios

B. Parameters

Los parámetros en GAMS son los elementos que contienen la información. Esta información puede tener dos propósitos:

- Información necesaria para la optimización

Estos parámetros contienen información que va a ser necesaria para las ecuaciones de las restricciones y la función objetivo

- Información obtenida de la optimización

  La información obtenida de la optimización está de forma "bruta" en las variables de decisión y el resto de las variables necesarias para la solución del modelo estocástico. Sin embargo, hay ocasiones como la que estudia este trabajo, en que se requiere ordenar y almacenar dicha información en parámetros de varias entradas, de forma que se pueda presentar dicha información y tratar con esos datos posteriormente a la optimización.

  Además, las variables van cambiando de valor en los distintos bucles necesarios para la resolución del modelo y será necesario conservar los valores obtenidos en cada bucle.

Por todo ello, se dividirá la declaración de los parámetros en dos secciones según el propósito que persiguen:

i. Parámetros con información previa a la solución del modelo:

Los parámetros que aparecen son:

- Potencia demandada en el nodo n para un escenario s → D(n,s)
- Potencia renovable generada en el nodo n para un escenario s → R(n,s)
- Potencia térmica disponible en el nodo n para un escenario s → B_cap(n,s)
- Coste de inversión para una línea de tipo t → InvestCost(t)
- Capacidad de transporte de energía de la línea t → Cap(t)
- Penalización por demanda no satisfecha → Penalty
- Precio por unidad de potencia térmica generada → Price_B
- Probabilidad de que se dé el escenario s → InitialProb(s)
- Umbral de tolerancia de conflictos medido en PU → Threshold

Todos estos parámetros, como se mostrará más adelante, contienen datos que será necesario usar en las ecuaciones de resolución del modelo a optimizar.

ii. Parámetros de almacenamiento

En estos, se almacenará la información de los resultados de la optimización, así como de los conflictos.

- Un parámetro de seguimiento del índice de la solución que se está construyendo o analizando → LastIdx
- Las probabilidades de los escenarios que se tendrán en cuenta para las soluciones particulares y poder explorar pares, ternas, etc. → Prob(s)
- Las probabilidades para cada escenario s con las que se ha estudiado una solución Idx del modelo → SaveProbsOpt(idx,s)
- Coste optimizado de implementar la solución Idx → SaveOptFixed(idx)
- Coste de inversión de cada línea entre los nodos n y m del tipo t en la solución Idx → SaveOptInv(idx,n,m,t)

- Coste variable optimizado de implantar la solución Idx → SaveOptVariable(idx)
- Coste total optimizado de implantar la solución Idx → SaveOptTotal(idx)
- Coste variable simulado para cada solución explorada aplicada a cada escenario → SaveSimVariable(idx,s)
- Coste total simulado para cada solución explorada aplicada a cada escenario → SaveSimTotal(idx,s)
- Valor esperado simulado para cada solución Idx → SaveSimExpectedVal(idx)
- Dato auxiliar para obtener el valor esperado → pAux
- Valoración del conflicto entre la solución Idx y el escenario s → Conflict(idx,s)
- Valoración en PU del conflicto entre la solución Idx y el escenario s en PU para poder compararlo con el umbral Threshold → ConflictPU(idx,s)
- Los parámetros a, b, c, cc se introdujeron para contar el número de conflictos que iban apareciendo en los pares, ternas, etc. Si se observa con atención se puede ver que no están en el código primero, sino que fueron añadidas en la depuración que se hizo del algoritmo en este trabajo.

C. Variables

Son aquellos elementos que el algoritmo puede cambiar hasta encontrar el valor óptimo del modelo.

Algunas de las variables sólo se han implementado para obtener datos que serán incorporados en los parámetros de almacenamiento de información.

- Se definen variables no negativas:
  - ✓ Una variable que contendrá el flujo de potencia del nodo n al m en el escenario s
  - ✓ Una variable que contendrá la potencia de reserva que se tendrá que generar en el nodo n en el escenario s
  - ✓ Una variable que contendrá la demanda de potencia no satisfecha
  - ✓ Una variable que contendrá el vertido que será necesario hacer en cada nodo y cada escenario con la potencia sobrante
  - ✓ Una variable que contendrá el coste de inversión
  - ✓ Una variable que contendrá el coste óptimo
- Se define también una variable booleana que contendrá información de si se invierte en la línea del tipo t del nodo n al m
- Finalmente, una variable del coste total que será la variable que se persigue optimizar en el proceso

D. Las ecuaciones que tendrá que cumplir el modelo de optimización:

Todo proceso de optimización se basa en el cumplimiento de unas ecuaciones. Estas se dividen en dos tipos:

La ecuación de optimización, que es la ecuación que se desea maximizar o minimizar, también llamada función objetivo.

Las restricciones que se tienen que cumplir, que se introducen en forma de ecuaciones y que marcarán los límites dentro de los cuales se desea obtener la mejor solución.

La función objetivo (1) se declara a partir del propósito del modelo que se desea optimizar. En este caso, el objetivo es minimizar el coste total. GAMS exige que dicha ecuación vaya asociada a la variable que se optimizará. Dicho coste total vendrá dado por la suma de las inversiones de poner una línea de tipo t entre los nodos n y m más la suma del precio de generar la potencia de reserva necesaria en el escenario s en el nodo n, más la suma de las penalizaciones por demanda de potencia no satisfecha en el nodo n en el escenario s. Por supuesto las variables y parámetros que dependen de los escenarios llevan aplicada la probabilidad de que se dé dicho escenario, ya que en ello consiste la programación estocástica.

$$\sum_{cand(n,m,t)} invest(n,m,t) * InvestCost(t) + \sum_{n,s} g\_B_{(n,s)} * Price\_B * Prob(s) + \sum_{n,s} nse(n,s) * Penalty * Prob(s) \quad (1)$$

La restricción de equilibrio de potencia implica que, en cada nodo, la potencia generada, tanto por renovables según su escenario como de reserva, más la potencia que entra por las líneas, tiene que ser igual a la potencia que sale por las líneas más la consumida. Esta última será la diferencia entre la demanda y la demanda no satisfecha.

Hay además un límite para las variables definidas. Las ecuaciones de estas restricciones son muy sencillas porque simplemente fijan un límite superior (el inferior está fijado porque las variables son no negativas).

- El flujo por cada línea no puede superar la capacidad de la línea o las líneas con las que se ha diseñado el sistema.
- La generación de energía térmica debe ser menor o igual que la capacidad de generación que tiene cada nodo en cada escenario
- La potencia demandada no satisfecha no puede exceder a la demanda por nodo para cada escenario
- Finalmente, el vertido no puede superar a la producción de energía renovable en cada nodo y para cada escenario

Se definen dos ecuaciones (invCostdef y opCostdef) mas con el único objetivo de obtener una información en unas variables (invCost y opCost) que luego se irá almacenando en los parámetros. Una contendrá el coste total de inversión; la otra el coste debido a la generación de reserva y a las posibles penalizaciones. La suma de ambas será el coste total optimizando el modelo.

Con todas estas ecuaciones se define el modelo NetworkExpansion que se irá resolviendo, cambiando los parámetros según las necesidades del problema.

E. Resolución del problema

En una primera ejecución, se pasa a optimizar dicho modelo sin cambiar ningún parámetro para obtener la solución estocástica global del problema, minimizando la variable de la función objetivo mediante programación entera mixta:

Se definen las probabilidades de los escenarios según el problema inicial y se resuelve la optimización.

Una vez conseguida la solución inicial, se pasa a plantear la optimización bajo distintas circunstancias. El primer planteamiento será buscar la solución óptima para nuestro modelo dado que sólo se diera un único escenario repitiendo esto mismo para cada uno de ellos, siempre salvando que sea cada escenario como si fuera el único posible. Para ello, en lugar de cambiar el

modelo, se cambiará el perfil de probabilidades asignando para cada solución una probabilidad de 1 a un escenario y 0 al resto. Se repetirá esta operación tantas veces como escenarios hay y así se conseguirá la solución óptima para cada escenario:

En estas líneas, se asignan unos perfiles de probabilidad al parámetro que contiene las probabilidades con las que se estudiarán las distintas variantes del modelo. Para ello se redefine el conjunto de sucesos de manera que sólo contenga los escenarios que se van a considerar en cada optimización, recalculando las probabilidades para cada combinación de escenarios. También se da un valor al contador de soluciones para poder enumerarlas.

Una vez hecho esto, se pasa a resolver cada uno de los modelos:

Este bucle lo primero que hace es ir resolviendo los modelos de optimización correspondientes a las seis primeras soluciones (Idx1 → Idx6) con sus respectivos perfiles de probabilidades para los distintos escenarios.

Primero vuelve a liberar la variable booleana invest antes de iniciar este bucle, porque más adelante en el bucle será fijada para poder aplicar la solución de cada conjunto de sucesos a cada uno de los escenarios de manera individual. Por esto es necesario liberarla al entrar en cada lazo del bucle. Luego va resolviendo. Una vez resuelto, almacena los valores de las variables en parámetros y así se podrá disponer de la información para el objetivo del algoritmo y del trabajo.

En este momento es cuando se fija la variable invest y se procede a resolver de nuevo el modelo para cada escenario particular, pero esta vez con la variable invest (la que dice si entre dos nodos hay línea y de qué tipo) fijada, de forma que el modelo en estudio sea óptimo. Para cada uno de estos escenarios, se almacena la información de las variables opCost (en SaveSimVariable) y totalCost (en SaveSimTotal). Así, el parámetro SaveSimVariable contiene, para cada solución (Idx), el valor del coste variable de cada escenario (s) con las líneas que resultaron óptimas para dicha solución (de ahí que se dejaran fijas); y el parámetro SaveSimTotal igual, pero con el coste total (inversión más variable) optimizado para cada modelo y cada solución con las líneas que salieron de resolver el modelo para la solución en cuestión.

Con toda esta información, y antes de salir del bucle de la solución que se está estudiando, se calcula el valor esperado del coste total de esta solución haciendo la media ponderada por las probabilidades de cada escenario del parámetro SaveSimTotal y se almacena en SaveSimExpectedVal.

Con esto se cierra el bucle y ya se tiene toda la información necesaria para las seis primeras soluciones: la estocástica y la particular de cada escenario (Idx1 → Idx6).

Una vez se tiene toda esta información de las soluciones para los cinco escenarios tomados de forma individual, se pueden calcular los conflictos. Para ello, se comparará el coste total simulado de cada solución y se comparará con la solución óptima de dicho conflicto. Para poder comparar unos con otros se decide, tal como se presenta en la ecuación (1), sacar un valor relativo en Pu dividiendo la diferencia entre el valor del coste total de una solución aplicado a un escenario y el valor óptimo de la solución teniendo en cuenta únicamente dicho escenario por este último. Para evitar que el valor de esta cuantificación del conflicto sea infinito (o NaN en los displays del código de GAMS), se le suma un valor muy pequeño 10^4 al dividendo.

$$ConflictPU_{(idx,s)} = \frac{SaveSimTotal_{(idx,s)} - SaveOptTotal_{(idxx)}}{SaveOptTotal_{(idxx)} + 0.0001} \quad (2)$$

La diferencia entre los índices Idx de ConflictPU Y SaveSimTotal con el Idxx (alias de Idx) de SaveOptTotal es porque el índice Idx es el que corresponde a la solución óptima para el escenario s tanto de ConflictPU, como de SaveSimTotal.

Una vez obtenido esta valoración del conflicto, se compara con el umbral que se ha establecido en los parámetros para establecer si hay o no conflicto entre una solución (Idx) y un escenario (s, cuya solución óptima individual se ha almacenado previamente también con índice Idx que se puede llamar con su alias Idxx para que GAMS no lo confunda con Idx).

Terminado este bucle, Se procede a subir un nivel en las soluciones a calcular, evaluando la red óptima para parejas de escenarios en conflicto. Para ello, primero se eligen los escenarios entre los que hay conflictos (o más propiamente, las soluciones a escenarios individuales que ofrecen conflictos aplicados a escenarios individuales). En el caso de que haya conflicto entre dos escenarios, se redefine el conjunto de sucesos escenarios para esos únicos dos escenarios en conflicto, se recalculan las probabilidades como se indica en las ecuaciones (2), (3) y (4). Las probabilidades del resto de escenarios se fijan en cero.

$$pAux \;=\; InitialProb(s) + InitialProb(ss); \tag{3}$$

$$SaveProbsOpt(idx, s)\;\$[ord(idx) \;=\; LastIdx + 1] \;=\; \frac{InitialProb(s)}{pAux}; \tag{4}$$

$$SaveProbsOpt(idx, ss)\$[ord(idx) = \; LastIdx + 1] = \frac{InitialProb(ss)}{pAux}; \tag{5}$$

Con este nuevo set de probabilidades calculado con el nuevo conjunto de sucesos posibles s y ss, se vuelve a calcular todo exactamente igual que se explicó para el bucle correspondiente a los conjuntos de un único escenario. Con esos cálculos, se obtiene el valor esperado de cada solución aplicada a cada uno de los escenarios y los conflictos con cada escenario, almacenando las respuestas en sus correspondientes parámetros contenedores para cada solución calculada.

Una vez hecho esto, se sube un nuevo nivel, evaluando la red óptima para ternas de escenarios en conflicto. Las ternas se obtienen de las parejas para las que se calculó todo el modelo y el escenario con el que entran en conflicto. Las nuevas probabilidades con las que se procederá a la optimización se obtienen de manera análoga a como se hizo en el bucle anterior como se puede ver en las ecuaciones (5), (6), (7) y (8).

$$pAux \;=\; InitialProb(s) + InitialProb(ss) + InitialProb(sss); \tag{6}$$

$$SaveProbsOpt(idx, s)\;\$[ord(idx) \;=\; LastIdx + 1] \;=\; \frac{InitialProb(s)}{pAux}; \tag{7}$$

$$SaveProbsOpt(idx, ss)\;\$[ord(idx) \;=\; LastIdx + 1] \;=\; \frac{InitialProb(ss)}{pAux}; \tag{8}$$

$$SaveProbsOpt(idx, sss)\ \$[ord(idx)\ =\ LastIdx + 1]\ =\ \frac{InitialProb(sss)}{pAux};\qquad (9)$$

Una vez calculadas, se procede exactamente igual que antes, almacenando la información para posterior análisis.

Se procede del mismo modo con las cuaternas de escenarios.

Así, una vez terminados todos los bucles, se han analizado 43 combinaciones de escenarios, dando lugar a un total de 94 conflictos.

III. El Código en lenguaje Python

Se optó por pasar el algoritmo ya explicado a un código de Python por la versatilidad que este entorno tiene para generar gráficos de todo tipo. Esto ha supuesto el mayor de los esfuerzos de este trabajo, ya que no se disponía de conocimientos previos de programación en este lenguaje. Pero esta es otra de las aportaciones más ricas de este trabajo a quien lo firma. Realmente, sólo el tiempo dedicado a implementar y codificar el algoritmo anterior en Python, justificaría más de las horas correspondientes a los créditos del TFG.

A continuación, se presenta el código completo para la resolución de conflictos en Python, que posteriormente se explicará de forma mucho más sucinta y sólo en lo que supone una diferencia con el algoritmo de GAMS:

Se ha querido simplificar el código lo máximo posible, como se ve en el diagrama de flujo de la Figura 2, sobre todo si se compara con el de la Figura 1.

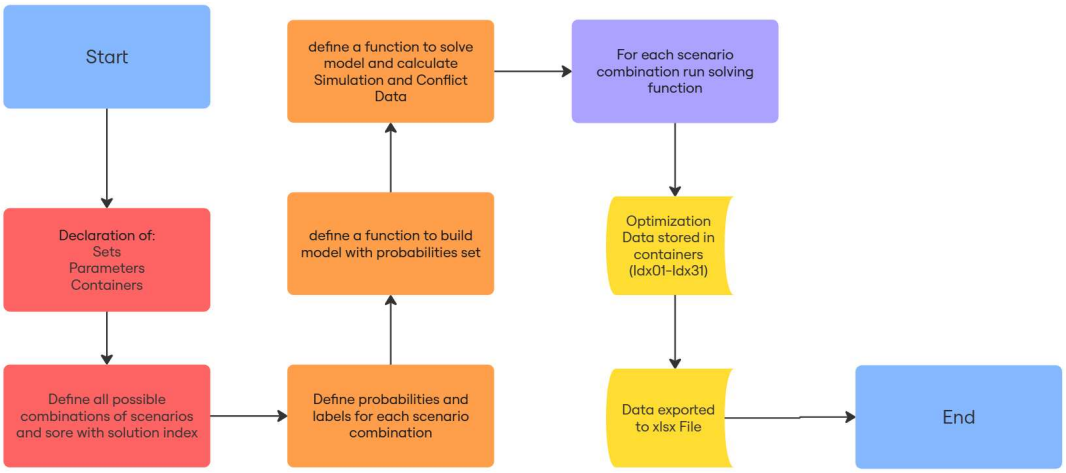


Figura 2: Flow diagram of the Python code

Se observa que el algoritmo de GAMS estudiaba 43 combinaciones diferentes de escenarios, cuando en realidad, sólo había 31 combinaciones diferentes:

$$1 + \frac{5!}{1! * 4!} + \frac{5!}{2! * 3!} + \frac{5!}{3! * 2!} + \frac{5!}{4! * 1!} = 1 + 5 + 10 + 10 + 5 = 31$$

- ⇨ 1 por la solución estocástica
- ⇨ 5 combinaciones de escenarios individuales
- ⇨ 10 combinaciones de escenarios tomados de dos en dos
- ⇨ 10 combinaciones de escenarios tomados de tres en tres
- ⇨ 5 combinaciones de escenarios tomados de cuatro en cuatro

Al estudiar conjuntamente solo las combinaciones que ofrecían conflictos sin verificar si la combinación ya había sido estudiada, se daban muchas duplicidades innecesarias y había combinaciones que no se estudiaban por no haber dado conflicto en una etapa del desarrollo. Se vio que sería más efectivo establecer todas las combinaciones posibles, estudiarlas y ver los resultados que ofrecían, tuvieran o no conflictos entre sí. Se consideró que también sería valiosa la

información que ofrecían las combinaciones que a priori no deberían ofrecer conflictos dado que no habían dado conflictos en etapas anteriores.

Para ello, se calculan todas las combinaciones posibles con sus sets de probabilidades asociadas y una etiqueta con los escenarios considerados para dicha solución. Una vez almacenados estos datos y asociados a un índice de solución Idx, se codifica una función para definir el modelo en función del set de probabilidades.

Con el modelo definido, se implementa una función para ejecutar la optimización con el modelo definido y para calcular todos los parámetros que serán necesarios para el propósito de este trabajo que ya se explicaron cuando se analizó el algoritmo de GAMS.

Finalmente, se almacenan todos estos datos en un archivo del cual, con otro código, se van a extraer todos los gráficos en una aplicación streamlit que permite una interactividad muy a propósito para este proyecto.

Como ya se ha indicado más arriba, en este punto, sólo se ha llegado al punto de partida de la aportación que este trabajo pretende conseguir. A pesar del ingente trabajo que ha costado, lo único que se ha buscado conseguir es una forma de tener la información necesaria para graficar la solución de la optimización estocástica y ofrecer a los decisores una herramienta para entender gráficamente los resultados de un proceso matemático sólo asequible para los expertos en programación estocástica. Sin embargo, se ha afrontado este esfuerzo ya que para un trabajo fin de grado, se pensó que tener que introducir los datos a mano, ofrecía poca seriedad. También es verdad que no se valoró con precisión la cantidad de trabajo que iba a costar habida cuenta del total desconocimiento de programación en Python del que se disponía.

Se puede ver el código resultante en el Anexo 1.B.

IV. Graficabilidad e interpretabilidad de la solución estocástica

Se entra aquí en la aportación nuclear de este trabajo: ofrecer una comprensión más intuitiva y visual de los resultados del proceso de optimización estocástica para los tomadores de decisiones, que no suelen ser especialistas en la materia. Los estadísticos enfrentan en muchos proyectos el grave problema de hacer el proceso comprensible para los no especialistas. Aquí se persigue el objetivo de desentrañar los entresijos de la solución estocástica. Para ello se utilizará la teoría de decisión "Multicriteria", y se aplicará a la programación estocástica, en la multiplicidad de los escenarios tenidos en cuenta.

Hasta ahora, como se ha presentado en la explicación del algoritmo y del código, se ha visto cómo el estudio de los conflictos es de vital importancia para la comprensión de lo que la solución estocástica no explica. Todo el presente trabajo intenta explotar la información de los conflictos entre soluciones semi-estocásticas que tienen en cuenta diversas combinaciones de escenarios y cada uno de los escenarios en solitario. Se denominan soluciones semi-estocásticas a las soluciones estocásticas parciales o que sólo tienen en cuenta combinaciones de más de un escenario y menos de la totalidad (a la que se llama solución estocástica).

Aunque la cuantificación y cualificación de estos conflictos es ya sumamente provechosa y esclarecedora, con el presente trabajo, se pretende dar un paso más allá y presentar dicha información de una forma gráfica y visual de manera que los decisores tengan una idea más intuitiva de lo que sucede dentro de los mecanismos de optimización estocástica. Para ello, se ha trabajado mucho en idear gráficos a partir de los datos obtenidos del código ya explicado y para reunir toda la información en una herramienta que permita tener todos los gráficos y los datos de una forma fácil, cómoda, intuitiva y a la vez potente. Dicha herramienta, "Dashboard.py" se presenta también en el Anexo 1.C. Esta herramienta es la verdadera aportación original de este TFG. Lo anterior, pese a la cantidad de trabajo que ha llevado No tanto como "Dashboard.py", pero casi), no deja de ser el punto de partida.

Con la información de dichos conflictos, ya sea información booleana (hay o no hay conflicto), que depende del umbral de conflicto que se haya querido imponer, ya sea información cuantificada en PU, se pretende ofrecer una comprensión de la solución global, asequible para cualquier persona no experta en estocástica. La intuición originaria es que, si se pueden cuantificar y cualificar los conflictos, se podrán graficar todos esos datos de manera que al decisor le resulte más intuitivo entender cuáles son los escenarios que dan más "problemas" (conflictos) y por tanto cuáles de los escenarios están exigiendo más recursos en la solución óptima final. De este modo, de una manera fácil sabrá cómo modificar los parámetros originales del problema (si se considera que se puede), conocedor como experto en redes de transporte de energía; o si puede asumir el riesgo de no tener en cuenta un escenario, sabiendo que es el que más está condicionando la solución obtenida por los expertos en programación estocástica, en las coordenadas iniciales del problema.

A. Las bases: estudio de los conflictos

Para esto, como se dijo más arriba, se emplea el código "Conflict research in Stochastic Programming". Este código, calcula las combinaciones que va a ser necesario resolver, les asigna un índice y calcula la distribución de probabilidades de cada combinación de escenarios (Tabla 1):

Scenario solution	s1	s2	s3	s4	s5
s1_s2_s3_s4_s5	0,2	0,2	0,2	0,2	0,2
s1	1	0	0	0	0
s2	0	1	0	0	0
s3	0	0	1	0	0
s4	0	0	0	1	0
s5	0	0	0	0	1
s1_s2	0,5	0,5	0	0	0
s1_s3	0,5	0	0,5	0	0
s1_s4	0,5	0	0	0,5	0
s1_s5	0,5	0	0	0	0,5
s2_s3	0	0,5	0,5	0	0
s2_s4	0	0,5	0	0,5	0
s2_s5	0	0,5	0	0	0,5
s3_s4	0	0	0,5	0,5	0
s3_s5	0	0	0,5	0	0,5
s4_s5	0	0	0	0,5	0,5
s1_s2_s3	0,333333	0,333333	0,333333	0	0
s1_s2_s4	0,333333	0,333333	0	0,333333	0
s1_s2_s5	0,333333	0,333333	0	0	0,333333
s1_s3_s4	0,333333	0	0,333333	0,333333	0
s1_s3_s5	0,333333	0	0,333333	0	0,333333
s1_s4_s5	0,333333	0	0	0,333333	0,333333
s2_s3_s4	0	0,333333	0,333333	0,333333	0
s2_s3_s5	0	0,333333	0,333333	0	0,333333
s2_s4_s5	0	0,333333	0	0,333333	0,333333
s3_s4_s5	0	0	0,333333	0,333333	0,333333
s1_s2_s3_s4	0,25	0,25	0,25	0,25	0
s1_s2_s3_s5	0,25	0,25	0,25	0	0,25
s1_s2_s4_s5	0,25	0,25	0	0,25	0,25
s1_s3_s4_s5	0,25	0	0,25	0,25	0,25
s2_s3_s4_s5	0	0,25	0,25	0,25	0,25

Tabla 1: Distribución de probabilidades de cada escenario en cada combinación estudiada.

Se han considerado escenarios equiprobables en los dos problemas que serán presentados, pero estos parámetros podrían cambiarse fácilmente en el código y el mismo código, calcularía la distribución con cualquier distribución inicial de probabilidades que se impusiere al modelo.

Para explicar el funcionamiento de la herramienta de análisis y *graficación* "Dashboard.py", se mostrarán los resultados de dos problemas distintos suponiendo la misma red. De uno a otro sólo se cambiarán los parámetros de la situación energética de cada nodo y la penalización por demanda no satisfecha.

B. Primera aproximación: parámetros iniciales (Problema 1)

La aplicación de todo lo anteriormente explicado, con la caracterización de los escenarios como se muestra en el anexo de gráficos (Figura 5 a la Figura 9), y los parámetros tal como estaban en el algoritmo originario, ofrece los conflictos como se presentan en la Tabla 2.

Scenario solution	s1	s2	s3	s4	s5
s1_s2_s3_s4_s5	1	0	0	1	1
s1	0	1	1	1	1
s2	1	0	0	1	1
s3	1	0	0	1	1
s4	1	1	1	0	0
s5	1	1	1	0	0
s1_s2	1	0	0	1	1
s1_s3	1	0	0	1	1
s1_s4	1	1	1	0	0
s1_s5	1	1	1	0	0
s2_s3	1	0	0	1	1
s2_s4	1	1	1	0	0
s2_s5	1	1	1	0	0
s3_s4	1	1	1	0	0
s3_s5	1	1	1	0	0
s4_s5	1	1	1	0	0
s1_s2_s3	1	0	0	1	1
s1_s2_s4	1	0	0	1	1
s1_s2_s5	1	0	0	1	1
s1_s3_s4	1	0	0	1	1
s1_s3_s5	1	0	0	1	1
s1_s4_s5	1	1	1	0	0
s2_s3_s4	1	0	0	1	1
s2_s3_s5	1	0	0	1	1
s2_s4_s5	1	1	1	0	0
s3_s4_s5	1	1	1	0	0
s1_s2_s3_s4	1	0	0	1	1
s1_s2_s3_s5	1	0	0	1	1
s1_s2_s4_s5	1	1	1	0	0
s1_s3_s4_s5	1	1	1	0	0
s2_s3_s4_s5	1	1	1	0	0

Tabla 2: Tabla de conflictos en la optimización del primer problema

Esta es la información bruta con la que se desea presentar una interpretación sencilla y gráfica de lo que está influyendo en la solución estocástica. La información cuantificada (el grado de conflicto en PU), se puede ver en un mapa de calor en la Figura 41. Se volverá sobre este gráfico más adelante.

Como ya se ha dicho, esta información es ya de mucho valor, mucho más que lo que ofrece la solución estocástica sola. El reto que asume este trabajo es presentarla de forma gráfica.

En este caso, las soluciones obtenidas de este problema para cada combinación de escenarios presentan unos valores asociados, tanto estocásticos como esperados, que se pueden observar en la Figura 3. Para el desafío que enfrenta este trabajo no es de mucha información. Pero da una idea de los costes que se están valorando. Las tablas de donde se han extraído los datos de este gráfico también se presentan en la Tabla 3 y la Tabla 4. El valor esperado, en una combinación de escenarios equiprobables, no da más información que la variación del coste óptimo cuando se aplica a una serie de escenarios conocidas sus probabilidades. Sin embargo, saber que el valor esperado se aparta mucho del valor óptimo de dicha solución, sí puede ser una información valiosa a tener en cuenta.

Scenario solution	Value
s1_s2_s3_s4_s5	46000
s1	69000
s2	46000
s3	46000
s4	47500
s5	49000
s1_s2	46000
s1_s3	46000
s1_s4	47500
s1_s5	49000
s2_s3	46000
s2_s4	47500
s2_s5	49000
s3_s4	47500
s3_s5	49000
s4_s5	47500
s1_s2_s3	46000
s1_s2_s4	46000
s1_s2_s5	46000
s1_s3_s4	46000
s1_s3_s5	46000
s1_s4_s5	47500
s2_s3_s4	46000
s2_s3_s5	46000
s2_s4_s5	47500
s3_s4_s5	47500
s1_s2_s3_s4	46000
s1_s2_s3_s5	46000
s1_s2_s4_s5	47500
s1_s3_s4_s5	47500
s2_s3_s4_s5	47500

Tabla 3: Expected value

Scenario solution	Value
s1_s2_s3_s4_s5	46000
s1	0
s2	20000
s3	27500
s4	47500
s5	62500
s1_s2	20000
s1_s3	23750
s1_s4	43750
s1_s5	51250
s2_s3	23750
s2_s4	43750
s2_s5	51250
s3_s4	47500
s3_s5	55000
s4_s5	55000
s1_s2_s3	22500
s1_s2_s4	39166,67
s1_s2_s5	41666,67
s1_s3_s4	41666,67
s1_s3_s5	44166,67
s1_s4_s5	50000
s2_s3_s4	41666,67
s2_s3_s5	44166,67
s2_s4_s5	50000
s3_s4_s5	52500
s1_s2_s3_s4	36250
s1_s2_s3_s5	38125
s1_s2_s4_s5	47500
s1_s3_s4_s5	49375
s2_s3_s4_s5	49375

Tabla 4: Optimal cost

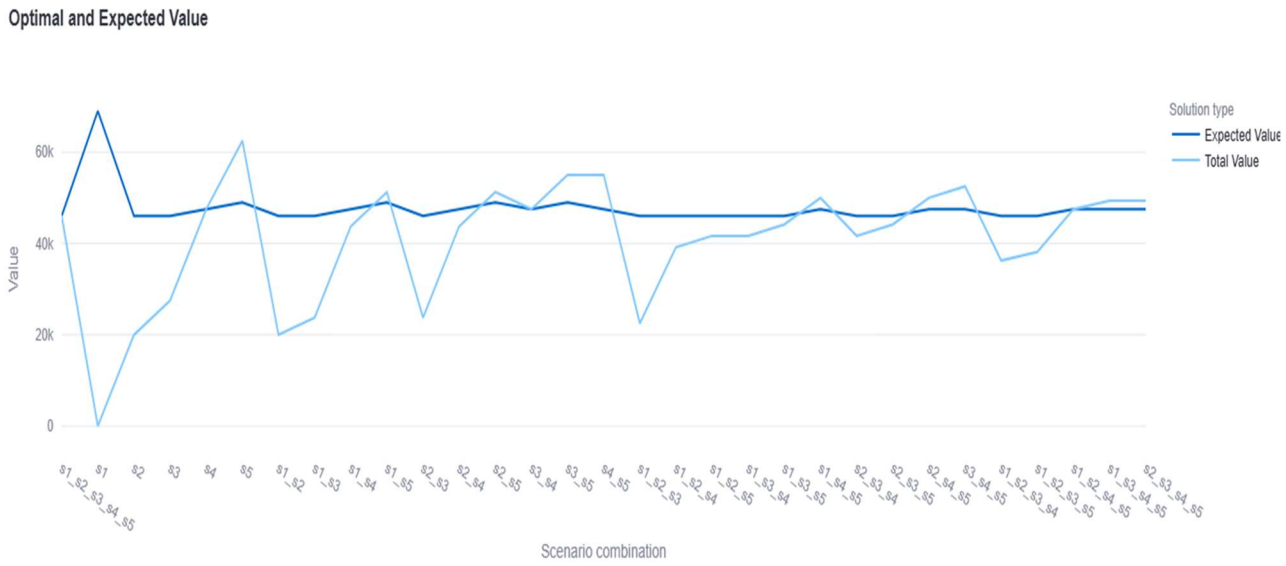


Figura 3: Coste total optimizado y Valor esperado del coste de una solución aplicado a los distintos escenarios

En las figuras del anexo entre la Figura 10 a la Figura 40 se puede ver de una forma directa la solución que se ha calculado óptima para la serie de escenarios de estudio. En la línea de este trabajo, pretende ofrecer una representación gráfica e intuitiva de la solución.

Con una idea clara de las condiciones de potencia de cada uno de los escenarios y de la decisión fundamental que caracteriza cada solución y de sus costes calculados, se presentan los gráficos que ofrecen información valiosa para los decisores. También se presentarán gráficos cuya inteligibilidad no está al alcance de los no versados en la estadística y la investigación operativa, ofrecen información muy valiosa para los especialistas. Estos gráficos, por estar fuera del propósito de este trabajo se han colocado en el apéndice.

Mapa de calor

En la Figura 41, se presenta un gráfico de mucho valor. En él se puede entender de una forma muy directa el núcleo de lo que este trabajo pretende, a saber: la confrontación entre las soluciones y cada uno de los escenarios. Se ha querido ofrecer en el mismo gráfico la información numérica y visual de manera que los decisores tengan en una misma impresión, la mayor cantidad de información posible. Los valores corresponden a la cuantificación de los conflictos en PU. Los colores del mapa de calor están asociados al umbral de 0,2 que se ha querido fijar como criterio de cualificación de los conflictos. Las celdas coloreadas son las que según este criterio tienen un grado de conflicto superior al admisible (más de un 20%). Las celdas en blanco son las que están dentro del umbral que se ha considerado “bueno”. Así, con las etiquetas de los escenarios incluidos en cada solución resulta muy fácil ver que el escenario 1 es el que más está exigente. Y que el resto tienen más o menos conflictividades parecidas.

Gráficos de Barras

A continuación, en las figuras de la Figura 42 a la Figura 46, se presentan agrupados por cantidad de escenarios incluidos en la combinación de estudio, cinco gráficos de barras (individuales, parejas, ternas, cuaternas y la totalidad de escenarios) en los que se cuantifica el número de conflictos que ofrece cada escenario. Lo que se busca es ver cómo evoluciona la cantidad de conflictos que ofrece cada escenario conforme se añade un escenario más a la solución. La cuantificación total de conflictos por escenario, si bien sí se ofrece en la aplicación streamlit de Dashboard.py, no se ha

presentado aquí, por no sobrecargar el trabajo dado que tampoco ofrece más información que la que ya se presenta. En esta sección de gráficos, se observa con mucha claridad que hay tres grupos claros de escenarios: el s1, el s2-s3 y el s4-s5. Seguramente, en los dendrogramas que se presentan en el apéndice, se verá que las distancias entre s2 y s3, por un lado, y entre s4 y s5 por otro, serán pequeñas.

Gráfico de Evolución de la conflictividad

Se considera que este gráfico, observable en la Figura 47, es el de mayor valor. No tiene un nombre normalizado porque es aportación original de la colaboración de este TFG. En él se puede ver qué combinaciones de escenarios son las más efectivas a la hora de reducir conflictos. Y efectivamente se observa que, como ya se pudo señalar en los gráficos de la sección anterior, hay dos familias de combinaciones, más o menos igual de efectivas en reducir conflictos: las que incluyen a los escenarios s4 y s5, y las que incluyen s2 y s3. Sin embargo, y debido a la diferencia tan grande que hay entre parámetros de los diferentes escenarios, ni siquiera la solución estocástica ofrece una reducción significativa de conflictos. Esto se debe a que la solución estocástica no es una solución global absoluta del problema, sino que incluye las probabilidades de los escenarios. Por eso hay escenarios incluidos tanto en la solución estocástica, como en las semi-estocásticas con los que dichas soluciones generan conflicto.

Cuando ya la parte de implementación de código y graficación estaba terminada, se descubrió la necesidad de ofrecer asociada a este mismo gráfico, información específica de cada una de las soluciones. Esto se debió a que, en el fondo, el mismo número de conflictos podía esconder soluciones idénticas en todo, o soluciones muy diversas. Pensando qué información ofrecer y cómo ofrecerla, se llegó a la conclusión de que el coste óptimo de cada solución, así como su valor esperado en función de la probabilidad con que se da cada escenario, daría una información valiosa a los decisores. El poder ver en un mismo gráfico, el orden en que van desapareciendo los conflictos en función de las combinaciones estudiadas, de más simples a más complejas, y los dos costes de cada solución, permite captar qué soluciones son iguales entre sí, a la vez que cuáles son más favorables económicamente para el constructor de la red en estudio. De ahí que se vea el gráfico sobrepuesto en dos matices de rojo (para remarcar con el contraste de colores la información que se quiere presentar), el gráfico simple del coste total de la decisión de inversión en la red y el del valor esperado dada la distribución de probabilidades entre los escenarios.

A la vista de toda la información, tanto visual como conceptual, que esta herramienta ofrece, se puede concluir que la inversión más beneficiosa es la que se extrae de la solución que tiene en cuenta únicamente al escenario s2. Si bien hay que tener en cuenta que debido a la incertidumbre con la que trabaja la estocástica (ni el valor esperado, ni el coste total son valores reales y objetivos, sino que maneja probabilidades de forma diferente: una antes de la optimización y la otra después), ninguna solución reduce más conflictos, ninguna solución tiene un valor esperado menor, y es la que tiene el menor coste total óptimo.

Se presenta aquí este resultado, sabiendo que no es el resultado del TFG, sino un caso ejemplo del funcionamiento de la herramienta que es el verdadero producto de este trabajo. Se ofrece en este punto para hacer ver el valor de la aplicación streamlit codificada. De ahí que no se hayan propuesto multitud de simulaciones diferentes, sino sólo dos. El propósito es dar una idea del funcionamiento y del valor de lo conseguido tras el esfuerzo realizado. De hecho, en la presentación se hará una demostración en vivo del funcionamiento de la herramienta para mostrar una interactividad que un trabajo estático como éste, no puede mostrar.

C. Segunda aproximación: parámetros cambiados (Problema 2)

En esta segunda aproximación, se han cambiado los parámetros de potencia de algunos de los escenarios. Se caracterizan como se muestra en las figuras de la Figura 48 a la Figura 52. Al igual que en el caso anterior, los resultados se exponen en la Tabla 5. Y del mismo modo se observa que, incluso tras este primer análisis, haría falta una especialización en teoría de la decisión multicriteria fuera del alcance de la población a la que se desea llegar con este trabajo.

Scenario solution	s1	s2	s3	s4	s5
s1_s2_s3_s4_s5	1	1	1	1	1
s1	0	1	1	1	1
s2	1	0	1	1	1
s3	1	1	0	1	1
s4	1	1	1	0	0
s5	1	1	1	0	0
s1_s2	1	0	1	1	1
s1_s3	1	1	0	1	0
s1_s4	1	1	1	0	0
s1_s5	1	1	0	1	0
s2_s3	1	1	0	1	1
s2_s4	1	1	1	0	0
s2_s5	1	1	1	0	0
s3_s4	1	1	0	0	0
s3_s5	1	1	0	1	0
s4_s5	1	1	1	0	0
s1_s2_s3	1	1	0	1	1
s1_s2_s4	1	1	1	0	0
s1_s2_s5	1	1	1	0	0
s1_s3_s4	1	1	0	0	0
s1_s3_s5	1	1	0	1	0
s1_s4_s5	1	1	1	0	0
s2_s3_s4	1	1	1	1	1
s2_s3_s5	1	1	0	1	0
s2_s4_s5	1	1	1	0	0
s3_s4_s5	1	1	0	0	0
s1_s2_s3_s4	1	1	1	1	1
s1_s2_s3_s5	1	1	0	1	0
s1_s2_s4_s5	1	1	1	0	0
s1_s3_s4_s5	1	1	0	0	0
s2_s3_s4_s5	1	1	1	1	1

Tabla 5: Conflict table for problem 2

Habiendo llegado hasta estos resultados, Dashboard.py nos ofrece, a partir de los datos contenidos en la Tabla 6 y en la Tabla 7 (extraídos del código de optimización estocástica), el gráfico de la Figura 4.

Scenario solution	Value
s1_s2_s3_s4_s5	76500
s1	1110500
s2	440500
s3	355000
s4	255000
s5	255000
s1_s2	440500
s1_s3	355000
s1_s4	255000
s1_s5	355000
s2_s3	355000
s2_s4	165000
s2_s5	165000
s3_s4	166500
s3_s5	355000
s4_s5	255000
s1_s2_s3	355000
s1_s2_s4	165000
s1_s2_s5	165000
s1_s3_s4	166500
s1_s3_s5	355000
s1_s4_s5	255000
s2_s3_s4	76500
s2_s3_s5	266500
s2_s4_s5	165000
s3_s4_s5	166500
s1_s2_s3_s4	76500
s1_s2_s3_s5	266500
s1_s2_s4_s5	165000
s1_s3_s4_s5	166500
s2_s3_s4_s5	76500

Tabla 6: Expected value

Scenario solution	Value
s1_s2_s3_s4_s5	76500
s1	7500
s2	37500
s3	62500
s4	62500
s5	62500
s1_s2	37500
s1_s3	55000
s1_s4	55000
s1_s5	55000
s2_s3	58750
s2_s4	65000
s2_s5	65000
s3_s4	72500
s3_s5	62500
s4_s5	62500
s1_s2_s3	55000
s1_s2_s4	62500
s1_s2_s5	62500
s1_s3_s4	67500
s1_s3_s5	57500
s1_s4_s5	57500
s2_s3_s4	77500
s2_s3_s5	70000
s2_s4_s5	67500
s3_s4_s5	72500
s1_s2_s3_s4	75000
s1_s2_s3_s5	66875
s1_s2_s4_s5	65000
s1_s3_s4_s5	68750
s2_s3_s4_s5	78750

Tabla 7: Optimal cost

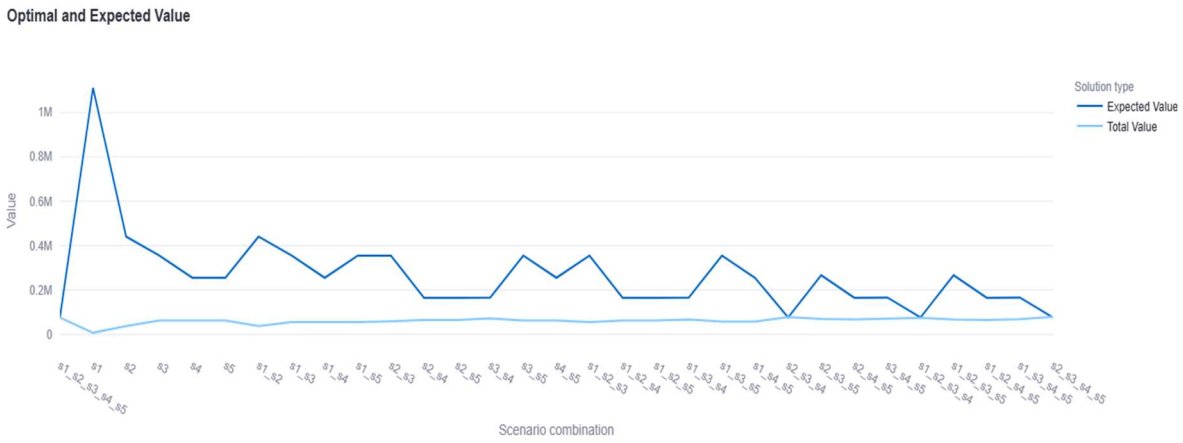


Figura 4: Coste total optimizado y Valor esperado del coste de una solución aplicado a los distintos escenarios

Como ya se ha dicho, los valores que se muestran en este gráfico no representan valores concretos y reales que puedan ayudar al decisor a comprender el problema, debido a la propia naturaleza de la programación estocástica. El coste total resulta de un proceso de optimización en el que todos los escenarios tenidos en cuenta tienen asociada una probabilidad en la misma variable de optimización (Ecuación 1). Y el valor esperado, con una red fijada ya por la optimización realizada, ofrece una suma ponderada por las correspondientes probabilidades de las aplicaciones de dicha red a cada uno de los escenarios. De ahí que sea necesario avanzar en la comprensión y explicación de lo obtenido en la optimización multi-escenario.

Del mismo modo que en el caso base, las soluciones se caracterizan como se muestra en los gráficos de la Figura 53 a la Figura 83. Se puede avanzar en la comprensión de todos estos datos con los recursos que ofrece esta herramienta.

Mapa de calor

El Mapa de calor de la Figura 84 muestra una conflictividad muy diferente a la del caso anterior. Al no haber ningún escenario con coste total estocástico de valor cero (como sucedía en el escenario 1 en el caso anterior), los conflictos no suben tanto de valor. Sin embargo, se ve a primera vista que no hay a penas conflictos “intermedios”. O son muy bajos o son muy altos. Esto ya nos da una idea de la solución que se está tratando de esclarecer. El escenario s1 sigue siendo el más problemático, pero el escenario s5 es bastante favorable. También se ve que no va a ser tan fácil establecer “familias de soluciones”. Llama la atención además que la solución estocástica, que es la que contempla todos los escenarios posibles, resulte conflictiva con todos y cada uno de los escenarios. Sólo alguien que conozca bien la programación estocástica y lo que en el proceso de resolución de este tipo de optimización puede entender esta eventualidad. De ahí que sea necesario profundizar en una explicación más intuitiva de los resultados de esta programación.

Gráficos de Barras

Los gráficos de barras de la Figura 85 a la Figura 89 confirman lo ya dicho. Los escenarios s1 y s2 son los que más están exigiendo a las distintas soluciones. Y tanto es así, que ni siquiera muchas de las que los incluyen en la combinación de estudio, dejan de tener conflictos con ellos. Sin embargo, el escenario s5 se señala por ser el menos conflictivo de todos con cierto margen.

Gráfico de Evolución de la Conflictividad

En la Figura 90 se presenta el gráfico de evolución de la conflictividad ordenado por el número de conflictos y de la sencillez de las soluciones. Como en el caso anterior, es un gráfico muy útil e intuitivo. En él se ve la complejidad de las soluciones (mayor cantidad de escenarios tenidos en cuenta) no ofrece garantías de mejor conflictividad. Las soluciones que más conflictos dan están entre las más complejas (incluida la solución estocástica). Y sin embargo la solución semi-estocástica que sólo tiene en cuenta los escenarios s3 y s4, parece ser la mejor. Si comparamos con el Mapa de Calor, se ve que no es tan sencilla la cosa porque, aunque el número de conflictos es menor, la cuantificación en PU de dichos conflictos, comparada por ejemplo con la de la solución estocástica, es más desfavorable. Aquí todo va a depender de lo fijo que consideremos el umbral establecido como criterio de discriminación de la conflictividad o no entre una solución y un escenario.

V. Graficabilidad Avanzada

Se añade un apéndice con unos gráficos que están en fuera de los límites del propósito de este TFG por la complejidad interpretativa de su naturaleza. Si bien están implementados en la herramienta Dashboard.py con una versatilidad que un trabajo escrito no puede reproducir. Y el objetivo de implementarlos y mostrarlos en apéndice (de la Figura 91 a la Figura 108) a este trabajo, aunque no sean comprensibles para los tomadores de decisiones, es que para los que desean hacerles asequible el proceso de optimización estocástica, pueden ser de gran ayuda.

Tanto los dendrogramas como los gráficos de reducción dimensional (Embedding diagrams) ofrecidos son potentes instrumentos para poder clasificar las soluciones de modo que se pueda dar un paso más en la comprensión de lo que sucede dentro de los procesos de programación estocástica. Queda para un trabajo ulterior el estudio de una clusterización a partir de la información que ofrecen dichos gráficos, que pueda ahondar en la interpretabilidad de la programación estocástica.

VI. Alineación con los Objetivos de desarrollo sostenible

La incertidumbre que introduce en el sistema la generación de energías renovables (que está detrás de las causas del apagón sucedido en España esta primavera), hace que los operadores de redes miren a este tipo de generación con verdadero temor. Ofrecer una herramienta que les ayude a comprender los resultados de un proceso estocástico de optimización puede ayudar a tener una mayor seguridad a la hora de poder implementar este tipo de programación y por tanto a arrostrar la incorporación de las energías renovables en las redes de transporte de energía.

- **ODS 7: Energía asequible y no contaminante**:

La mejora en la planificación de infraestructuras energéticas mediante programación estocástica que asuma la diversidad de escenarios en lo que a la generación renovable atañe, contribuye a un mayor desarrollo de las renovables y a una gestión más eficiente y sostenible de los recursos energéticos.

- **ODS 9: Industria, innovación e infraestructura**:

El desarrollo de nuevas metodologías para la interpretación de modelos de optimización y la mejora en la interpretabilidad de modelos estocásticos favorece el diseño de infraestructuras resilientes y mejora la infraestructura energética.

La capacidad de planificación fomenta la innovación y el desarrollo tecnológico.

- **ODS 13: Acción por el clima**:

Optimizar el transporte de energía tomando en cuenta la incertidumbre en la generación de energía renovable favorece la reducción de emisiones y el uso más eficiente de fuentes sostenibles y facilita la gestión e integración de estas energías verdes en la red con los consiguientes beneficios climáticos y energéticos.

Además, al optimizar el diseño de redes de transporte de energía, se reduce el impacto ambiental y se contribuye a la lucha contra el cambio climático.

VII. Conclusiones

Cuando se afrontó el reto de este trabajo, no se valoró bien la cantidad de trabajo ni la complejidad y dificultad del mismo. Tampoco se pretendió ofrecer una herramienta tan potente como la que finalmente se ha conseguido. El proyecto inicial, como se explicó en la presentación de la memoria explicativa del Anexo B, era poder pensar y presentar unos gráficos que ayudaran a la comprensión de la programación estocástica. Sin embargo, gracias a la audacia de la directora y a la inconsciencia del alumno, se ha hecho mucho más. Una vez depurado el algoritmo de inicio, se asumió el reto de aprender a programar en Python y con la ayuda auxiliar de la IA (ya que a tanto no dan los tiempos), obtener una herramienta que permite desentrañar los entresijos de la optimización estocástica. Si bien los agobios han estado presentes a lo largo de todo el proceso debido a las semanas enteras sin avance visible debido a la falta de experiencia en programación (en algún momento se pensó que ni el 31 de agosto sería suficiente plazo para poder tener la herramienta a tiempo), el resultado ha sido enormemente satisfactorio. Se experimenta en este punto un cierto sabor agridulce por no poder mostrar, si no es con el uso, la herramienta implementada en este trabajo.

A lo largo del camino se ha podido desarrollar la creatividad, la investigación, el apoyo de la IA, el aprendizaje, el trabajo duro y constante (ya se ha mencionado que se han invertido casi el triple de horas de las 150 que la universidad exige para los TFG). Pero con sorpresa y asombro, se ha podido llegar a este resultado. Las presentes páginas han pretendido dar testimonio de un trabajo Holístico.

Queda para otro trabajo, como ya se insinuó más arriba (quizá el TFM), el estudio de los conflictos de todas las soluciones con todas las soluciones y no sólo los escenarios en particular. Se considera que dicho estudio puede arrojar más información valiosa para el mismo propósito de este trabajo. También la profundización en las potencialidades del Clustering, se juzga que podrían ser muy provechosas para seguir adelante en la presente investigación. Finalmente, seguir tirando del hilo de la evolución de las soluciones en función de los conflictos y cómo afecta la presencia o ausencia de un escenario o una pareja de escenarios en la evolución de la conflictividad, puede ser también una línea interesante de desarrollo.

VIII. Anexos

Anexo 1: Códigos

Anexo 1.A: Algoritmo original

```
* ---------------------------------------------
* GAMS Model: Network Expansion Optimization with 3 Nodes and 5 Scenarios
* Objective: Minimize total investment and operational costs
* ---------------------------------------------
$onListing

Sets
    n   Nodes /n1, n2, n3/
    s   Scenarios /s1, s2, s3, s4, s5/
    t   Types /t1, t2/
    idx index solution /idx01*idx99/
;
alias (n,m), (s,ss,sss,ssss,sssss), (idx,idxx)
;
sets
    cand(n,m,t) Corridors between nodes n and m of type t
        /
        n1.n2.t1, n1.n2.t2,
        n1.n3.t1, n1.n3.t2,
        n2.n3.t1, n2.n3.t2
        /

    conflicts(s,ss)
    conflictsthree(s,ss,sss)
    conflictsfour(s,ss,sss,ssss)

;

* Parameters Definition
Parameter
    D(n,s)          Demand at node n under scenario s
        /
        n1.s1 100, n1.s2 0,    n1.s3 100,  n1.s4 0,    n1.s5 100,
        n2.s1 0,   n2.s2 100,  n2.s3 100,  n2.s4 100,  n2.s5 100,
        n3.s1 0,   n3.s2 0,    n3.s3 0,    n3.s4 100,  n3.s5 100
        /,

    R(n,s)          Renewable generation at node n under scenario s
        /
        n1.s1 150, n1.s2 150, n1.s3 150, n1.s4 150, n1.s5 150,
        n2.s1 0,   n2.s2 0,   n2.s3 0,   n2.s4 0,   n2.s5 0,
        n3.s1 0,   n3.s2 0,   n3.s3 0,   n3.s4 0,   n3.s5 0
        /,
    B_cap(n,s)      Backup generation capacity at node n under scenario s
        /
        n1.s1 50, n1.s2 50, n1.s3 50, n1.s4 50, n1.s5 50,
        n2.s1 50, n2.s2 50, n2.s3 50, n2.s4 50, n2.s5 50,
        n3.s1 50, n3.s2 50, n3.s3 50, n3.s4 50, n3.s5 50
        /,
    InvestCost(t)    Investment cost for corridor type t
        /
        t1 20000,
        t2 30000
        /,
    Cap(t)          Capacity for corridor type t
```

```
        /
        t1 100,
        t2 200
        /,
     Penalty         Penalty cost for NSE
        /1000/
    ,
    Price_B         Price per unit of backup generation (single scalar)
        /150/
    ,
    InitialProb(s)         Scenario probability
        /s1 0.2, s2 0.2, s3 0.2, s4 0.2, s5 0.2/
    ,


    Threshold         Threshold for conflict
        /0.20/

    LastIdx   last index

    Prob(s)   scenario probability to use


    SaveProbsOpt(idx,s) probabilities in optimization
    SaveOptFixed(idx) investment cost optimized
    SaveOptInv(idx,n,m,t) investment variables
    SaveOptVariable(idx) variable cost optimized
    SaveOptTotal(idx) total cost optimized
    SaveSimVariable(idx,s) variable cost simulated for each explored solution
    SaveSimTotal(idx,s) total cost simulated
    SaveSimExpectedVal(idx) expected value simulated
    pAux

    Conflict(idx,s) conflict
    ConflictPU(idx,s) conflict in pu

     ;

* Variables Definition
Positive Variables
    flow(n,m,s)      Power flow through corridor
    g_B(n,s)         Backup generation
    nse(n,s)         Non-Served Energy
    invCost
    opCost
;

Binary Variable
    invest(n,m,t)    Capacity invested in corridor
;

Variable
    totalCost;

* Equations Definition
Equations
    obj                  * Objective function
    powerBalance(n,s)    * Power balance at each node and scenario
    flowLimit(n,m,s)     * Flow limits based on invested corridor capacities
    backupLimit(n,s)     * Backup generation limits
    invCostdef           * investment cost definition
    opCostdef            * operation  cost definition
;

* Objective Function: Minimize Investment + Operational Costs
```

```
obj..
    totalCost =e=
        sum((n,m,t)$(cand(n,m,t)), invest(n,m,t) * InvestCost(t)) +
        sum((n,s), g_B(n,s) * price_B * Prob(s)) +
        sum((n,s), nse(n,s) * Penalty * Prob(s));

* Power Balance Constraint
powerBalance(n,s)..
    R(n,s) + g_B(n,s) + sum(m, flow(m,n,s)) -
    sum(m, flow(n,m,s)) =L= D(n,s) + nse(n,s);

* Flow Limits Constraint: Flow <= Sum of Invested Capacity across Types
flowLimit(n,m,s)..
    flow(n,m,s) =L= sum(t, invest(n,m,t)*Cap(t));

* Backup Generation Limit Constraint
backupLimit(n,s)..
    g_B(n,s) =L= B_cap(n,s);

invCostdef..
    invCost =e= sum((n,m,t)$(cand(n,m,t)), Invest(n,m,t) * InvestCost(t));

opCostdef..
     opCost =e= sum((n,s), g_B(n,s) * price_B * Prob(s)) +
        sum((n,s), nse(n,s) * Penalty * Prob(s));

* Model Definition
Model NetworkExpansion /all/;

**********************************************************************
* INITIAL EXECUTION
**********************************************************************

* First test execution
Prob(s) = InitialProb(s);
conflicts(s,ss) = no;

Solve NetworkExpansion minimizing totalCost using MIP;
Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;


**********************************************************************
* CALCULATION PER INDIVIDUAL SCENARIO
**********************************************************************

* First we will calculate the initial solution again, then we will optimize each scenario
separately
SaveProbsOpt('idx01',s) = InitialProb(s);
SaveProbsOpt(idx,s)$[(ord(idx)>1) and (ord(idx)<=card(s)+1)]= 0+1$[ord(s)=ord(idx)-1];
LastIdx = card(s)+1;

**** THIS CAN BE AN INCLUDE, IT WILL ALWAYS BE THE SAME ******
loop(idx$(ord(idx)<=LastIdx),

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
```

```
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;
        SaveSimVariable(idx,sss)= opCost.l;
        SaveSimTotal(idx,sss) = totalCost.l;

     );
     SaveSimExpectedVal(idx) = sum(sss, InitialProb(sss) * SaveSimTotal(idx,sss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$(ord(idx)<=LastIdx),

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END

* Calculating conflict pairs
conflicts(s,ss) = no;
conflicts(s,ss)$[[ord(s)< ord(ss)] and [sum(idx$[ord(idx)= ord(s)+1],ConflictPU(idx,ss))]
>= Threshold] = yes;
display conflicts;


*********************************************************************
* CALCULATION OF SOLUTIONS WITH TWO SCENARIOS
*********************************************************************

* Creating new solutions to solve the conflicts between pairs of scenarios

loop(conflicts(s,ss),
    pAux = InitialProb(s)+InitialProb(ss);
    SaveProbsOpt(idx,s) $[ord(idx) = LastIdx+1] = InitialProb(s)/pAux;
    SaveProbsOpt(idx,ss)$[ord(idx) = LastIdx+1] = InitialProb(ss)/pAux;

    LastIdx = LastIdx+1;
);

* Evaluating them.
**** THIS CAN BE AN INCLUDE ******
loop(idx$[(ord(idx)> card(s)+1) and (ord(idx)<=LastIdx)],

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
```

```
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;
        SaveSimVariable(idx,sss)= opCost.l;
        SaveSimTotal(idx,sss) = totalCost.l;

     );
     SaveSimExpectedVal(idx) = sum(sss, InitialProb(sss) * SaveSimTotal(idx,sss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$[(ord(idx)> card(s)+1) and (ord(idx)<=LastIdx)],

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END
display ConflictPU;
conflictsthree(s,ss,sss) = no;
loop(idx$ [[ord(idx)> card(s)+1] and [ord(idx)<=LastIdx]],

    conflictsthree(s,ss,sss)$[[ord(s)<    ord(ss)]    and    [ord(sss)>ord(ss)]    and
[SaveProbsOpt(idx,s)>0]   and   [SaveProbsOpt(idx,ss)>0]   and   [ConflictPU(idx,sss)>=
Threshold]] = yes;

);
display conflictsthree;


* CALCULATE FOR THE TRIPLETS
* Creating new solutions to save the conflicts
loop(conflictsthree(s,ss,sss),
    pAux = InitialProb(s)+InitialProb(ss)+InitialProb(sss);
    SaveProbsOpt(idx,s) $[ord(idx) = LastIdx+1] = InitialProb(s)/pAux;
    SaveProbsOpt(idx,ss)$[ord(idx) = LastIdx+1] = InitialProb(ss)/pAux;
    SaveProbsOpt(idx,sss)$[ord(idx) = LastIdx+1] = InitialProb(sss)/pAux;
    LastIdx = LastIdx+1;
);

* Now calculating

**** THIS CAN BE AN INCLUDE ******
```

```
loop(idx$[(ord(idx)> card(s)+1 + card(conflicts)) and (ord(idx)<=LastIdx)],

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;
        SaveSimVariable(idx,sss)= opCost.l;
        SaveSimTotal(idx,sss) = totalCost.l;

     );
     SaveSimExpectedVal(idx) = sum(sss, InitialProb(sss) * SaveSimTotal(idx,sss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$[(ord(idx)> card(s)+1 + card(conflicts)) and (ord(idx)<=LastIdx)],

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END
* OJO esTE LOOP LOS VALORES
display ConflictPU;
conflictsfour(s,ss,sss,ssss) = no;
loop(idx$ [[ord(idx)> card(s)+1 + card(conflicts)] and [ord(idx)<=LastIdx]],

    conflictsfour(s,ss,sss,ssss)$[[ord(ss)>ord(s)]     and     [ord(sss)>ord(ss)]     and
[ord(ssss)>ord(sss)]   and   [SaveProbsOpt(idx,s)>0]   and   [SaveProbsOpt(idx,ss)>0]   and
[SaveProbsOpt(idx,sss)>0] and [ConflictPU(idx,ssss)>= Threshold]] = yes;

);
display conflictsfour;

*****************************************************************************
* CALCULATING FOR GROUPS OF FOUR
*****************************************************************************

* Creating new solutions to save the conflicts
loop(conflictsfour(s,ss,sss,ssss),
```

```
    pAux = InitialProb(s)+InitialProb(ss)+InitialProb(sss)+ InitialProb(ssss);
    SaveProbsOpt(idx,s) $[ord(idx) = LastIdx+1] = InitialProb(s)/pAux;
    SaveProbsOpt(idx,ss)$[ord(idx) = LastIdx+1] = InitialProb(ss)/pAux;
    SaveProbsOpt(idx,sss)$[ord(idx) = LastIdx+1] = InitialProb(sss)/pAux;
    SaveProbsOpt(idx,ssss)$[ord(idx) = LastIdx+1] = InitialProb(ssss)/pAux;
    LastIdx = LastIdx+1;
);

* Now calculating

**** THIS CAN BE AN INCLUDE ******
loop(idx$[(ord(idx)>   card(s)+1   +   card(conflicts)   +   card(conflictsthree))   and
(ord(idx)<=LastIdx)],

* Freeing investment variables
    invest.up(n,m,t) = 1;
    invest.lo(n,m,t) = 0;

    Prob(s) = SaveProbsOpt(idx,s);
    Solve NetworkExpansion minimizing totalCost using MIP;

    Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;

* Saving information on the optimization
    SaveOptFixed(idx) = invCost.l;
    SaveOptInv(idx,n,m,t)= invest.l(n,m,t);
    SaveOptVariable(idx) = opCost.l;
    SaveOptTotal(idx)= totalCost.l;

* Simulating for each scenario
* Fixing investment decisions
    invest.fx(n,m,t)= invest.l(n,m,t);

    loop(sss,
        Prob(s) = 0;
        Prob(s)$[ord(s)=ord(sss)] = 1;


        Solve NetworkExpansion minimizing totalCost using MIP;
        Display invest.l, flow.l, g_B.l, nse.l, totalCost.l, invCost.l, opCost.l;
        SaveSimVariable(idx,sss)= opCost.l;
        SaveSimTotal(idx,sss) = totalCost.l;

     );
     SaveSimExpectedVal(idx) = sum(sss, InitialProb(sss) * SaveSimTotal(idx,sss));
);

* Calculating conflicts between solutions and scenarios
loop(idx$[(ord(idx)>   card(s)+1   +   card(conflicts)   +   card(conflictsthree))   and
(ord(idx)<=LastIdx)],

    loop(s,
        pAux = sum(idxx$[ord(idxx) = ord(s)+1], SaveOptTotal(idxx));

        Conflict(idx,s) = SaveSimTotal(idx,s)-pAux;
        ConflictPU(idx,s) = SaveSimTotal(idx,s)/[pAux+0.0001] -1;
    );
);

**** THIS CAN BE AN INCLUDE - END

conflictsfour(s,ss,sss,sss) = no;
loop(idx$   [[ord(idx)>   card(s)+1   +   card(conflicts)+   card(conflictsthree)]   and
[ord(idx)<=LastIdx]],
```

```
   conflictsfour(s,ss,sss,ssss)$[[ord(ss)>ord(s)]     and     [ord(sss)>ord(ss)]     and
[ord(ssss)>ord(sss)]   and   [SaveProbsOpt(idx,s)>0]   and   [SaveProbsOpt(idx,ss)>0]   and
[SaveProbsOpt(idx,sss)>0] and [ConflictPU(idx,ssss)>= Threshold]] = yes;

);
display   ConflictPU,   SaveProbsOpt,   SaveSimTotal,   SaveSimExpectedVal,   SaveOptInv,
SaveOptTotal;
display conflicts, conflictsthree, conflictsfour;
```

Anexo 1.B: Código del algoritmo original en Python: “Conflict research in Stochastic Programming”

```python
import pyomo.environ as pye
import pandas as pd
import itertools
import subprocess
import numpy as np

# --- Sets ---
nodes = ['n1', 'n2', 'n3']
scenarios = ['s1', 's2', 's3', 's4', 's5']
types = ['t1', 't2']

idx_solutions = ['idx01'] + [f'idx{i+2:02}' for i in range(len(scenarios))]

# Type t Lines between nodes n y m
cand = [
    ('n1', 'n2', 't1'), ('n1', 'n2', 't2'),
    ('n1', 'n3', 't1'), ('n1', 'n3', 't2'),
    ('n2', 'n3', 't1'), ('n2', 'n3', 't2'),
    ('n2', 'n1', 't1'), ('n2', 'n1', 't2'),
    ('n3', 'n1', 't1'), ('n3', 'n1', 't2'),
    ('n3', 'n2', 't1'), ('n3', 'n2', 't2')
]
flowset = [
    ('n1', 'n2', 's1'), ('n1', 'n2', 's2'), ('n1', 'n2', 's3'), ('n1', 'n2', 's4'),
('n1', 'n2', 's5'),
    ('n1', 'n3', 's1'), ('n1', 'n3', 's2'), ('n1', 'n3', 's3'), ('n1', 'n3', 's4'),
('n1', 'n3', 's5'),
    ('n2', 'n3', 's1'), ('n2', 'n3', 's2'), ('n2', 'n3', 's3'), ('n2', 'n3', 's4'),
('n2', 'n3', 's5'),
    ('n2', 'n1', 's1'), ('n2', 'n1', 's2'), ('n2', 'n1', 's3'), ('n2', 'n1', 's4'),
('n2', 'n1', 's5'),
    ('n3', 'n1', 's1'), ('n3', 'n1', 's2'), ('n3', 'n1', 's3'), ('n3', 'n1', 's4'),
('n3', 'n1', 's5'),
    ('n3', 'n2', 's1'), ('n3', 'n2', 's2'), ('n3', 'n2', 's3'), ('n3', 'n2', 's4'),
('n3', 'n2', 's5')
]

# --- Parameters ---
data = {
    'D': pd.DataFrame([
        ['n1', 's1', 150], ['n1', 's2', 0], ['n1', 's3', 150], ['n1', 's4', 0],
['n1', 's5', 100],
```

```
        ['n2', 's1', 0], ['n2', 's2', 150], ['n2', 's3', 150], ['n2', 's4', 150],
['n2', 's5', 100],
         ['n3', 's1', 0], ['n3', 's2', 0], ['n3', 's3', 0], ['n3', 's4', 150],
['n3', 's5', 100]
    ], columns=['n', 's', 'value']).set_index(['n', 's']),

    'R': pd.DataFrame([
         ['n1', 's1', 150], ['n1', 's2', 150], ['n1', 's3', 150], ['n1', 's4',
150], ['n1', 's5', 150],
        ['n2', 's1', 0], ['n2', 's2', 0], ['n2', 's3', 0], ['n2', 's4', 0], ['n2',
's5', 0],
        ['n3', 's1', 0], ['n3', 's2', 0], ['n3', 's3', 0], ['n3', 's4', 0], ['n3',
's5', 0]
    ], columns=['n', 's', 'value']).set_index(['n', 's']),

    'B_cap': pd.DataFrame([
         ['n1', 's1', 50], ['n1', 's2', 50], ['n1', 's3', 50], ['n1', 's4', 50],
['n1', 's5', 50],
         ['n2', 's1', 50], ['n2', 's2', 50], ['n2', 's3', 50], ['n2', 's4', 50],
['n2', 's5', 50],
         ['n3', 's1', 50], ['n3', 's2', 50], ['n3', 's3', 50], ['n3', 's4', 50],
['n3', 's5', 50]
    ], columns=['n', 's', 'value']).set_index(['n', 's']),

    'InvestCost': {'t1': 20000, 't2': 30000},
    'Cap': {'t1': 100, 't2': 200},
    'Penalty': 10000,
    'Price_B': 150,
    'InitialProb': {'s1': 0.2, 's2': 0.2, 's3': 0.2, 's4': 0.2, 's5': 0.2},
    'Threshold': 0.20
}

# Converts parameters to dictionary lookup for easier integration in Pyomo
D = data['D']['value'].to_dict()
R = data['R']['value'].to_dict()
B_cap = data['B_cap']['value'].to_dict()
InvestCost = data['InvestCost']
Cap = data['Cap']
Penalty = data['Penalty']
Price_B = data['Price_B']
InitialProb = data['InitialProb']
Threshold = data['Threshold']

# Containers
SaveOptFixed = pd.Series(0.0, index=idx_solutions)
SaveOptInv = pd.DataFrame(0.0, index=idx_solutions, columns=cand)
SaveFlux = pd.DataFrame(0.0, index=idx_solutions, columns=flowset)
SaveOptVariable = pd.Series(0.0, index=idx_solutions)
SaveOptTotal = pd.Series(0.0, index=idx_solutions)
SaveSimVariable = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)
SaveSimTotal = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)
SaveSimExpectedVal = pd.Series(0.0, index=idx_solutions)
probsglobal = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)
label = pd.Series('s1_s2_s3_s4_s5', index=idx_solutions)
Inv = pd.DataFrame(0.0, index=pd.MultiIndex.from_product([nodes, nodes, types],
names=['n', 'm', 't']),columns=['value'])
```

```
Conflict = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)
ConflictPU = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)

conflicts = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)


for s in scenarios:
    probsglobal.loc['idx01'] = pd.Series(InitialProb)

for i, scenario_to_scene in enumerate(scenarios):
    current_idx_label = f'idx{i+2:02}'
    if current_idx_label not in idx_solutions:
        idx_solutions.append(current_idx_label)
        probsglobal = probsglobal.reindex(idx_solutions)
        label = label.reindex(idx_solutions)
    for s in scenarios:
            probsglobal.loc[current_idx_label] = pd.Series({s: (1 if s ==
scenario_to_scene else 0) for s in scenarios})
    label.loc[current_idx_label] = scenario_to_scene

new_solutions_to_add = []
for s_pair in itertools.combinations(scenarios, 2):
    s1, s2 = s_pair
    new_solutions_to_add.append(tuple(sorted([s1, s2])))


new_solutions_to_add = sorted(list(set(new_solutions_to_add)))

current_solutions_idx_start = len(idx_solutions)

for pares in new_solutions_to_add:
    s_label = "_".join(pares)
    current_idx_label = f'idx{len(idx_solutions)+1:02}'
    idx_solutions.append(current_idx_label)

    probsglobal = probsglobal.reindex(idx_solutions)
    label = label.reindex(idx_solutions)

    p_aux_sum = sum(InitialProb[s_item] for s_item in pares)
    probsglobal.loc[current_idx_label] = pd.Series({s: (InitialProb[s] / p_aux_sum
if s in pares else 0) for s in scenarios})
    label.loc[current_idx_label] = s_label

new_solutions_to_add = []
for triplet in itertools.combinations(scenarios, 3):
    s1, s2, s3 = triplet
    new_solutions_to_add.append(tuple(sorted([s1, s2, s3])))


new_solutions_to_add = sorted(list(set(new_solutions_to_add)))

current_solutions_idx_start = len(idx_solutions)

for triplet in new_solutions_to_add:
    s_label = "_".join(triplet)
```

```
    current_idx_label = f'idx{len(idx_solutions)+1:02}'
    idx_solutions.append(current_idx_label)

    probsglobal = probsglobal.reindex(idx_solutions)
    label = label.reindex(idx_solutions)

    p_aux_sum = sum(InitialProb[s_item] for s_item in triplet)
    probsglobal.loc[current_idx_label] = pd.Series({s: (InitialProb[s] / p_aux_sum
if s in triplet else 0) for s in scenarios})
    label.loc[current_idx_label] = s_label

new_solutions_to_add = []
for quadruplet in itertools.combinations(scenarios, 4):
    s1, s2, s3, s4 = quadruplet
    new_solutions_to_add.append(tuple(sorted([s1, s2, s3, s4])))


new_solutions_to_add = sorted(list(set(new_solutions_to_add)))

current_solutions_idx_start = len(idx_solutions)

for quadruplet in new_solutions_to_add:
    s_label = "_".join(quadruplet)
    current_idx_label = f'idx{len(idx_solutions)+1:02}'
    idx_solutions.append(current_idx_label)

    probsglobal = probsglobal.reindex(idx_solutions)
    label = label.reindex(idx_solutions)

    p_aux_sum = sum(InitialProb[s_item] for s_item in quadruplet)
    probsglobal.loc[current_idx_label] = pd.Series({s: (InitialProb[s] / p_aux_sum
if s in quadruplet else 0) for s in scenarios})
    label.loc[current_idx_label] = s_label

SaveOptFixed = SaveOptFixed.reindex(idx_solutions)
SaveOptInv = SaveOptInv.reindex(idx_solutions)
SaveOptVariable = SaveOptVariable.reindex(idx_solutions)
SaveOptTotal = SaveOptTotal.reindex(idx_solutions)
SaveSimVariable = SaveSimVariable.reindex(idx_solutions)
SaveSimTotal = SaveSimTotal.reindex(idx_solutions)
Conflict = Conflict.reindex(idx_solutions)
ConflictPU = ConflictPU.reindex(idx_solutions)
conflicts = conflicts.reindex(idx_solutions)
lines1 = pd.DataFrame(np.nan, index=idx_solutions, columns=nodes)
lines2 = pd.DataFrame(np.nan, index=idx_solutions, columns=nodes)
InScenarios = pd.DataFrame(0.0, index=idx_solutions, columns=scenarios)

first_segment=idx_solutions[1:6]
second_segment=idx_solutions[6:16]
third_segment=idx_solutions[16:26]
fourth_segment=idx_solutions[26:31]

for ind in idx_solutions:
    for ss in scenarios:
        if probsglobal.loc[ind,ss] != 0:
            InScenarios.loc[ind,ss] = 1
```

```
solver = pye.SolverFactory('cbc')

# Model
def build_model(prob):
    m = pye.ConcreteModel()

    # Sets
    m.n = pye.Set(initialize=nodes)
    m.s = pye.Set(initialize=scenarios)
    m.t = pye.Set(initialize=types)
    m.idx = pye.Set(initialize=idx_solutions)
    m.cand = pye.Set(initialize=cand)
    m.fl = pye.Set(initialize=flowset)

    # Parameters
    m.D = pye.Param(m.n, m.s, initialize=D, default=0)
    m.R = pye.Param(m.n, m.s, initialize=R, default=0)
    m.B_cap = pye.Param(m.n, m.s, initialize=B_cap)
    m.InvestCost = pye.Param(m.t, initialize=InvestCost)
    m.Cap = pye.Param(m.t, initialize=Cap)
    m.Penalty = pye.Param(initialize=Penalty)
    m.Price_B = pye.Param(initialize=Price_B)

    # Variables
    m.flow = pye.Var(m.n, m.n, m.s, domain=pye.NonNegativeReals)
    m.g_B = pye.Var(m.n, m.s, domain=pye.NonNegativeReals)
    m.nse = pye.Var(m.n, m.s, domain=pye.NonNegativeReals)
    m.invCost = pye.Var(domain=pye.NonNegativeReals)
    m.opCost = pye.Var(domain=pye.NonNegativeReals)
    m.invest = pye.Var(m.cand, domain=pye.Binary)
    m.spill = pye.Var(m.n, m.s, domain=pye.NonNegativeReals)


    # Objective Function
    def obj_rule(model):
        return sum(model.invest[(n1, n2, tp)] * model.InvestCost[tp] for n1, n2,
tp in model.cand) + sum(model.g_B[n, sc] * model.Price_B * prob[sc] for n in
model.n for sc in model.s) + sum(model.nse[n, sc] * model.Penalty * prob[sc] for
n in model.n for sc in model.s)

    m.obj = pye.Objective(rule=obj_rule(m), sense=pye.minimize)

    # Power Balance Constraint
    def power_balance(model, node, sc):
       return (model.R[node, sc] + model.g_B[node, sc] + sum(model.flow[m_, node,
sc] for m_ in model.n) - sum(model.flow[node, m_, sc] for m_ in model.n)) ==
model.D[node, sc] - model.nse[node, sc] + model.spill[node, sc]
    m.powerBalance = pye.Constraint(m.n, m.s, rule=power_balance)

    # Flow Limit Constraint
    def flow_limit(model, n1, n2, sc):
        if n1 != n2:
            return model.flow[n1, n2, sc] <= sum(model.invest[(n1, n2, tp)] *
model.Cap[tp] for tp in model.t)
        else:
```

```
            return pye.Constraint.Skip

    m.flowLimit = pye.Constraint(m.n, m.n, m.s, rule=flow_limit)

    # Backup Generation Limit
    def backup_limit(model, node, sc):
        return model.g_B[node, sc] <= model.B_cap[node, sc]
    m.backupLimit = pye.Constraint(m.n, m.s, rule=backup_limit)

    # Non-served Energy Limit
    def nse_limit(model, node, sc):
        return model.nse[node, sc] <= model.D[node, sc]
    m.NonServedLimit = pye.Constraint(m.n, m.s, rule=nse_limit)

    # Spillage Limit
    def spill_limit(model, node, sc):
        return model.spill[node, sc] <= model.R[node, sc]
    m.SpillLimit = pye.Constraint(m.n, m.s, rule=spill_limit)

    # Investment Cost Definition
    def inv_cost_def(model):
            return   model.invCost   ==   sum(model.invest[(n1,   n2,   tp)]   *
model.InvestCost[tp] for n1, n2, tp in cand)
    m.invCostDef = pye.Constraint(rule=inv_cost_def)

    # Operation Cost Definition
    def op_cost_def(model):
        return model.opCost == (sum(model.g_B[n, sc] * model.Price_B * prob[sc]
for n in model.n for sc in model.s) + sum(model.nse[n, sc] * model.Penalty *
prob[sc] for n in model.n for sc in model.s))
    m.opCostDef = pye.Constraint(rule=op_cost_def)

    return m

# Optimization function
def run_optimization(Prob,current_idx):
    global probsglobal, SaveOptFixed, SaveOptInv, SaveOptVariable, SaveOptTotal
    global SaveSimVariable, SaveSimTotal, SaveSimExpectedVal
    global Conflict, ConflictPU, conflicts, SaveFlux

    print(f"\n--- Running optimization for solution {current_idx} ---")

    model = build_model(Prob)

    results = solver.solve(model)

            if     (results.solver.status     ==     pye.SolverStatus.ok)     and
(results.solver.termination_condition == pye.TerminationCondition.optimal):
        print(f"Optimization successful for {current_idx}")
        print('Status:', results.solver.status)
        print('Termination Condition:', results.solver.termination_condition)

        # Save optimization results
        SaveOptFixed.loc[current_idx] = pye.value(model.invCost)
        for n, m, t in model.cand:
```

```
                            SaveOptInv.loc[current_idx,    [(n,    m,    t)]]    =
pye.value(model.invest[(n,m,t)])
        for n, m, s in model.fl:
            SaveFlux.loc[current_idx, [(n, m, s)]] = pye.value(model.flow[n,m,s])
        SaveOptVariable.loc[current_idx] = pye.value(model.opCost)
                                           SaveOptTotal.loc[current_idx]          =
pye.value(next(model.component_objects(pye.Objective, active=True)))


        for i in range(len(first_segment)):
            probability=probsglobal.loc[f'idx{i+2:02}']
            model2=build_model(probability)

            for n, m, t in model2.cand:
                model2.invest[(n,m,t)].fix(pye.value(model.invest[(n,m,t)]))
            results = solver.solve(model2)
            for n, m, t in model2.cand:
                model2.invest[(n,m,t)].unfix()
           SaveSimVariable.loc[current_idx, f's{i+1}'] = pye.value(model2.opCost)
                                 SaveSimTotal.loc[current_idx,    f's{i+1}']    =
pye.value(next(model2.component_objects(pye.Objective, active=True)))


        SaveSimExpectedVal.loc[current_idx] = sum(probsglobal.loc['idx01', ss] *
SaveSimTotal.loc[current_idx, ss] for ss in scenarios)
    else:
                        print(f"Optimization    for    {current_idx}    failed:
{results.solver.termination_condition}")


#if __name__ == 'my_codeIII':
for idx in idx_solutions:
    probab=probsglobal.loc[idx]
    run_optimization(probab,idx)

for idxf in idx_solutions:
    for i in range(len(first_segment)):
        if SaveOptTotal.loc[f'idx{i+2:02}'] != 0:
            Conflict.loc[idxf, f's{i+1}'] = SaveSimTotal.loc[idxf, f's{i+1}'] -
SaveOptTotal.loc[f'idx{i+2:02}']
           ConflictPU.loc[idxf, f's{i+1}'] = SaveSimTotal.loc[idxf, f's{i+1}'] /
SaveOptTotal.loc[f'idx{i+2:02}'] - 1
        elif SaveSimTotal.loc[idxf, f's{i+1}'] != 0:
            Conflict.loc[idxf, f's{i+1}'] = SaveSimTotal.loc[idxf, f's{i+1}'] -
SaveOptTotal.loc[f'idx{i+2:02}']
            ConflictPU.loc[idxf, f's{i+1}'] = 100
        else:
            Conflict.loc[idxf, f's{i+1}'] = 0
            ConflictPU.loc[idxf, f's{i+1}'] = 0

        if ConflictPU.loc[idxf, f's{i+1}'] > Threshold:
            conflicts.loc[idxf, f's{i+1}'] = 1
        else:
            conflicts.loc[idxf, f's{i+1}'] = 0
```

```
for idx in idx_solutions:
    for nn in nodes:
        for nm in nodes:
            if nn != nm:
                if SaveOptInv.loc[idx, [(nn, nm, 't1')]].item() == 1:
                    lines1.loc[idx,nn]=1
                    lines1.loc[idx,nm]=1
                if SaveOptInv.loc[idx, [(nn, nm, 't2')]].item() == 1:
                    lines2.loc[idx,nn]=1
                    lines2.loc[idx,nm]=1


# Export results
SPO=pd.DataFrame(probsglobal)
SST=pd.DataFrame(SaveSimTotal)
SSEV=pd.DataFrame(SaveSimExpectedVal)
SCU=pd.DataFrame(ConflictPU)
SOF=pd.DataFrame(SaveOptFixed)
SF=pd.DataFrame(SaveFlux)
SOI = pd.DataFrame(SaveOptInv)
SOV=pd.DataFrame(SaveOptVariable)
SOT=pd.DataFrame(SaveOptTotal)
SSV=pd.DataFrame(SaveSimVariable)
LAB=pd.DataFrame(label)
CONFPU=pd.DataFrame(ConflictPU)
conflicts=pd.DataFrame(conflicts)

with
pd.ExcelWriter(r"C:\Users\modes\Documents\Universidad\Comillas\TFG\PYTHON\definitivo\model_results.xlsx", engine="openpyxl") as writer:
     label.to_excel(writer,  sheet_name="SaveProbsOpt",  startrow=0,  startcol=0, index=False, header=['Scenario solution'])
      SPO.to_excel(writer,  sheet_name="SaveProbsOpt",  startrow=0,  startcol=1, index=False)
     label.to_excel(writer,  sheet_name="SaveOptTotal",  startrow=0,  startcol=0, index=False, header=['Scenario solution'])
      SOT.to_excel(writer,  sheet_name="SaveOptTotal",  startrow=0,  startcol=1, index=False, header=['Value'])
         label.to_excel(writer,   sheet_name="SaveSimExpectedVal",   startrow=0, startcol=0, index=False, header=['Scenario solution'])
         SSEV.to_excel(writer,   sheet_name="SaveSimExpectedVal",   startrow=0, startcol=1, index=False, header=['Value'])
      label.to_excel(writer,  sheet_name="ConflictPu",  startrow=0,  startcol=0, index=False, header=['Scenario solution'])
      CONFPU.to_excel(writer,  sheet_name="ConflictPu",  startrow=0,  startcol=1, index=False)
       label.to_excel(writer,  sheet_name="Conflicts",  startrow=0,  startcol=0, index=False, header=['Scenario solution'])
     conflicts.to_excel(writer,  sheet_name="Conflicts",  startrow=0,  startcol=1, index=False)
     label.to_excel(writer,  sheet_name="lines_type_1",  startrow=0,  startcol=0, index=False, header=['Scenario solution'])
     lines1.to_excel(writer,  sheet_name="lines_type_1",  startrow=0,  startcol=1, index=False)
     label.to_excel(writer,  sheet_name="lines_type_2",  startrow=0,  startcol=0, index=False, header=['Scenario solution'])
```

```
    lines2.to_excel(writer, sheet_name="lines_type_2", startrow=0, startcol=1,
index=False)
      label.to_excel(writer,  sheet_name="Scenarios_in_solution",  startrow=0,
startcol=0, index=False, header=['Scenario solution'])
    InScenarios.to_excel(writer, sheet_name="Scenarios_in_solution", startrow=0,
startcol=1, index=False)
     label.to_excel(writer, sheet_name="SaveOptFixed", startrow=0, startcol=0,
index=False, header=['Scenario solution'])
      SOF.to_excel(writer,  sheet_name="SaveOptFixed",  startrow=0,  startcol=1,
index=False)
    label.to_excel(writer, sheet_name="SaveOptVariable", startrow=0, startcol=0,
index=False, header=['Scenario solution'])
     SOV.to_excel(writer, sheet_name="SaveOptVariable", startrow=0, startcol=1,
index=False)
      label.to_excel(writer,  sheet_name="SaveOptInv",  startrow=0,  startcol=0,
index=False, header=['Scenario solution'])
       SOI.to_excel(writer,  sheet_name="SaveOptInv",  startrow=0,  startcol=1,
index=False)
       label.to_excel(writer,  sheet_name="SaveFlux",  startrow=0,  startcol=0,
index=False, header=['Scenario solution'])
        SF.to_excel(writer,  sheet_name="SaveFlux",  startrow=0,  startcol=1,
index=False)
     label.to_excel(writer, sheet_name="SaveSimTotal", startrow=0, startcol=0,
index=False, header=['Scenario solution'])
      SST.to_excel(writer,  sheet_name="SaveSimTotal",  startrow=0,  startcol=1,
index=False)
    label.to_excel(writer, sheet_name="SaveSimVariable", startrow=0, startcol=0,
index=False, header=['Scenario solution'])
     SSV.to_excel(writer, sheet_name="SaveSimVariable", startrow=0, startcol=1,
index=False)


print("data succesfully saved in 'model_results.xlsx'")

subprocess.run(["streamlit",                                          "run",
r"C:\Users\modes\Documents\Universidad\Comillas\TFG\PYTHON\definitivo\dashboard.
py"])
```

Anexo 1.C: Código del algoritmo de “graficación”: “Dashboard.py”

```
# dashboard.py
import streamlit as st
import pandas as pd
import plotly.express as px
import numpy as np
import plotly.graph_objects as go
import plotly.figure_factory as ff
import math
from radial2 import plot_radar_scenarios
from sklearn.manifold import MDS, TSNE
from sklearn.metrics import pairwise_distances
```

```
from sklearn.decomposition import PCA
import pickle
from matplotlib.colors import to_rgba

nodes = ['n1', 'n2', 'n3']
scenarios = ['s1', 's2', 's3', 's4', 's5']

with open('data_trans.pkl', 'rb') as f:
    data_dict = pickle.load(f)

st.set_page_config(page_title="Electricity Net Expansion Dashboard",
layout="wide")

st.title("🌐 Electricity Net Optimization Dashboard")

@st.cache_data
def load_data():
    return {
        "Probabilities": pd.read_excel("model_results.xlsx",
sheet_name="SaveProbsOpt", index_col=0),
        "Total value for solution applied to scenario":
pd.read_excel("model_results.xlsx", sheet_name="SaveSimTotal", index_col=0),
        "Optimal cost for solution": pd.read_excel("model_results.xlsx",
sheet_name="SaveOptTotal", index_col=0),
        "Nodes connected by type 1 lines in solution":
pd.read_excel("model_results.xlsx", sheet_name="lines_type_1", index_col=0),
        "Nodes connected by type 2 lines in solution":
pd.read_excel("model_results.xlsx", sheet_name="lines_type_2", index_col=0),
        "Conflict degree between solution and scenarios in PU":
pd.read_excel("model_results.xlsx", sheet_name="ConflictPu", index_col=0),
        "Conflict between solution and scenarios (bool)":
pd.read_excel("model_results.xlsx", sheet_name="Conflicts"),
        "Expected value for solution applied to all scenarios":
pd.read_excel("model_results.xlsx", sheet_name="SaveSimExpectedVal"),
    }

data = load_data()

tab1, tab2, tab3, tab4, tab5, tab6, tab7, tab8, tab9 = st.tabs(["Display
scenarios", "Characterization of Solution", "Conflicts Graphics and data",
"Conflicts per Combinations", "Dendrograms", "🧮 Data", "📄 Estatistics",
"Multidimensional Embedding Visualization", "Conflict Solution Evolution"])

with tab1:
    parameters_to_plot = ['R', 'D', 'B_cap']

    df_scenarios_radar = pd.DataFrame(index=scenarios)
```

```
    for param_key in parameters_to_plot:
        if param_key not in data_dict:
            st.warning(f"Warning: Parameter '{param_key}' not found in the
'data' dictionary. It will be skipped.")
            continue

        df_pivot = data_dict[param_key].unstack(level='n')

        df_pivot.columns = [f"{param_key}_{col[1]}" for col in df_pivot.columns]

        df_scenarios_radar = df_scenarios_radar.merge(
            df_pivot, left_index=True, right_index=True, how='left'
        )

    if df_scenarios_radar.empty:
        st.error("Error: Could not restructure data for radar charts. Please
check the 'data' dictionary structure.")

    angles_deg_nodes = {'n1': 90, 'n2': 210, 'n3': 330}

    nodes_labels = ['Node 1', 'Node 2', 'Node 3']

    parameter_base_colors = ['blue', 'red', 'green', 'purple', 'orange']

    fill_alpha = 0.1
    parameter_display_names = {
        'D': 'Power Demand',
        'R': 'Renewable Generation',
        'B_cap': 'Back-up Thermal Generation Capacity'
    }

    num_scens = len(scenarios)
    cols_per_row = 1

    for i in range(0, num_scens, cols_per_row):
        cols = st.columns(cols_per_row)
        for j in range(cols_per_row):
            scen_idx = i + j
            if scen_idx < num_scens:
                with cols[j]:
                    scen_name = scenarios[scen_idx]

                    scen_row_data = df_scenarios_radar.loc[scen_name]

                    fig = go.Figure()

                    max_val_in_data = scen_row_data.max()
```

```
                    radial_axis_max = max(100, max_val_in_data * 1.1)

                    fig.update_layout(
                        polar=dict(
                            radialaxis=dict(
                                visible=True,
                                range=[0, radial_axis_max],
                                showticklabels=True,
                                tickangle=0,
                                tickfont=dict(size=9, color="gray"),
                                gridcolor='lightgray',
                                linecolor='gray',
                                linewidth=1
                            ),
                            angularaxis=dict(
                                visible=True,
                                linecolor='rgba(0,0,0,0)',
                                gridcolor='lightgray',
                                tickvals=[angles_deg_nodes[n] for n in nodes],
                                ticktext=nodes_labels,
                                showticklabels=True,
                                ticks="",
                                tickfont=dict(size=14, color="black"),
                                rotation=180,
                                direction="clockwise"
                            ),
                        ),
                        showlegend=True,
                        height=480,
                        width=480,
                        margin=dict(l=0, r=40, t=60, b=30),
                        title_text=f"Power data per node in {scen_name}
scenario",
                        legend=dict(
                            x=0.8,
                            y=0.87,

                            xanchor='right',
                            yanchor='bottom',

                            bgcolor='rgba(255, 255, 255, 0.7)',
                            bordercolor='LightGrey',
                            borderwidth=1,
                            font=dict(size=10)
                        )
                    )

                    for i_param, param in enumerate(parameters_to_plot):
```

```
                        r_values = []
                        theta_values = []

                        base_color_name = parameter_base_colors[i_param %
len(parameter_base_colors)]

                        rgba_tuple = to_rgba(base_color_name, alpha=fill_alpha)

                        fill_color_str = f"rgba({int(rgba_tuple[0]*255)},
{int(rgba_tuple[1]*255)}, {int(rgba_tuple[2]*255)}, {rgba_tuple[3]})"

                        for core in nodes:
                            col_name = f"{param}_{core}"
                            if col_name in scen_row_data.index and
pd.notna(scen_row_data[col_name]):
                                r_values.append(scen_row_data[col_name])
                                theta_values.append(angles_deg_nodes[core])
                            else:
                                r_values.append(0)
                                theta_values.append(angles_deg_nodes[core])

                        if r_values:
                            r_values.append(r_values[0])
                            theta_values.append(angles_deg_nodes[nodes[0]])

                            fig.add_trace(go.Scatterpolar(
                                r=r_values,
                                theta=theta_values,
                                mode='lines+markers',
                                name=parameter_display_names.get(param, param),

                                line=dict(color=base_color_name, width=2),
                                marker=dict(size=8),
                                fill='toself',

                                fillcolor=fill_color_str
                            ))
                            fig.data_dict[0].update(line=dict(width=3),
marker=dict(size=10))
                        else:
                            st.info(f"No complete data for parameter '{param}'
for scenario '{scen_name}'. No web will be drawn.")

                    st.plotly_chart(fig, use_container_width=True,
key=f"scen_radar_{scen_name}")

    st.write("---")
```

```
with tab2:


    st.title("Spider Connectivity Graphs for each solution")

    df_1 = pd.read_excel("model_results.xlsx", sheet_name="lines_type_1",
index_col=0)
    df_2 = pd.read_excel("model_results.xlsx", sheet_name="lines_type_2",
index_col=0)

    def plot_combined_spider_connectivity(df_row_type1, df_row_type2,
index_name):
        categories = ['n1', 'n2', 'n3']
        num_categories = len(categories)

        angles_deg = {
            'n1': 0,
            'n2': 120,
            'n3': 240
        }

        fig = go.Figure()

        fig.add_trace(go.Scatterpolar(
            r=[1.05] * num_categories,
            theta=[angles_deg[cat] for cat in categories],
            mode='lines',
            line=dict(color='rgba(250,0,0,0)'),
            fill='none',
            showlegend=False,
            hoverinfo='none',
            name='frame'
        ))

        for cat in categories:
            val_type1 = df_row_type1.get(cat)
            val_type2 = df_row_type2.get(cat)

            if (pd.notna(val_type1) and val_type1 == 1) or \
               (pd.notna(val_type2) and val_type2 == 1):
                fig.add_trace(go.Scatterpolar(
                    r=[1],
                    theta=[angles_deg[cat]],
                    mode='markers',
                    marker=dict(size=12, color='black', symbol='circle'),
                    showlegend=False,
                    hoverinfo='text',
                    text=f"{cat}"
```

```
                ))

        connections_names = [('n1', 'n2'), ('n2', 'n3'), ('n3', 'n1')]

        connected_r_type1 = []
        connected_theta_type1 = []

        for p1_name, p2_name in connections_names:
            val1_type1 = df_row_type1.get(p1_name)
            val2_type1 = df_row_type1.get(p2_name)

            if (pd.notna(val1_type1) and val1_type1 == 1) and \
               (pd.notna(val2_type1) and val2_type1 == 1):
                connected_r_type1.extend([1, 1, None])
                connected_theta_type1.extend([angles_deg[p1_name],
angles_deg[p2_name], None])

        # Ensure r and theta are not empty lists for legend visibility
        r1_display = connected_r_type1 if connected_r_type1 else [None]
        theta1_display = connected_theta_type1 if connected_theta_type1 else
[None]

        # Always add Type 1 lines trace, even if empty
        fig.add_trace(go.Scatterpolar(
            r=r1_display,
            theta=theta1_display,
            mode='lines',
            line=dict(color='blue', width=3),
            name='Type 1 lines',
            hoverinfo='text',
            text=["Type 1 Connection"] * (len(connected_r_type1) // 3) if
connected_r_type1 else [],
            showlegend=True # Explicitly ensure legend is shown for this trace
        ))

        connected_r_type2 = []
        connected_theta_type2 = []

        for p1_name, p2_name in connections_names:
            val1_type2 = df_row_type2.get(p1_name)
            val2_type2 = df_row_type2.get(p2_name)

            if (pd.notna(val1_type2) and val1_type2 == 1) and \
               (pd.notna(val2_type2) and val2_type2 == 1):
                connected_r_type2.extend([1, 1, None])
                connected_theta_type2.extend([angles_deg[p1_name],
angles_deg[p2_name], None])
```

```
        # Ensure r and theta are not empty lists for legend visibility
        r2_display = connected_r_type2 if connected_r_type2 else [None]
        theta2_display = connected_theta_type2 if connected_theta_type2 else
[None]

        # Always add Type 2 lines trace, even if empty, to ensure legend
visibility
        fig.add_trace(go.Scatterpolar(
            r=r2_display,
            theta=theta2_display,
            mode='lines',
            line=dict(color='red', width=3),
            name='Type 2 lines',
            hoverinfo='text',
            text=["Type 2 Connection"] * (len(connected_r_type2) // 3) if
connected_r_type2 else [],
            showlegend=True # Explicitly ensure legend is shown for this trace
        ))

        fig.update_layout(
            title_text=f"Node Connectivity for {index_name} scenario
combination",
            polar=dict(
                radialaxis=dict(
                    visible=False,
                    range=[0, 1.25]
                ),
                angularaxis=dict(
                    visible=True,
                    linecolor='rgba(0,0,0,0)',
                    gridcolor='rgba(0,0,0,0)',
                    tickvals=[angles_deg[cat] for cat in categories],
                    ticktext=['Node 1', 'Node 2', 'Node 3'],
                    tickfont=dict(size=20, color="darkgray"),
                    showticklabels=True,
                    ticks="",
                    rotation=90,
                    direction="clockwise"
                ),
            ),
            showlegend=True,
            legend=dict(
                x=0.9, y=1.1,
                xanchor="right", yanchor="top",
                bgcolor="rgba(255,255,255,0.7)",
                bordercolor="Black",
                borderwidth=1
            ),
```

```
            height=500,
            width=600,
            margin=dict(l=40, r=40, t=90, b=40)
        )

        return fig

    st.subheader("This panel displays the connections between nodes resulting
from the network optimization under multiple scenario combinations.\n")

    common_indices = df_1.index.intersection(df_2.index)

    for idx in common_indices:
        st.write(f"### Combined Solution for {idx} scenario combination")
        fig = plot_combined_spider_connectivity(df_1.loc[idx], df_2.loc[idx],
idx)
        st.plotly_chart(fig, use_container_width=True, key=f"combined_{idx}")




with tab3:
    st.subheader("📈 Heatmap of Conflicts (ConflictPU)")

    colorscale = [
        [0.0, "white"],
        [0.25, "white"],
        [0.35, "yellow"],
        [0.5, "orange"],
        [0.65, "red"],
        [1.0, "darkred"],
    ]

    fig = px.imshow(data["Conflict degree between solution and scenarios in
PU"], aspect="auto", zmin=-1, zmax=4, text_auto=True,
color_continuous_scale=colorscale)
    n_rows, n_cols = data["Conflict degree between solution and scenarios in
PU"].shape

    fig.update_layout(coloraxis_colorbar=dict(
        title="Valor",
        tickvals=[-1, 0.2, 1, 2, 20, 100],
        ticktext=["-1", "0.2", "1", "2", "20", ">20"]),
        width = max(600, 120 * n_cols),
        height = max(400, 30 * n_rows),
        margin=dict(l=50, r=50, t=50, b=50, pad=0))
```

```
    st.plotly_chart(fig, use_container_width=False)


    df1 = pd.read_excel("model_results.xlsx", sheet_name="Conflicts")
    df2 = df1.drop('Sum', errors='ignore')
    df1['Sum'] = df1.drop(columns='Scenario solution').sum(axis=1)

    fig = px.bar(
        df1,
        x='Sum',
        y='Scenario solution',
        orientation='h',
        labels={'Scenario solution': 'Scenario combination per solution', 'Sum':
'Number of conflicts'},
        title='Conflicts per solution',
    )

    fig.update_layout(
        height=800,
        plot_bgcolor='white',
        yaxis=dict(autorange='reversed'),
        bargap=0.2
    )

    st.title("Number of conflicts per solution")
    st.plotly_chart(fig, use_container_width=True)


    sumable_columns = df2.select_dtypes(include='number').columns.drop('Scenario
solution', errors='ignore')

    suma_columnas = df2[sumable_columns].sum()


    df_suma = suma_columnas.reset_index()

    df_suma.columns = ['Scenario', 'Sum']

    fig1 = px.bar(
        df_suma,
        x='Scenario',
        y='Sum',
        title='Conflicts sum per scenario',
        labels={'Sum': 'Number of conflicts', 'Scenario': 'Scenario'},
        text='Sum'
    )

    fig1.update_layout(plot_bgcolor='white', height=600, width=800, bargap=0.1)
```

```
    st.title("Total sum per Scenario")
    st.plotly_chart(fig1, use_container_width=False)

with tab4:

    sumable_columns = df2.select_dtypes(include='number').columns.drop('Solution
scenario', errors='ignore')

    ranges = {
        "Solution for individual scenarios": df2.iloc[1:6],
        "Solution for pairs of scenarios": df2.iloc[6:16],
        "Solution for triplets of scenarios": df2.iloc[16:26],
        "Solution for quadruplet of scenarios": df2.iloc[26:31],
        "Stochastic solution": df2.iloc[[0]]
    }

    st.title("Conflicts sum per scenario for individuals, pairs, triplets,
etc.")

    for name, subset in ranges.items():
        sum = subset[sumable_columns].sum()
        df_sum = sum.reset_index()
        df_sum.columns = ['Scenario', 'Sum']

        fig2 = px.bar(
            df_sum,
            x='Scenario',
            y='Sum',
            title=name,
            labels={'Scenario': 'Scenario', 'Sum': 'Number of conflicts'},
            text='Sum'
        )
        fig2.update_layout(plot_bgcolor='white', yaxis=dict(range=[0, 10]),
height=400, width=600, bargap=0.1)

        st.plotly_chart(fig2, use_container_width=False)

with tab5:


    df3 = pd.read_excel("model_results.xlsx", sheet_name="ConflictPu")

    df3.set_index('Scenario solution', inplace=True)

    data_vectors1 = df3.values
    labels1 = df3.index.tolist()
```

```
    fig5 = ff.create_dendrogram(data_vectors1, orientation='left',
labels=labels1)
    fig5.update_layout(width=800, height=30 * len(labels1), margin=dict(l=20,
r=20, t=40, b=20), title="Solutions Dendrogram in PU")

    st.title("Solutions Dendrogram")
    st.plotly_chart(fig5, use_container_width=True)

    data_vectors = []
    labels = []
    sumable_columns = sumable_columns[0:5]

    for name, subset in ranges.items():
        suma = subset[sumable_columns].sum().values
        data_vectors.append(suma)
        labels.append(name)

    data_array = np.array(data_vectors)

    fig3 = ff.create_dendrogram(data_array, orientation='left', labels=labels)
    fig3.update_layout(width=800, height=500, margin=dict(l=20, r=20, t=40,
b=20), title="Combinations Dendrogram in number of conflicts")

    st.title("Cobinations Dendrogram")
    st.plotly_chart(fig3, use_container_width=True)

    data_ar = np.transpose(data_vectors)
    data_array2 = np.array(data_ar)

    fig4 = ff.create_dendrogram(data_array2, orientation='left',
labels=scenarios)
    fig4.update_layout(width=800, height=500, margin=dict(l=20, r=20, t=40,
b=20), title="Scenarios Dendrogram in number of conflicts")

    st.title("Scenarios Dendrogram")
    st.plotly_chart(fig4, use_container_width=True)

with tab6:
    show_sheet = st.selectbox("Select sheet", list(data.keys()))
    st.dataframe(data[show_sheet], use_container_width=False)

with tab7:
    st.subheader("🔢 Expected value for solution applied to all scenarios")
    Expected_vals = pd.read_excel("model_results.xlsx",
sheet_name="SaveSimExpectedVal", index_col=0).iloc[:, 0]
    Total_vals = pd.read_excel("model_results.xlsx", sheet_name="SaveOptTotal",
index_col=0).iloc[:, 0]
```

```
    combined = pd.merge(Expected_vals, Total_vals, on='Scenario solution',
how='inner')
    combined = combined.reset_index()
    combined = combined.rename(columns={'Value_x': 'Expected Value', 'Value_y':
'Total Value'})
    df_long = combined.melt(id_vars=['Scenario solution'],
                            value_vars=['Expected Value', 'Total Value'],
                            var_name='Solution type',
                            value_name='Value')
    fig = px.line(df_long,
                x="Scenario solution",
                y="Value",
                color="Solution type",
                title="Optimal and Expected Value",
                labels={"Scenario solution": "Scenario combination", "Value":
"Value", "Solution": "Value type"})

    st.plotly_chart(fig, use_container_width=True)

with tab8:

    st.title("t-SNE, MDS and PCA Visualization of Solutions of Scenarios and
Combinations")

    df2 = pd.read_excel("model_results.xlsx", sheet_name="ConflictPu")

    df2 = df2.drop('Sum', errors='ignore')

    st.subheader("Visualization Settings")

    if df2.columns[0] != "Scenario solution":
        df2.rename(columns={df2.columns[0]: "Scenario solution"}, inplace=True)

    ranges = {
        "individuals": df2.iloc[1:6],
        "pairs": df2.iloc[6:16],
        "triplets": df2.iloc[16:26],
        "quadruplets": df2.iloc[26:31],
        "stochastic": df2.iloc[[0]],
    }

    groups_order = ["individuals", "pairs", "triplets", "quadruplets",
"stochastic"]

    embed_method = st.selectbox("Select Dimensionality Reduction method",
options=["t-SNE", "MDS", "PCA"], index=0)
    dim_option = st.selectbox("Select embedding dimension", ["2D", "3D"],
index=0)
```

```
#    show_individuals = st.checkbox("Show individual scenarios", value=True)
    selected_groups = st.multiselect("Select groups to show",
options=["individuals", "pairs", "triplets", "quadruplets", "stochastic"],
default=["individuals", "pairs", "triplets", "quadruplets", "stochastic"])

    dims = 2 if dim_option == "2D" else 3

    compute_individuals = True
    display_individuals = "individuals" in selected_groups # and
show_individuals

    data_to_embed = []
    labels = []
    group_labels = []

    if compute_individuals:
        data_to_embed.append(ranges["individuals"].iloc[:, 1:].values)
        labels.extend(ranges["individuals"]["Scenario
solution"].astype(str).tolist())
        group_labels.extend(["individuals"] * len(ranges["individuals"]))

    for g in groups_order:
        if g == "individuals":
            continue
        if g in selected_groups:
            data_to_embed.append(ranges[g].iloc[:, 1:].values)
            labels.extend(ranges[g]["Scenario solution"].astype(str).tolist())
            group_labels.extend([g] * len(ranges[g]))

    all_data = np.vstack(data_to_embed)
    distance_matrix = pairwise_distances(all_data, metric='euclidean')
    characteristic_matrix = np.array(all_data)

    if embed_method == "MDS":
        embedder = MDS(n_components=dims, dissimilarity='precomputed',
random_state=42)
        embedding = embedder.fit_transform(distance_matrix)
    elif embed_method == "t-SNE":
        perplexity_val = min(30, len(distance_matrix) - 1)
        embedder = TSNE(n_components=dims, metric="precomputed",
random_state=42, perplexity=perplexity_val, init="random")
        embedding = embedder.fit_transform(distance_matrix)
    elif embed_method == "PCA":
        embedder = PCA(n_components=dims)
        embedding = embedder.fit_transform(characteristic_matrix)

    fig = go.Figure()
    unique_groups = sorted(set(group_labels))
```

```
    colors_map = {
        "individuals": "blue",
        "pairs": "green",
        "triplets": "orange",
        "quadruplets": "purple",
        "stochastic": "red"
    }

    for g in unique_groups:
        idxs = [i for i, grp in enumerate(group_labels) if grp == g]
        x = embedding[idxs, 0]
        y = embedding[idxs, 1]
        if dims == 3:
            z = embedding[idxs, 2]
        hover_texts = [f"Scenario: {labels[i]}<br>Group: {g}" for i in idxs]
        if g == "individuals" and not display_individuals:
            continue
        if dims == 2:
            fig.add_trace(go.Scatter(
                x=x, y=y,
                mode='markers+text',
                marker=dict(color=colors_map.get(g, "gray"), size=8),
                text=[labels[i] for i in idxs],
                textposition="top center",
                name=g.capitalize(),
                hovertext=hover_texts,
                hoverinfo="text"
            ))
        else:
            fig.add_trace(go.Scatter3d(
                x=x, y=y, z=z,
                mode='markers+text',
                marker=dict(color=colors_map.get(g, "gray"), size=6),
                text=[labels[i] for i in idxs],
                textposition="top center",
                name=g.capitalize(),
                hovertext=hover_texts,
                hoverinfo="text"
            ))

    fig.update_layout(
        title="Embedding Visualization",
        legend_title="Groups",
        margin=dict(l=0, r=0, b=0, t=40),
        height=700
    )

    st.plotly_chart(fig, use_container_width=True)
```

```
with tab9:
    st.title("Conflict Solving Evolution")

    st.subheader("How diferent scenarios combinations solutions solve
conflicts")

    conflict_bool = pd.read_excel("model_results.xlsx", sheet_name="Conflicts",
index_col=0)

    num_zeros = (~conflict_bool).sum(axis=1)
    pattern_str = conflict_bool.apply(lambda row:
''.join(row.astype(int).astype(str)), axis=1)
    original_index = np.arange(len(conflict_bool))
    order_df = pd.DataFrame({
        'num_zeros': num_zeros,
        'pattern_str': pattern_str,
        'original_index': original_index
    }, index=conflict_bool.index)

    order_df = order_df.sort_values(by=['num_zeros', 'pattern_str',
'original_index'])

    first_solution = conflict_bool.index[0]

    order_df['original_index'] = np.arange(len(order_df))

    grouped = order_df.groupby(['num_zeros', 'pattern_str'])

    ordered_groups = []

    for group_keys, group in grouped:
        if first_solution in group.index:
            group = group.sort_values(by='original_index')
            group_without_first = group.drop(index=first_solution)
            first_row = group.loc[[first_solution]]
            group = pd.concat([group_without_first, first_row])
        else:
            group = group.sort_values(by='original_index')
        ordered_groups.append(group)

    order_df = pd.concat(ordered_groups)

    ordered_indices = order_df.index
    conflict_sorted = conflict_bool.loc[ordered_indices]

    n_solutions = conflict_sorted.shape[0]
    n_scenarios = conflict_sorted.shape[1]
```

```
    scenario_labels = conflict_sorted.columns.tolist()

    fig = go.Figure()

    for col_idx, solution in enumerate(conflict_sorted.index):
        col_data = conflict_sorted.loc[solution]
        y_true = np.where(col_data)[0]

        fig.add_trace(go.Scatter(
            x=[col_idx] * len(y_true),
            y=y_true,
            mode='markers',
            marker=dict(color='gray', size=4),
            showlegend=False
        ))

        true_positions = y_true
        if len(true_positions) > 0:
            blocks = []
            start = true_positions[0]
            prev = start
            for pos in true_positions[1:]:
                if pos == prev + 1:
                    prev = pos
                else:
                    blocks.append((start, prev))
                    start = pos
                    prev = pos
            blocks.append((start, prev))

            for start_block, end_block in blocks:
                y0 = start_block - 0.5
                y1 = end_block + 0.5
                fig.add_shape(
                    type="line",
                    x0=col_idx,
                    x1=col_idx,
                    y0=y0,
                    y1=y1,
                    line=dict(color="blue", width=2),
                    xref='x',
                    yref='y'
                )

    fig.update_layout(
        title='Conflict Resolution Diagram',
        yaxis=dict(
            title='Scenarios',
```

```
            tickmode='array',
            tickvals=list(range(n_scenarios)),
            ticktext=scenario_labels,
            tickfont=dict(size=16, color="darkgray"),
            autorange='reversed',
            showgrid=False,
            zeroline=False
        ),
        xaxis=dict(
            title='Solutions',
            tickmode='array',
            tickvals=list(range(n_solutions)),
            ticktext=list(conflict_sorted.index),
            tickfont=dict(size=16, color="darkgray"),
            tickangle=45,
            showgrid=False,
            zeroline=False
        ),
        plot_bgcolor='white',
        margin=dict(l=40, r=40, t=60, b=100),
        height=600
    )

    fig.update_yaxes(showline=False, showgrid=False)
    fig.update_xaxes(showline=True, showgrid=False)

    st.plotly_chart(fig, use_container_width=True)


st.markdown("---")
st.markdown("© 2025 - Dashboard desarrollado por MAC en Streamlit + Plotly |
Powered by Python")
```

Anexo 2: Gráficos resultantes del análisis con Dashboard.py

Anexo 2.A: Problema 1

Caracterización de los escenarios:

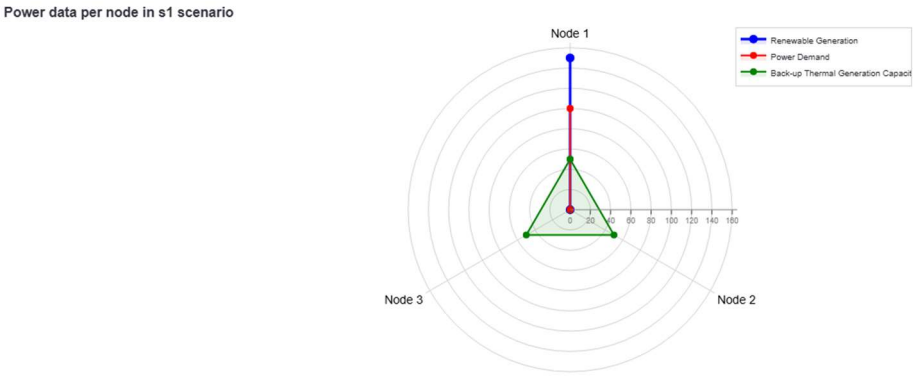


Figura 5

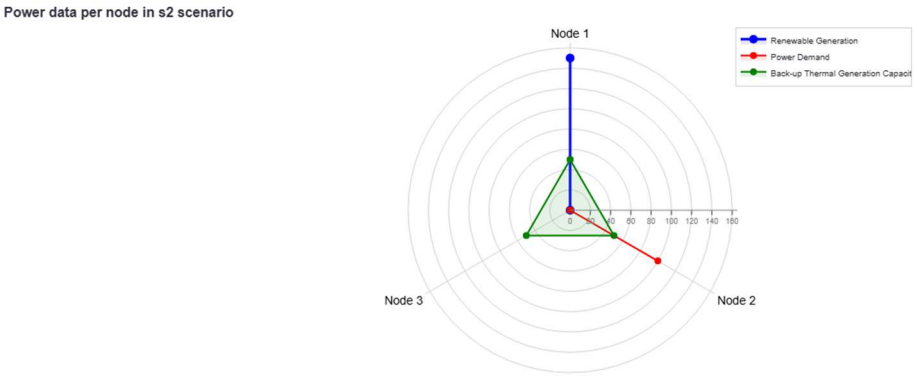


Figura 6

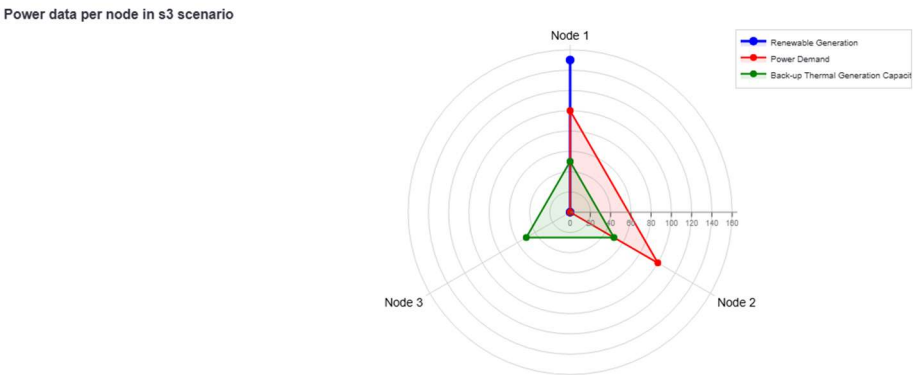


Figura 7

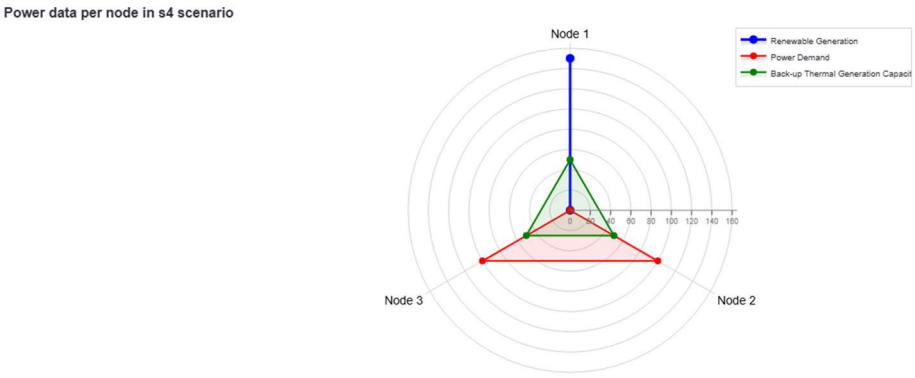


Figura 8

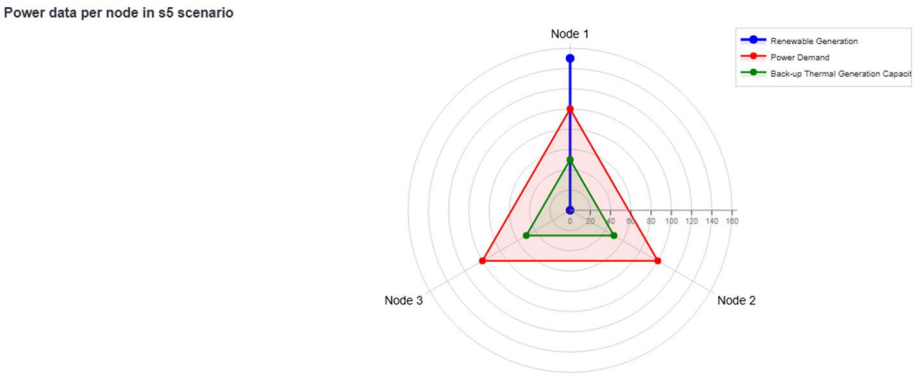


Figura 9

Caracterización de las soluciones:

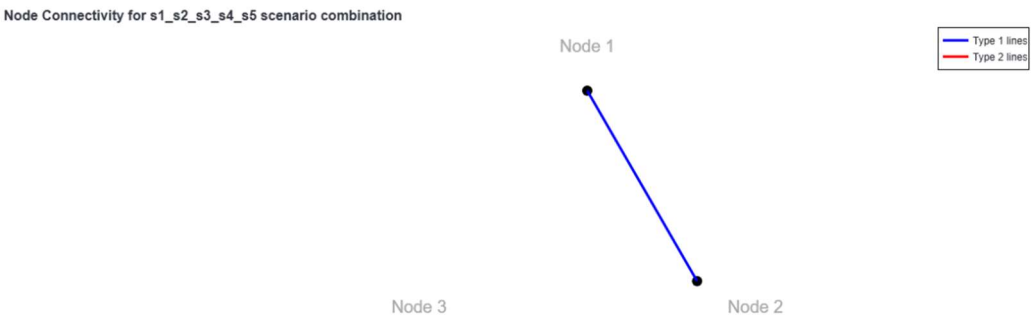


Figura 10

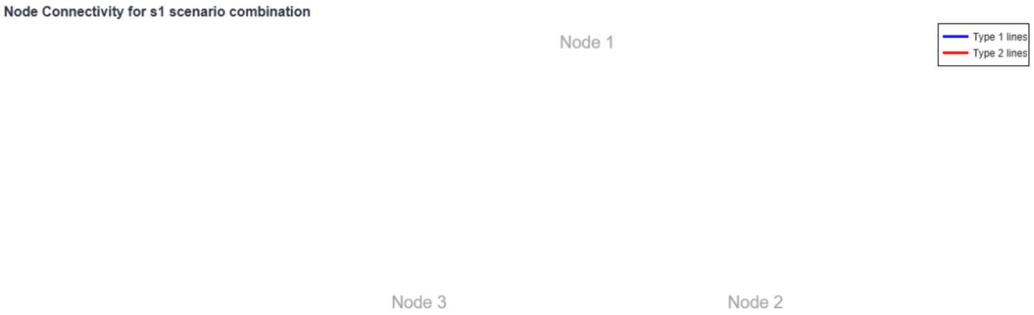


Figura 11

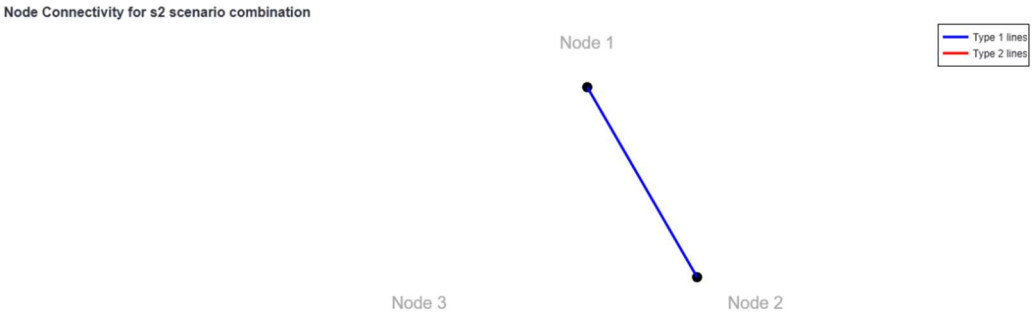


Figura 12

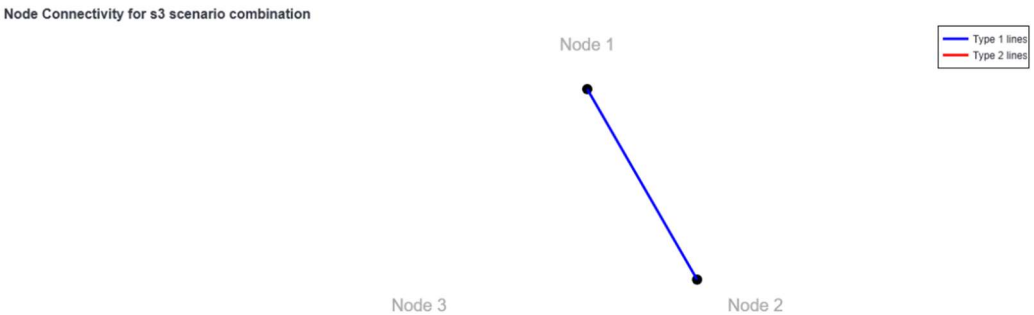


Figura 13

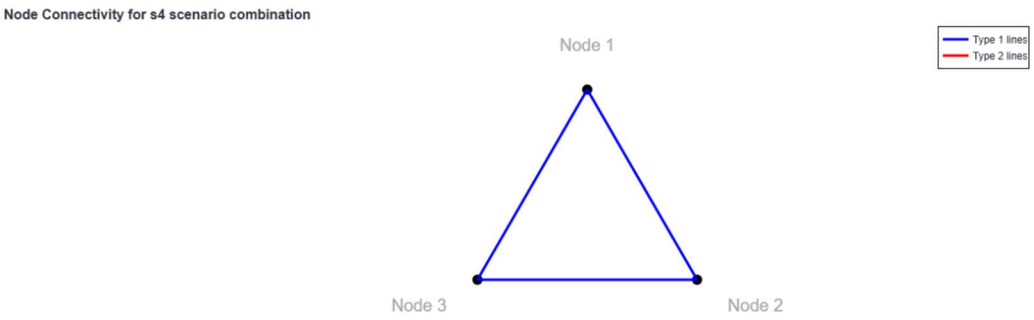


Figura 14

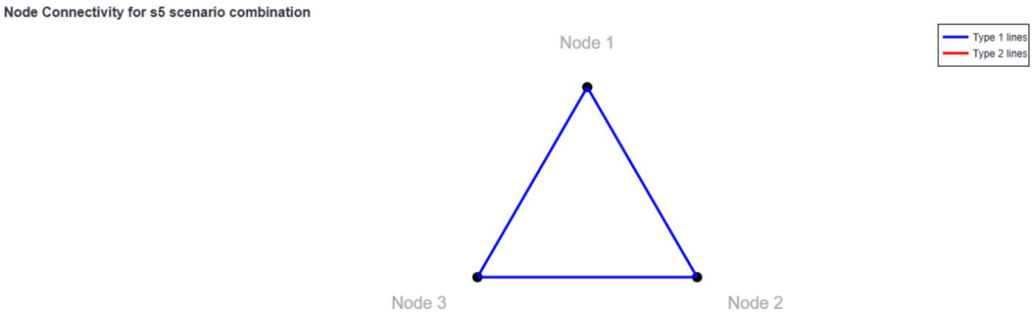


Figura 15

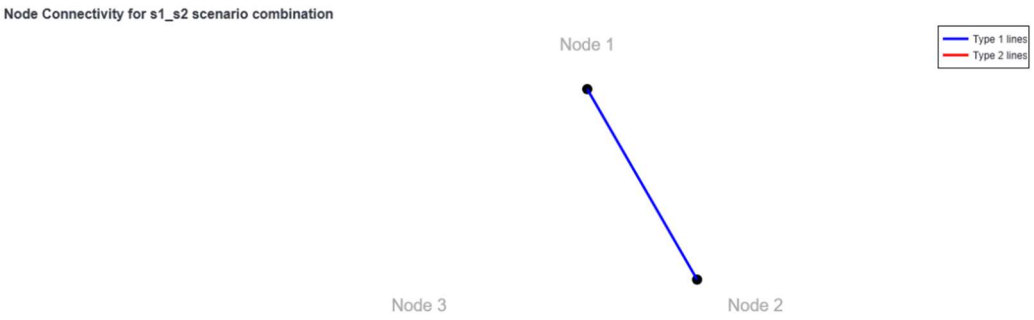


Figura 16

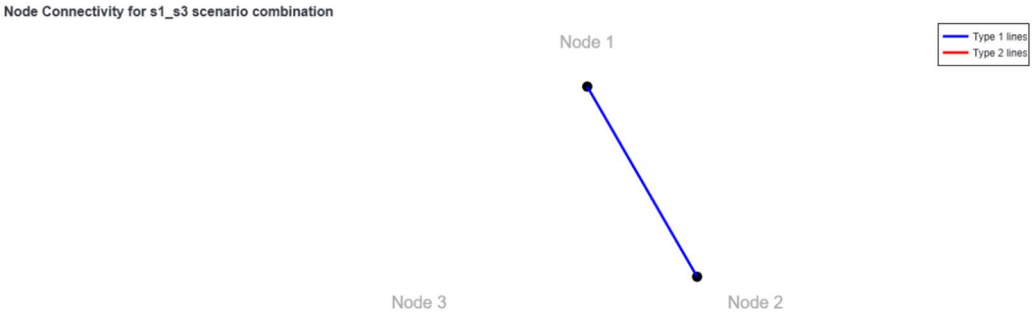


Figura 17

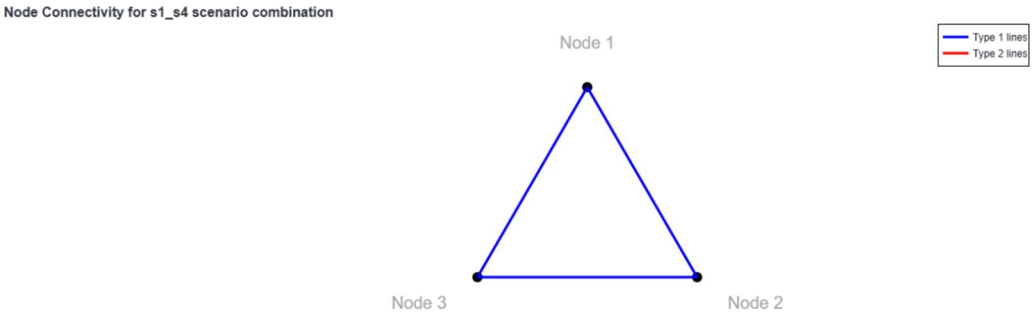


Figura 18

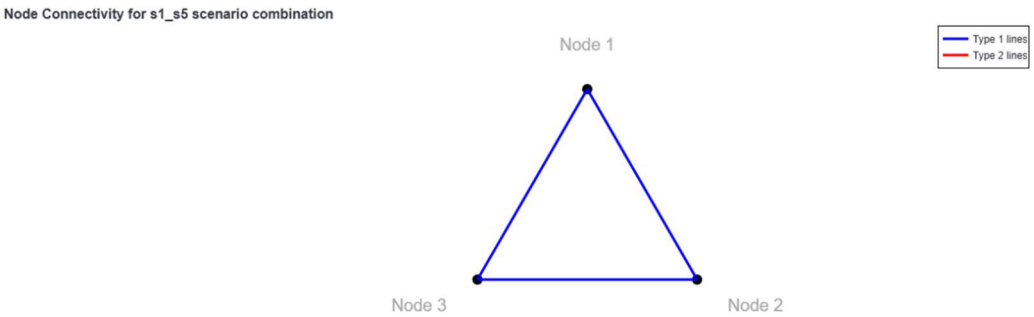


Figura 19

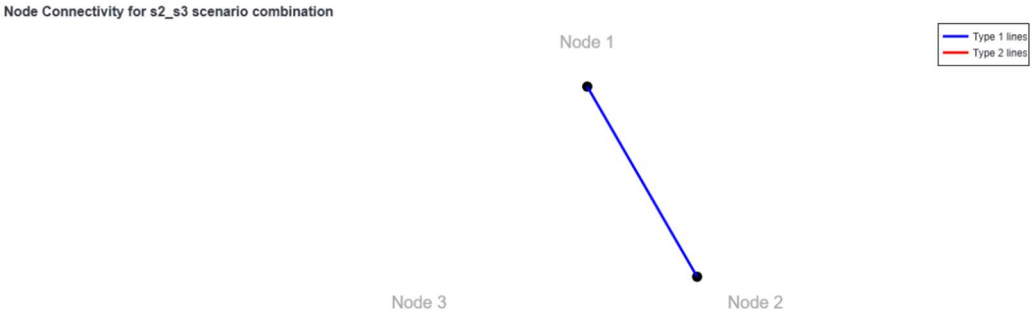


Figura 20

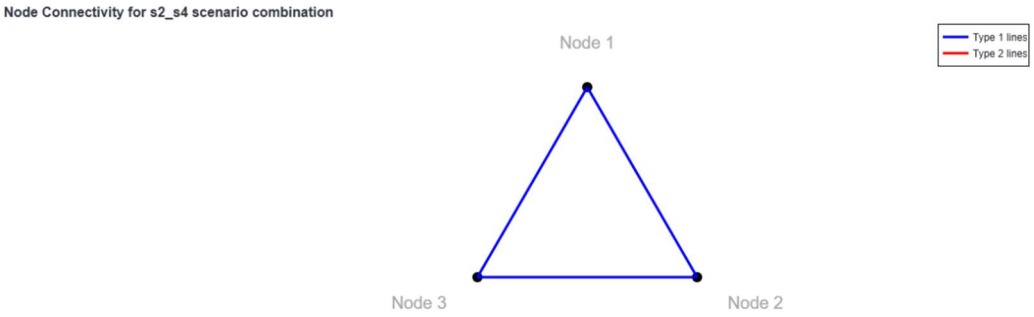


Figura 21

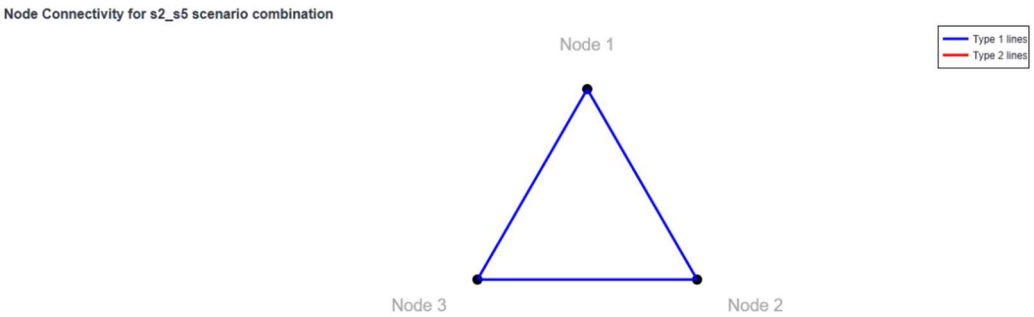


Figura 22

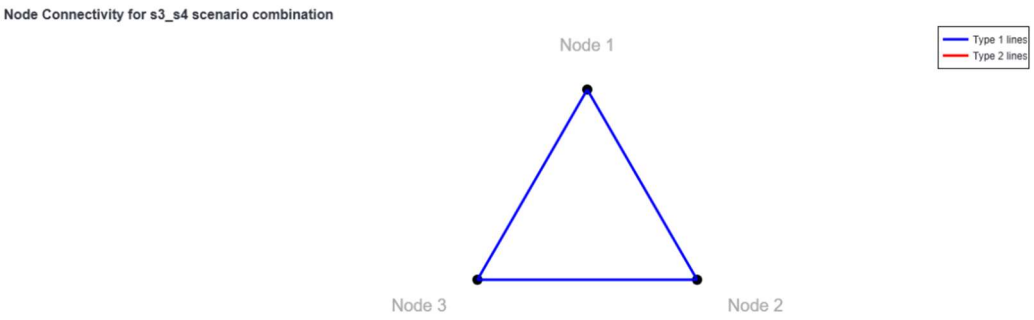


Figura 23

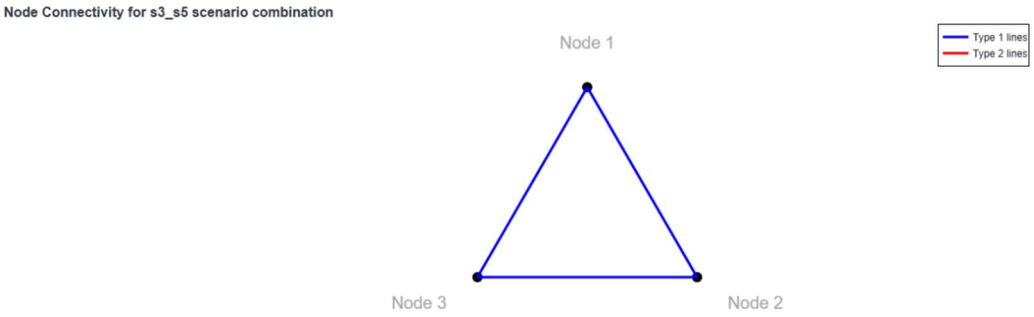


Figura 24

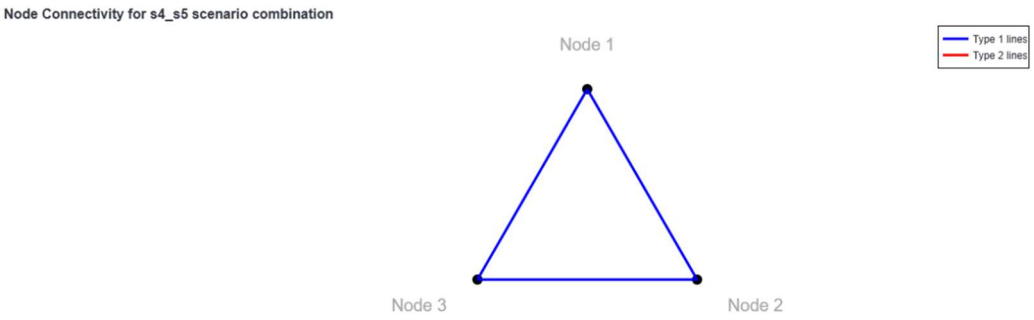


Figura 25

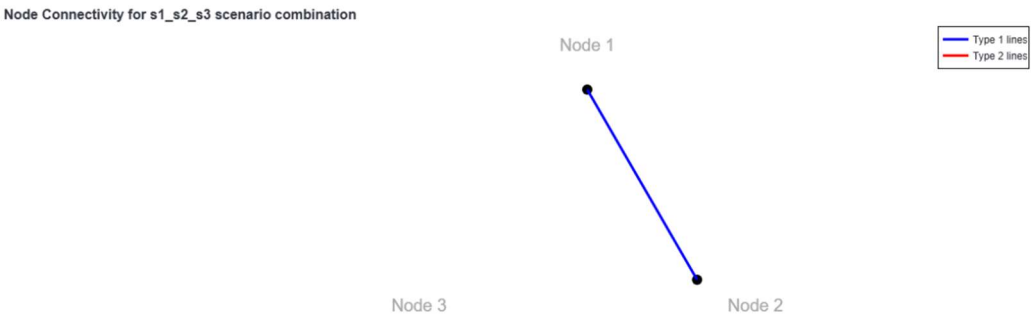


Figura 26

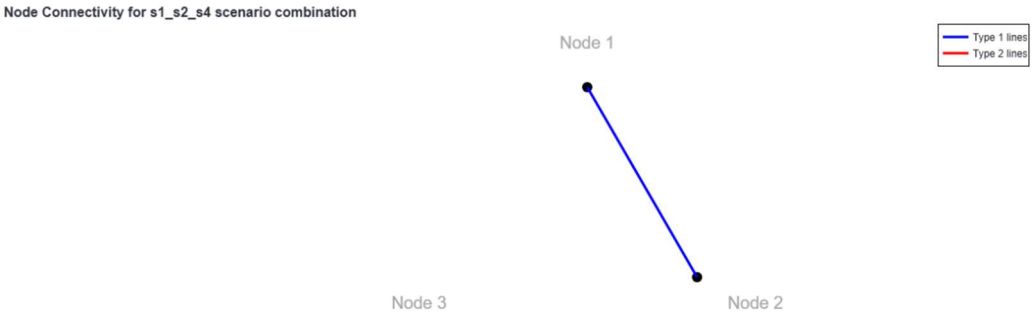


Figura 27

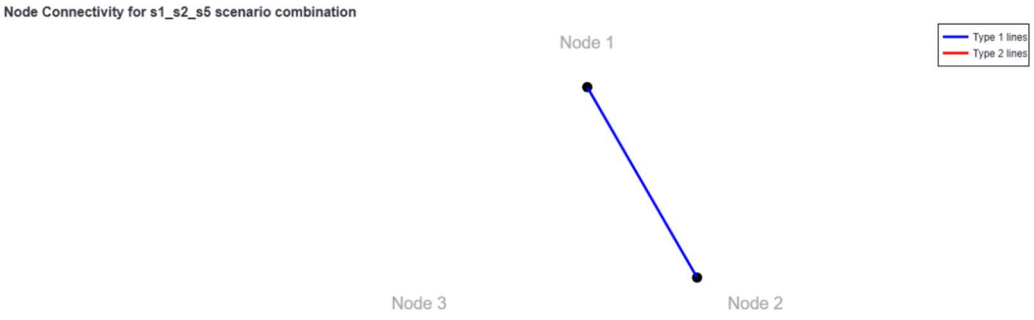


Figura 28

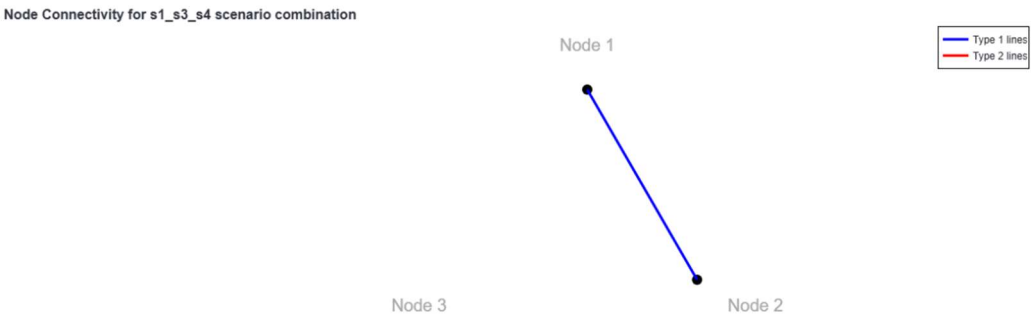


Figura 29

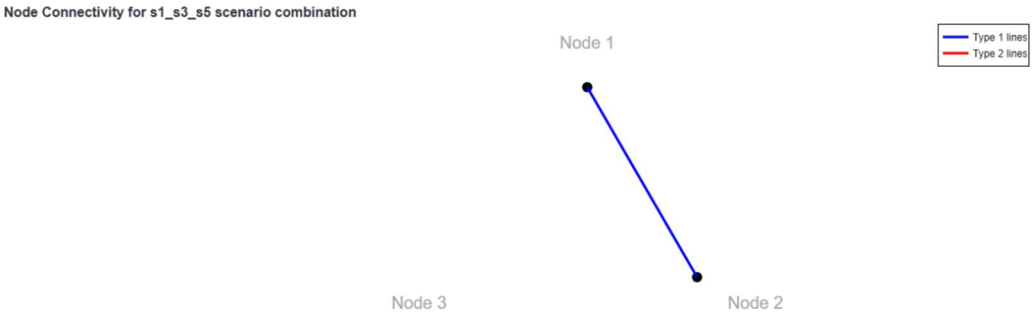


Figura 30

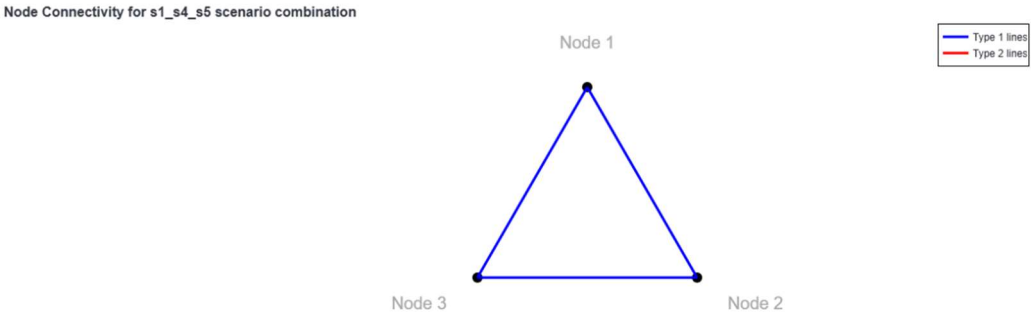


Figura 31

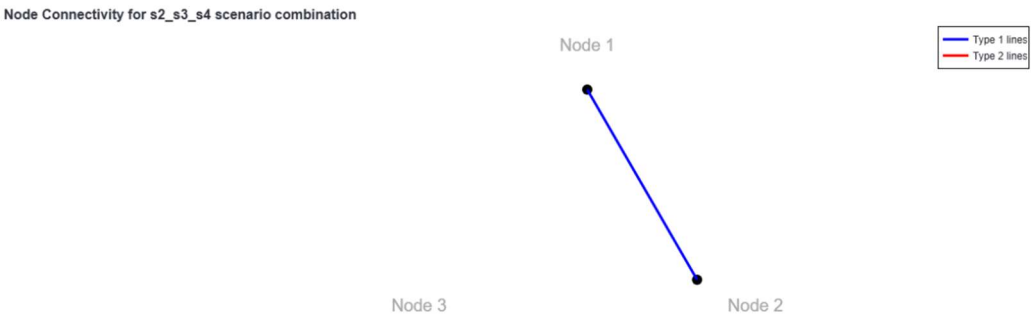


Figura 32

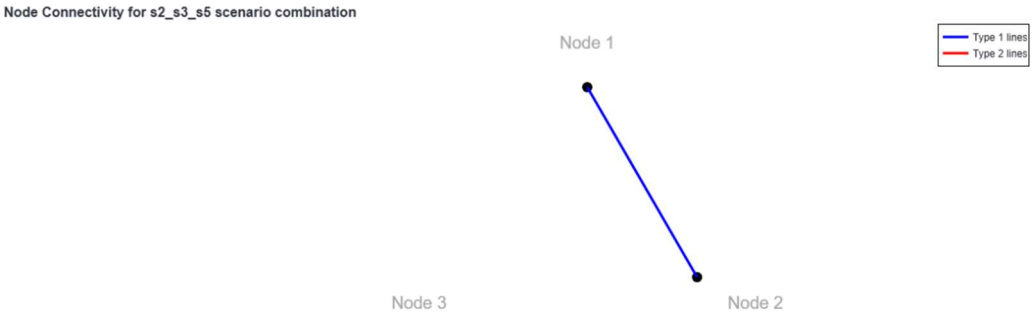


Figura 33

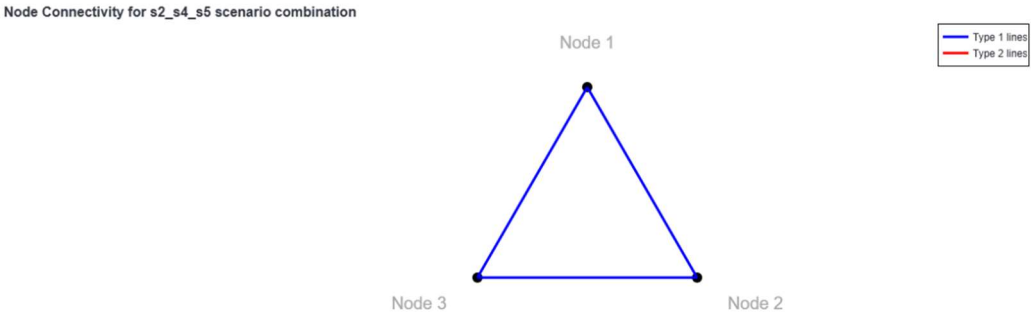


Figura 34

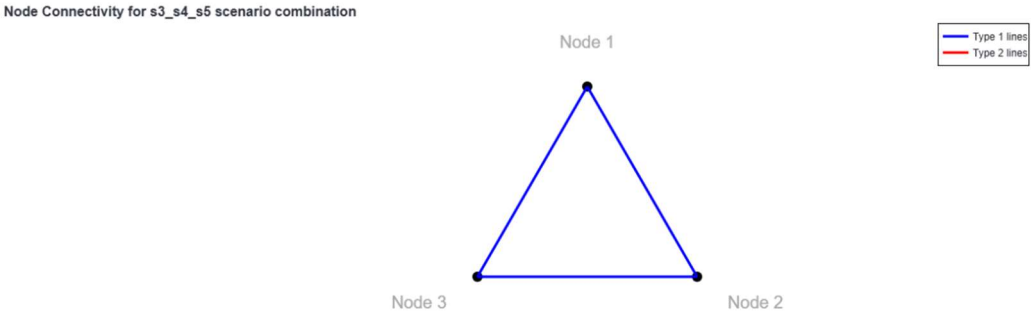


Figura 35

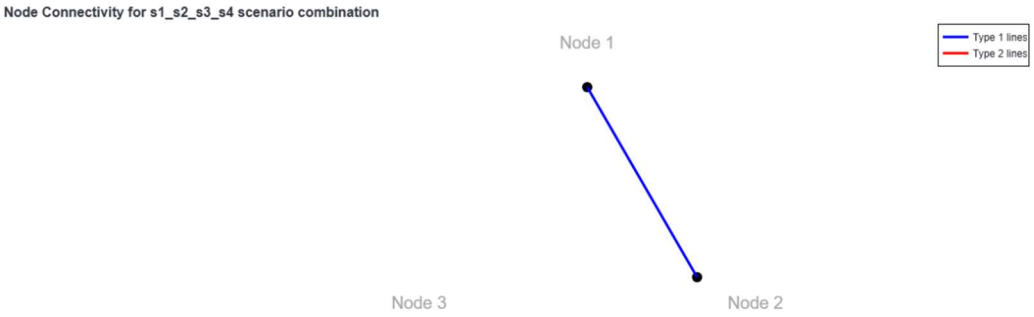


Figura 36

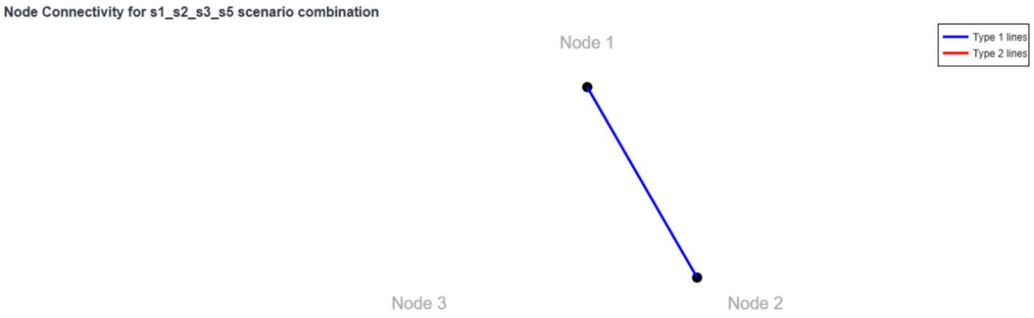


Figura 37

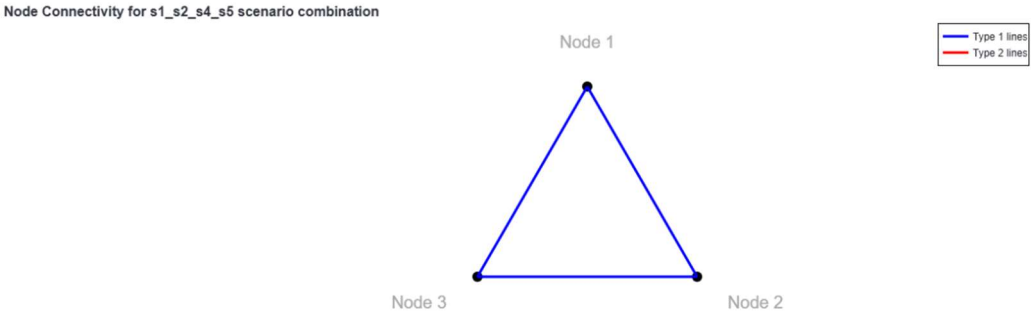


Figura 38

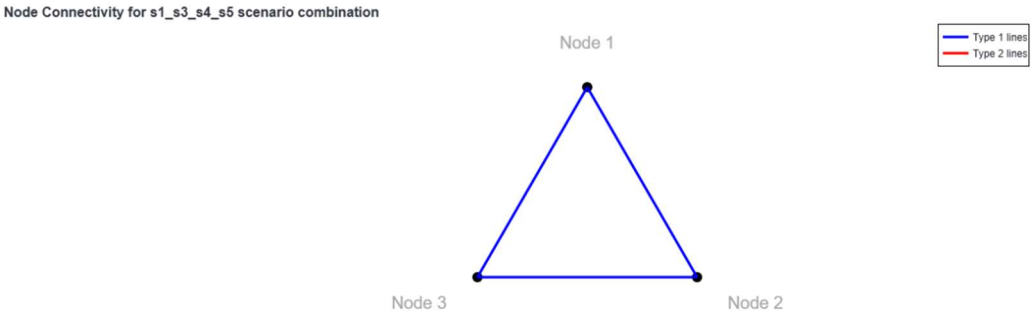


Figura 39

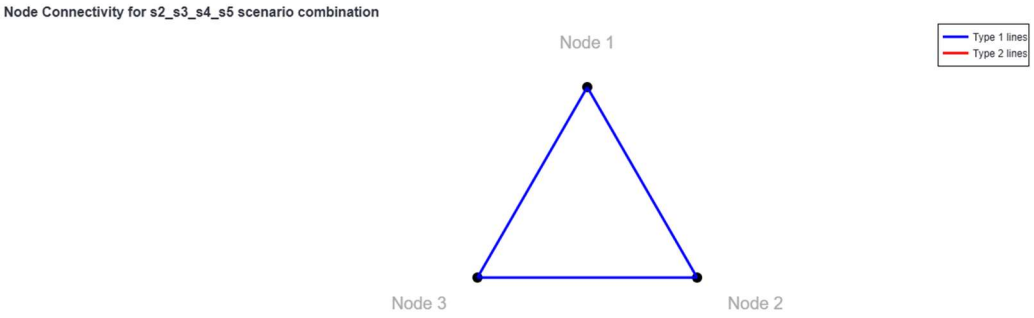


Figura 40

Mapa de calor:

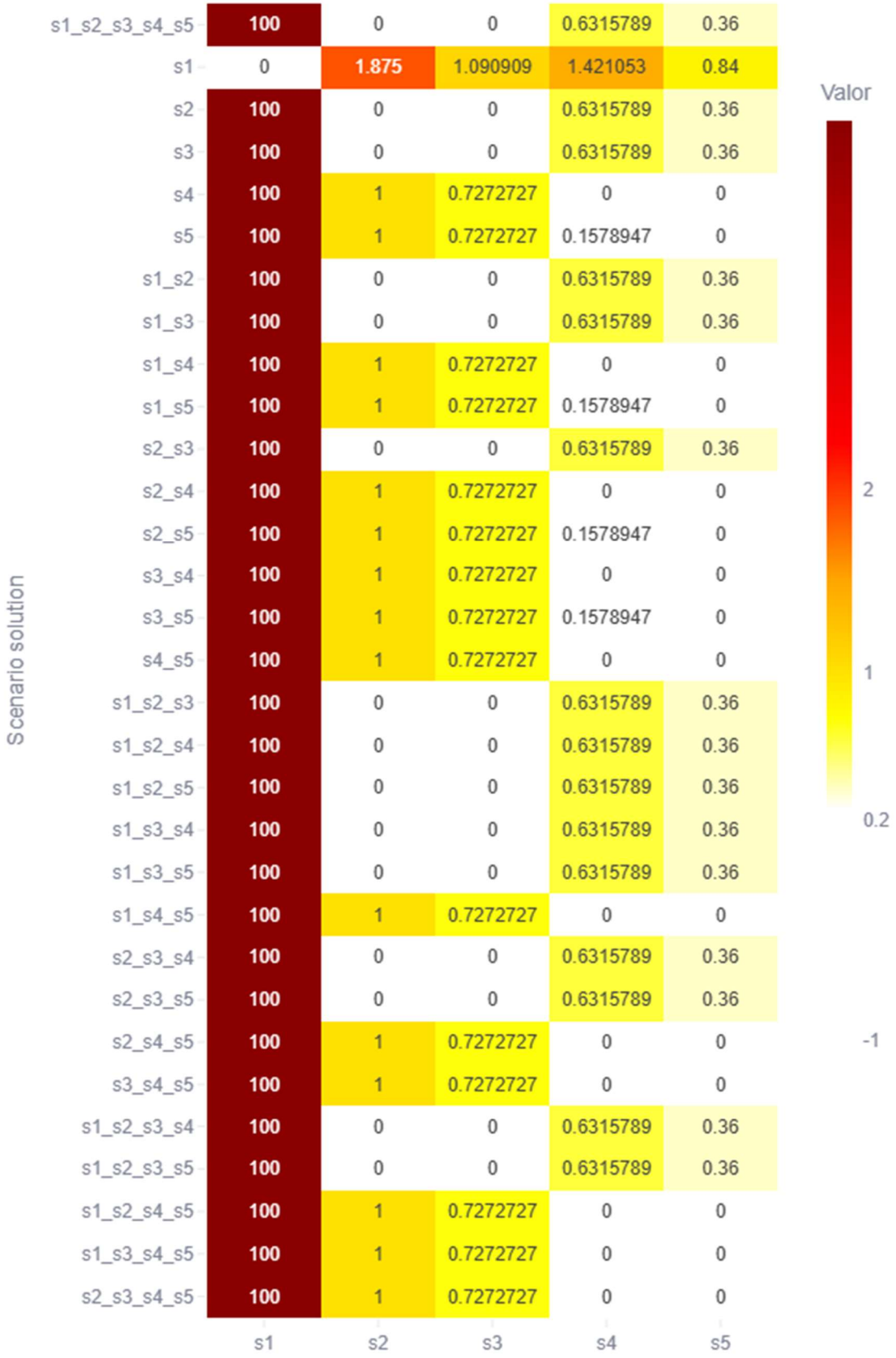

Scenario solution
s1_s2_s3_s4_s5 100 0 0 0.6315789 0.36
s1 0 1.875 1.090909 1.421053 0.84
s2 100 0 0 0.6315789 0.36
s3 100 0 0 0.6315789 0.36
s4 100 1 0.7272727 0 0
s5 100 1 0.7272727 0.1578947 0
s1_s2 100 0 0 0.6315789 0.36
s1_s3 100 0 0 0.6315789 0.36
s1_s4 100 1 0.7272727 0 0
s1_s5 100 1 0.7272727 0.1578947 0
s2_s3 100 0 0 0.6315789 0.36
s2_s4 100 1 0.7272727 0 0
s2_s5 100 1 0.7272727 0.1578947 0
s3_s4 100 1 0.7272727 0 0
s3_s5 100 1 0.7272727 0.1578947 0
s4_s5 100 1 0.7272727 0 0
s1_s2_s3 100 0 0 0.6315789 0.36
s1_s2_s4 100 0 0 0.6315789 0.36
s1_s2_s5 100 0 0 0.6315789 0.36
s1_s3_s4 100 0 0 0.6315789 0.36
s1_s3_s5 100 0 0 0.6315789 0.36
s1_s4_s5 100 1 0.7272727 0 0
s2_s3_s4 100 0 0 0.6315789 0.36
s2_s3_s5 100 0 0 0.6315789 0.36
s2_s4_s5 100 1 0.7272727 0 0
s3_s4_s5 100 1 0.7272727 0 0
s1_s2_s3_s4 100 0 0 0.6315789 0.36
s1_s2_s3_s5 100 0 0 0.6315789 0.36
s1_s2_s4_s5 100 1 0.7272727 0 0
s1_s3_s4_s5 100 1 0.7272727 0 0
s2_s3_s4_s5 100 1 0.7272727 0 0
s1 s2 s3 s4 s5
Valor
2
1
0.2
-1

Figura 41

Gráfico de barras de los conflictos entre combinaciones (por escenarios agrupados) y escenarios:

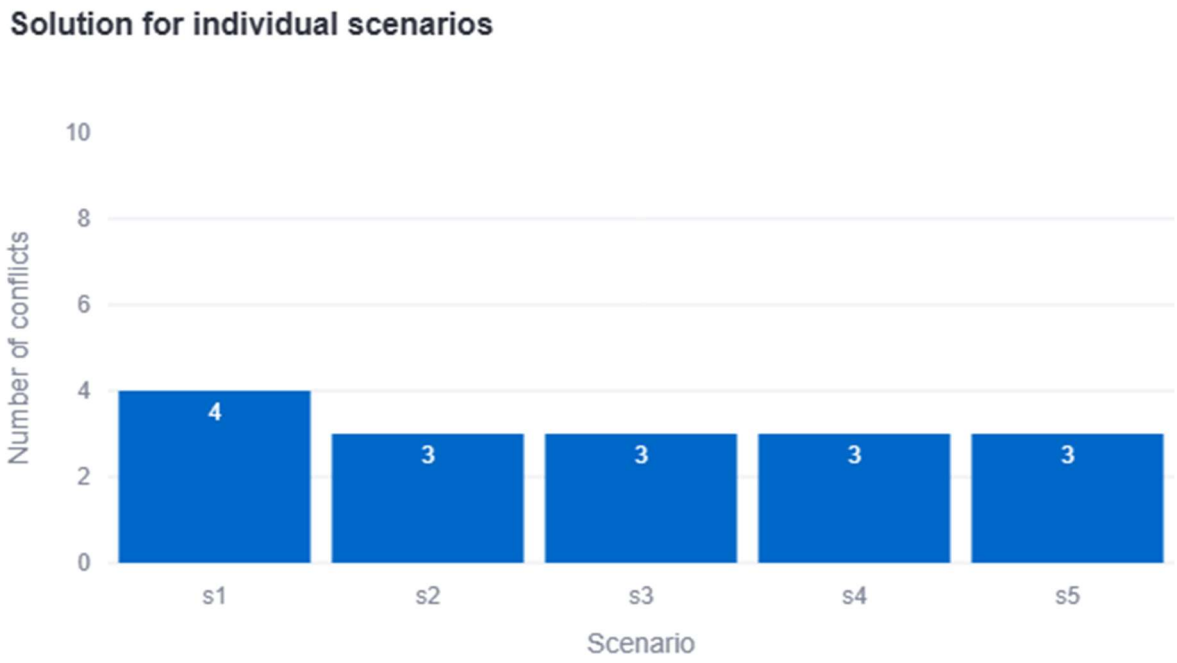


Figura 42

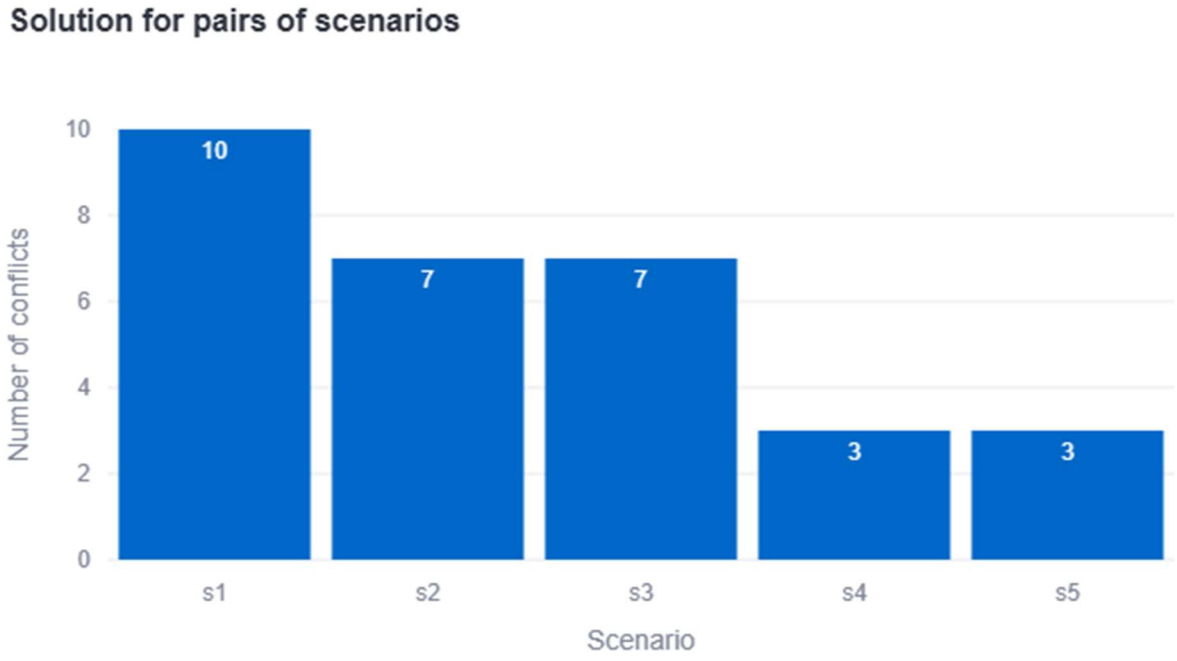


Figura 43

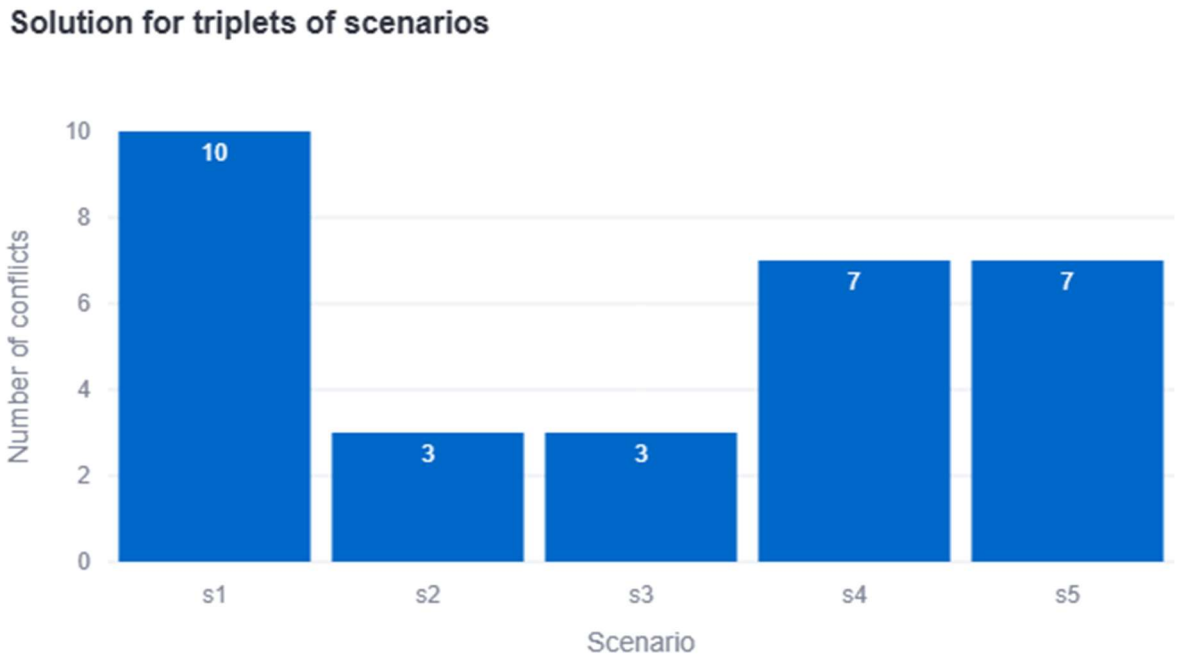


Figura 44

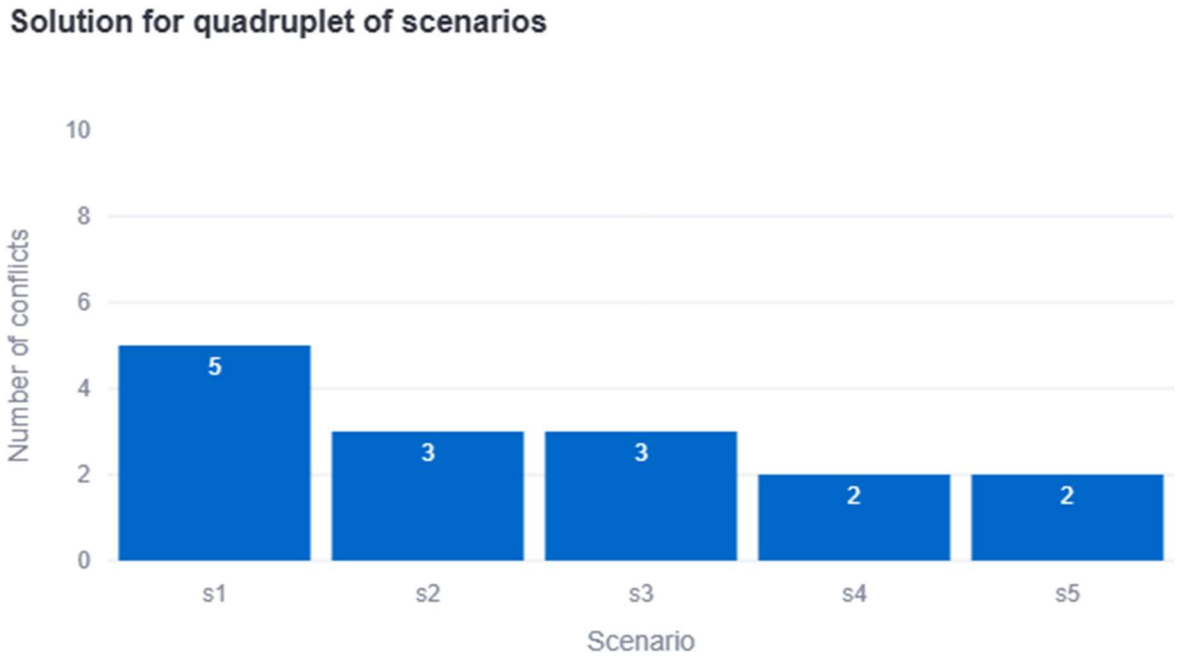


Figura 45

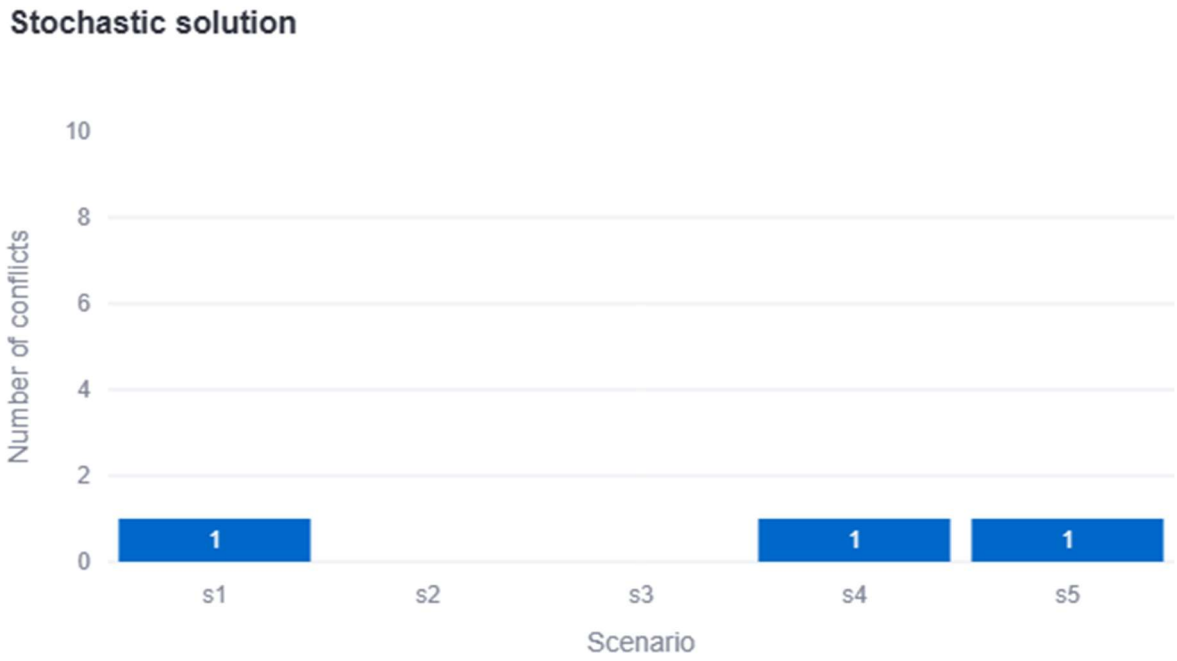


Figura 46

Resolución de conflictos:

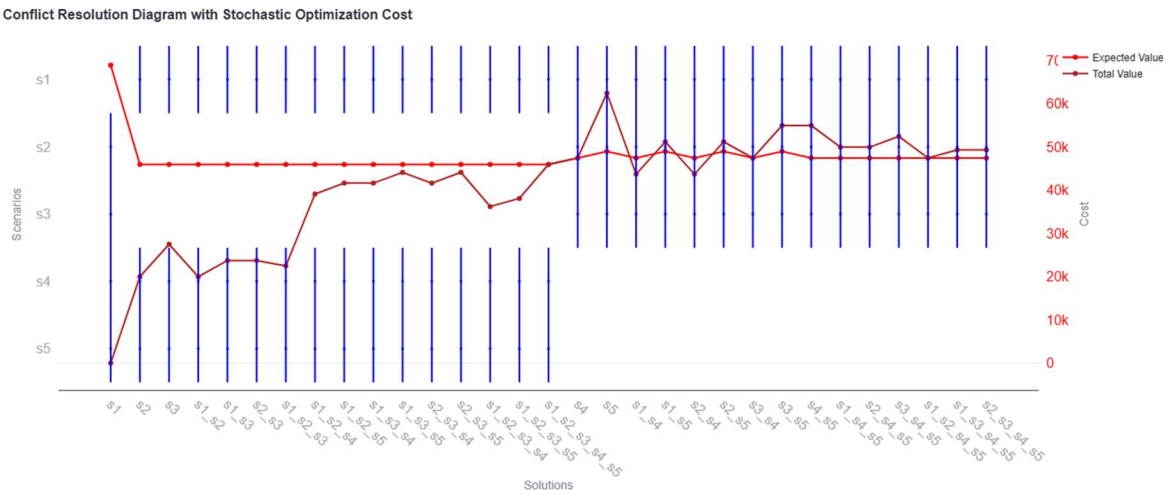


Figura 47

Anexo 2.B: Problema 2

Caracterización de los escenarios:

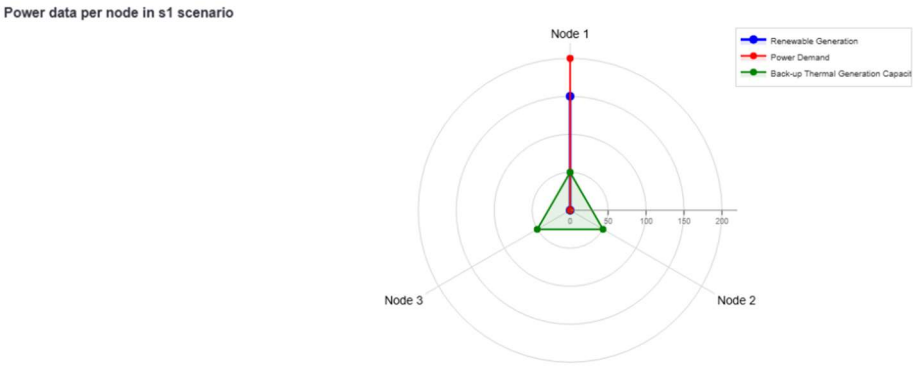


Figura 48

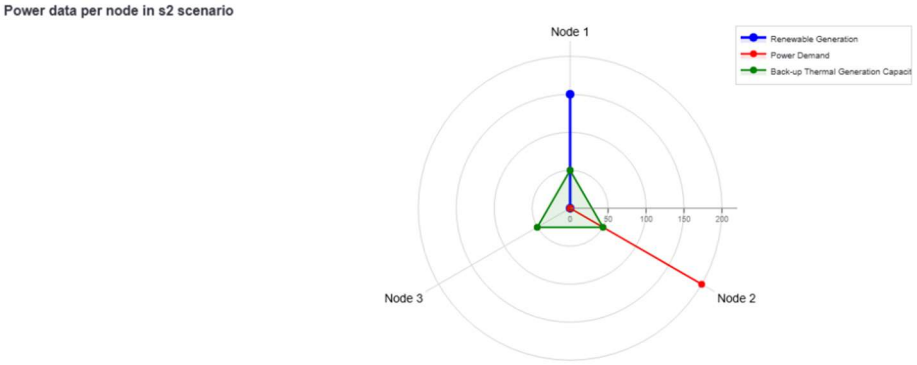


Figura 49

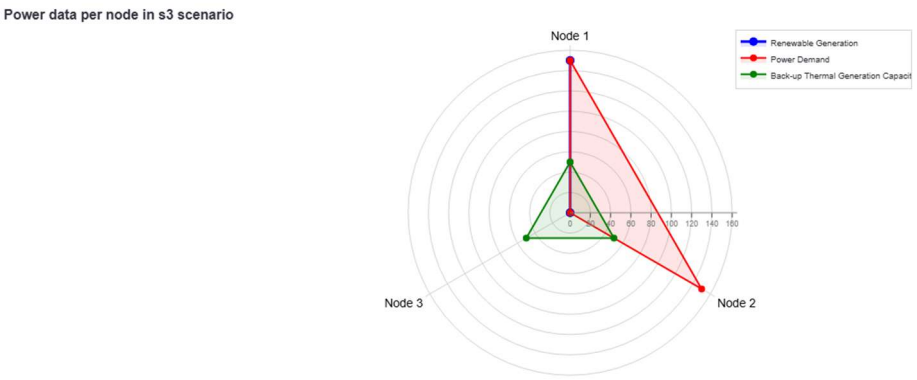


Figura 50

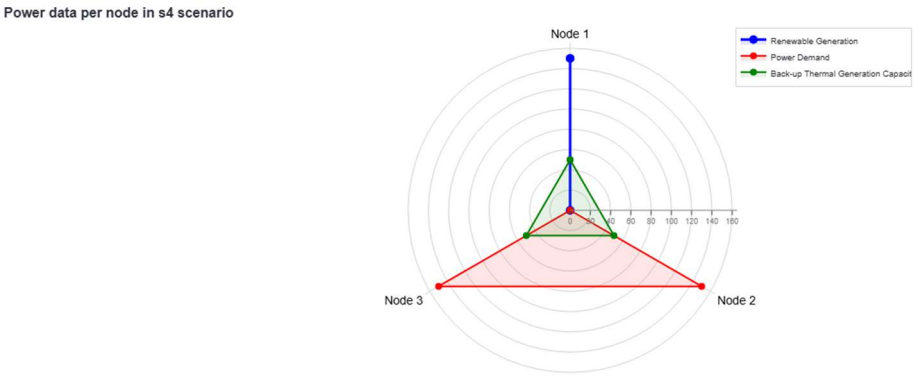


Figura 51

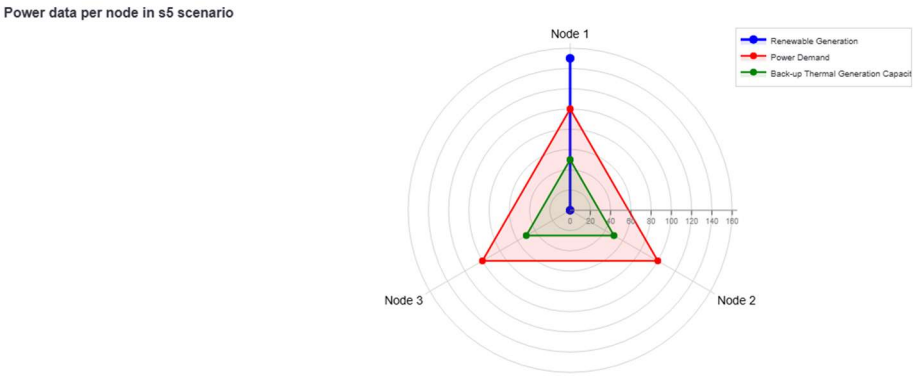


Figura 52

Caracterización de las soluciones:

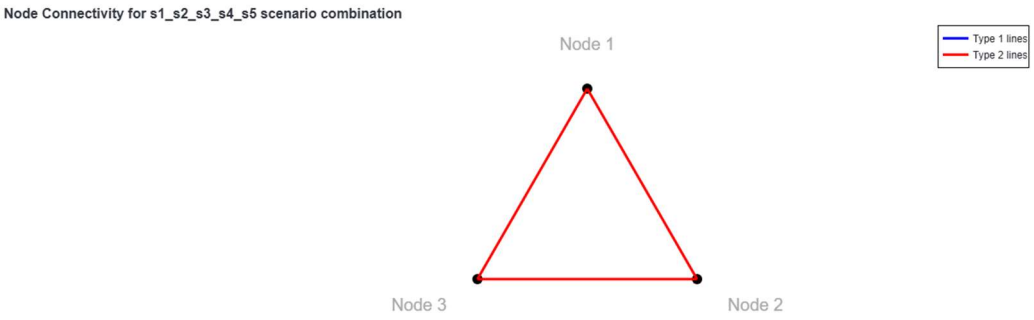


Figura 53

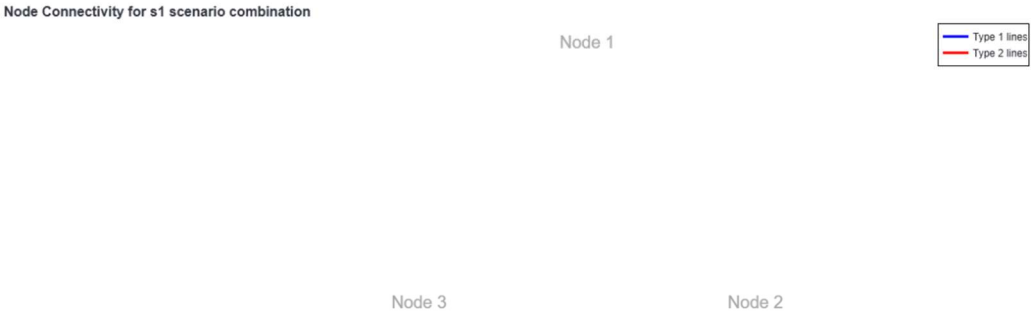


Figura 54

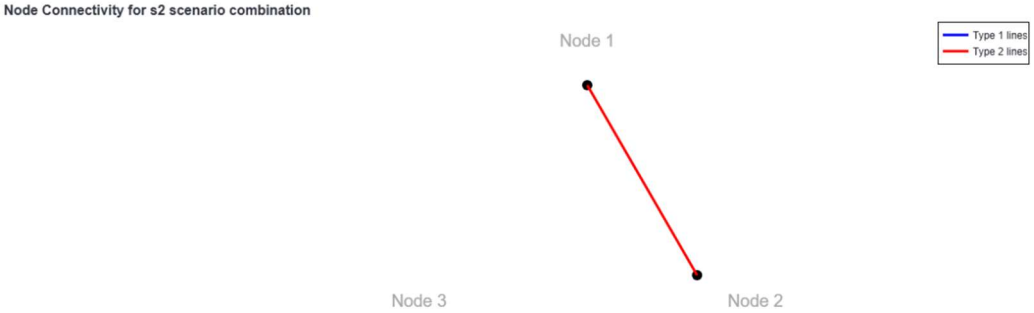


Figura 55

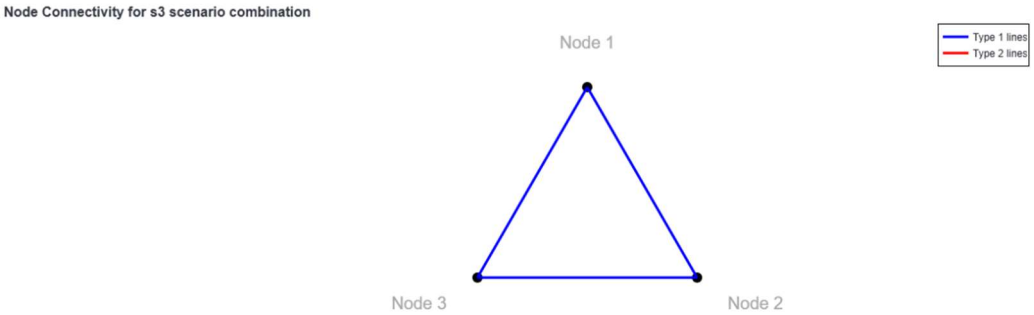


Figura 56

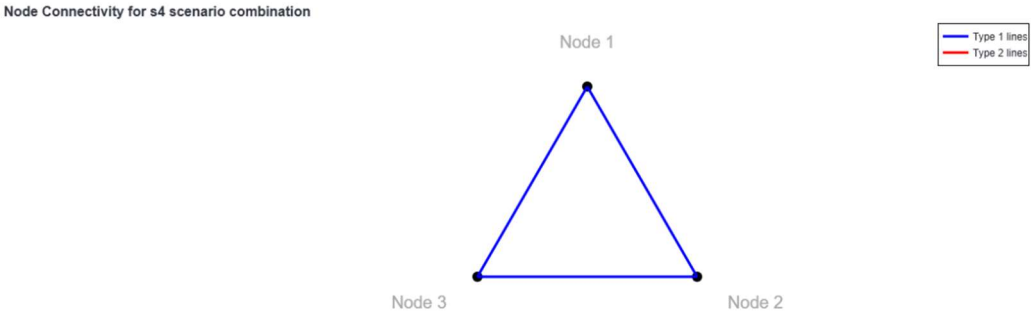


Figura 57

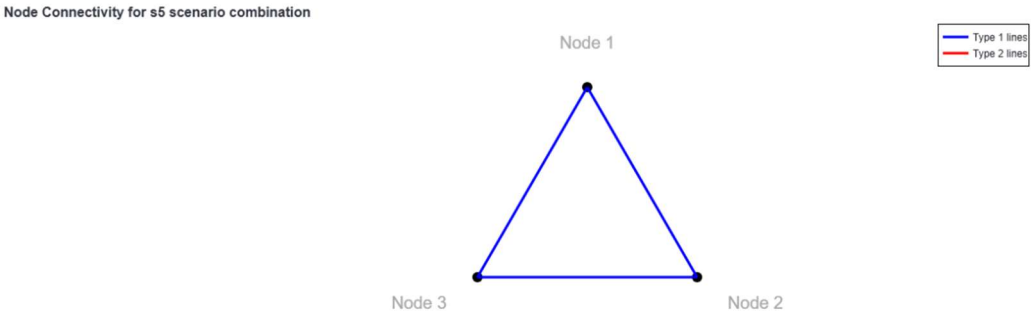


Figura 58

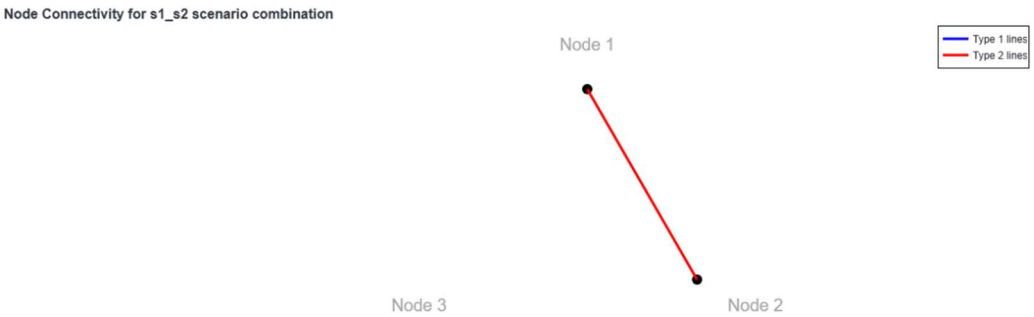


Figura 59

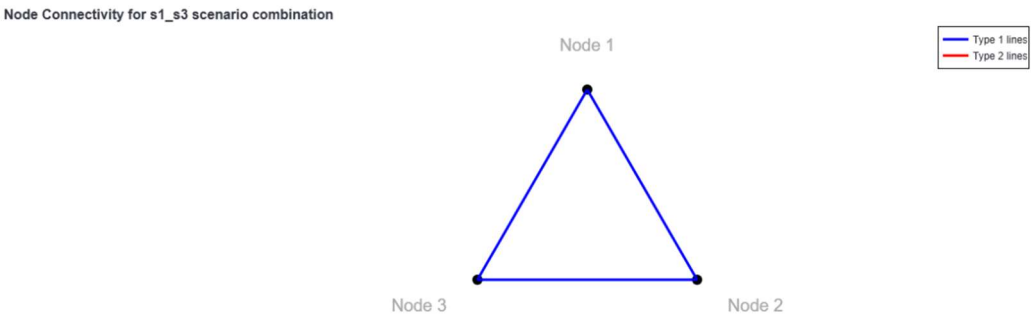


Figura 60

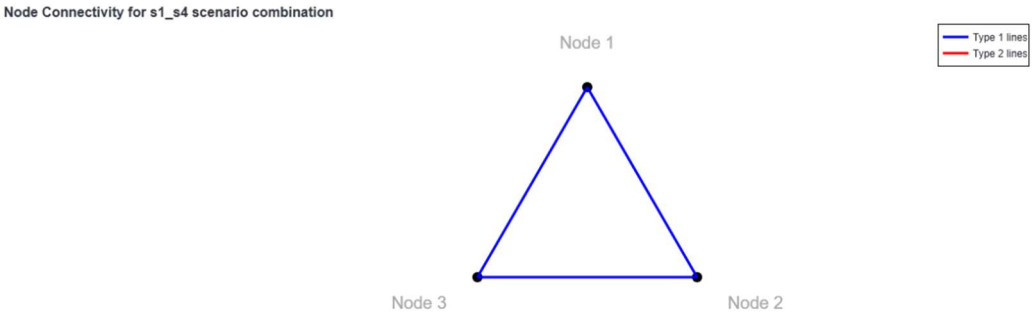


Figura 61

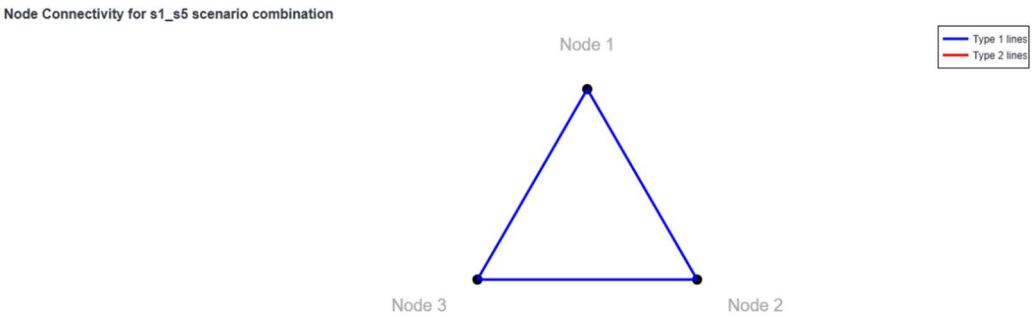


Figura 62

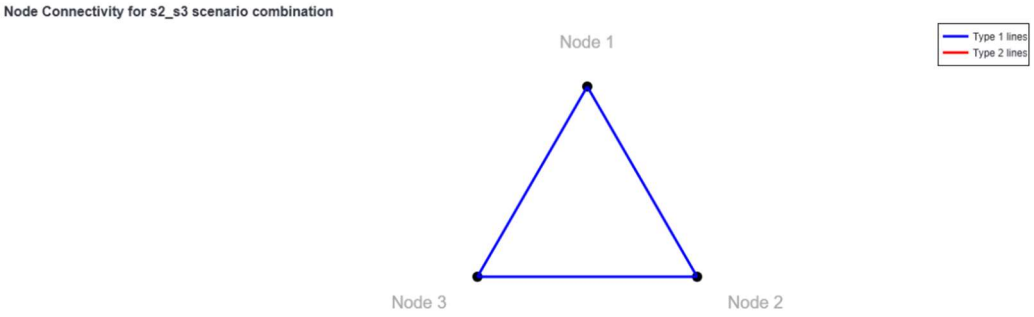


Figura 63

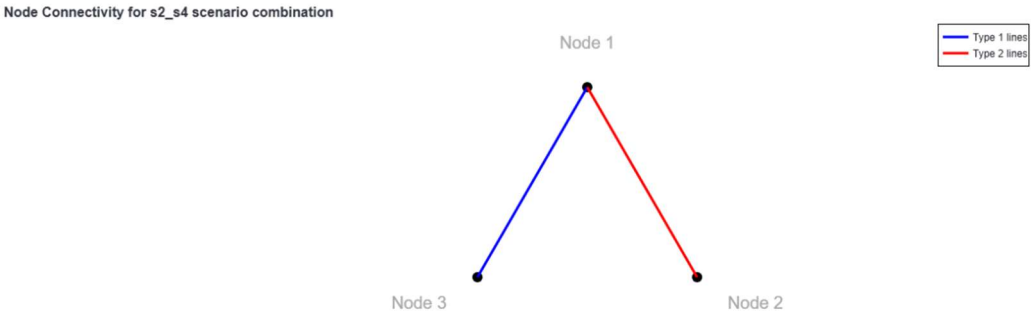


Figura 64

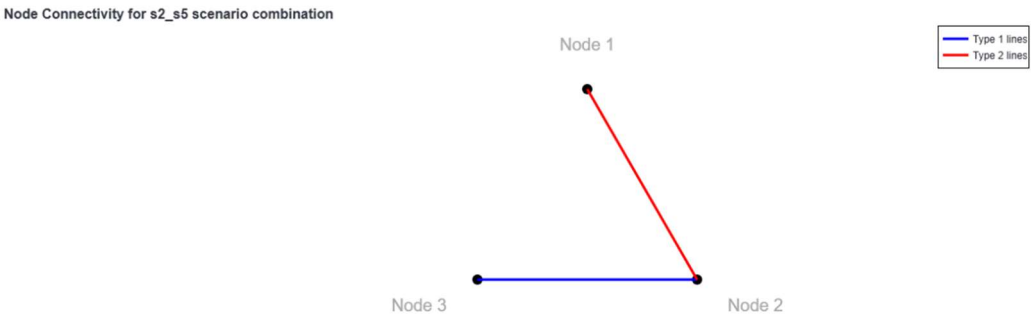


Figura 65

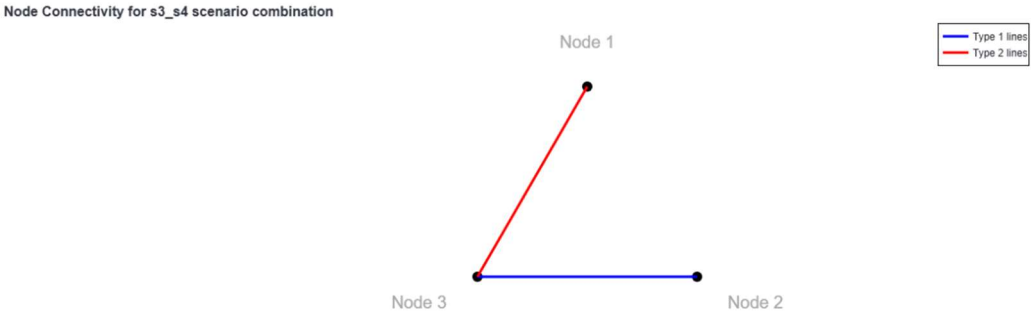


Figura 66

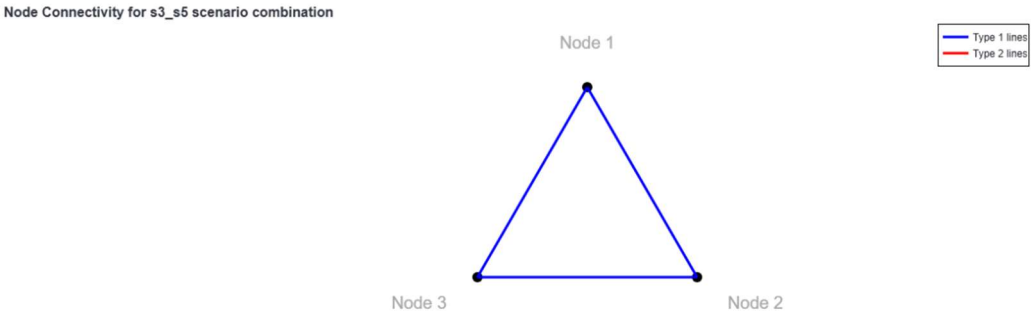


Figura 67

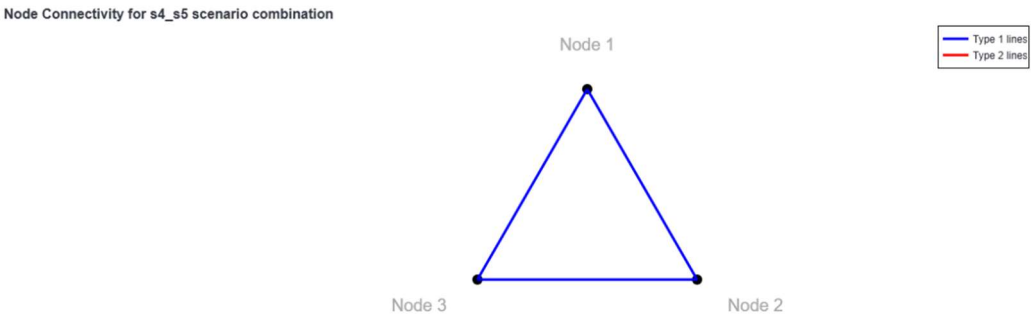


Figura 68

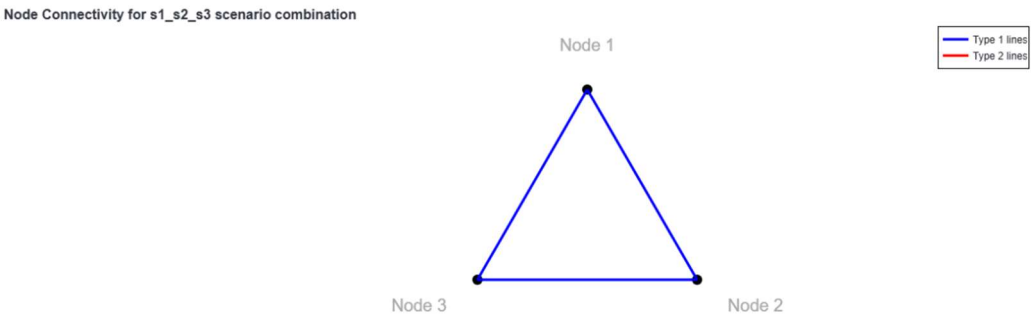


Figura 69

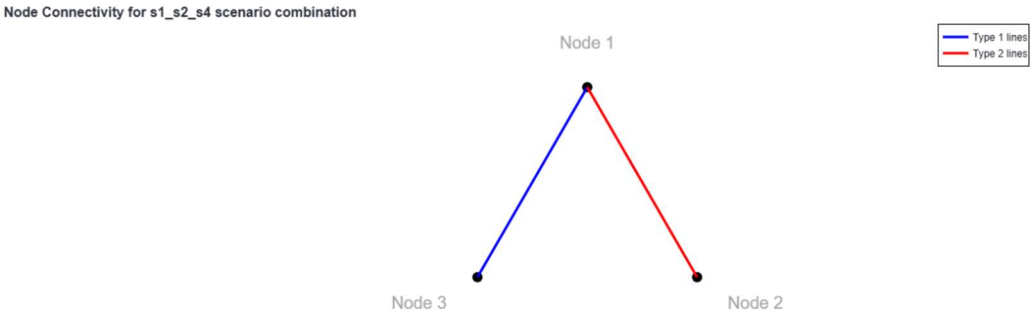


Figura 70

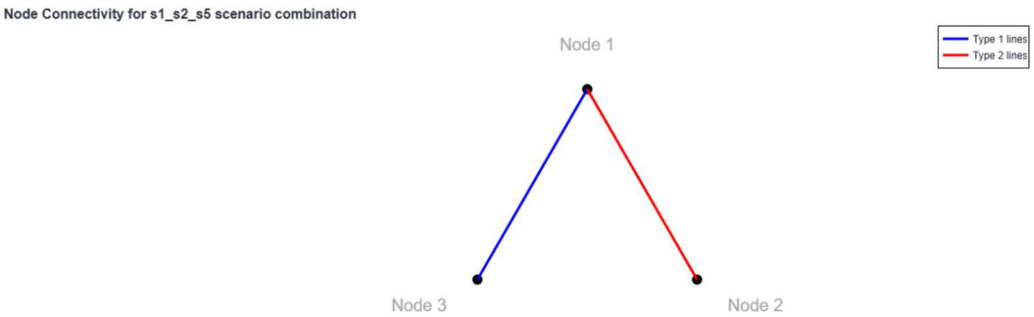


Figura 71

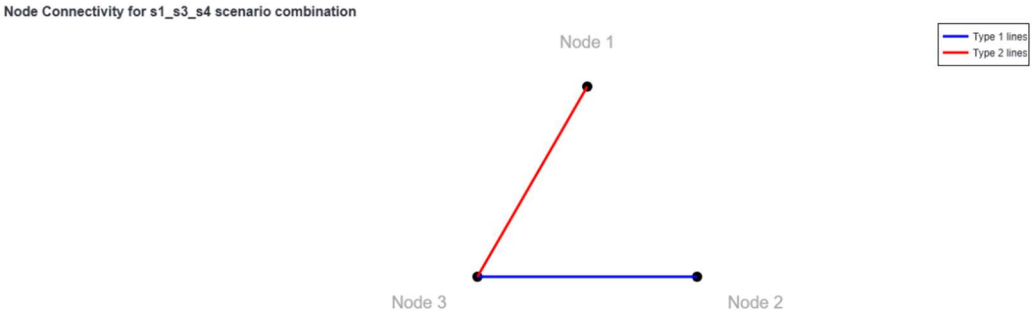


Figura 72

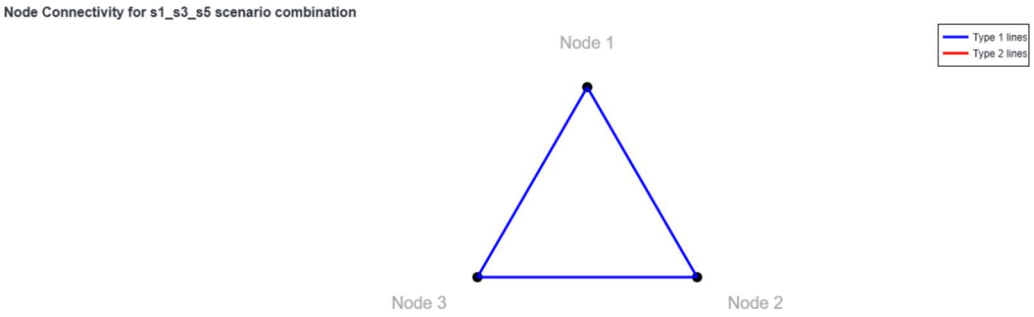


Figura 73

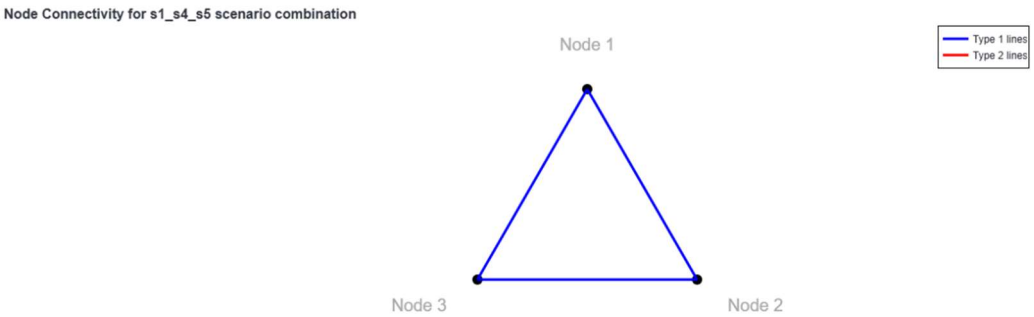


Figura 74

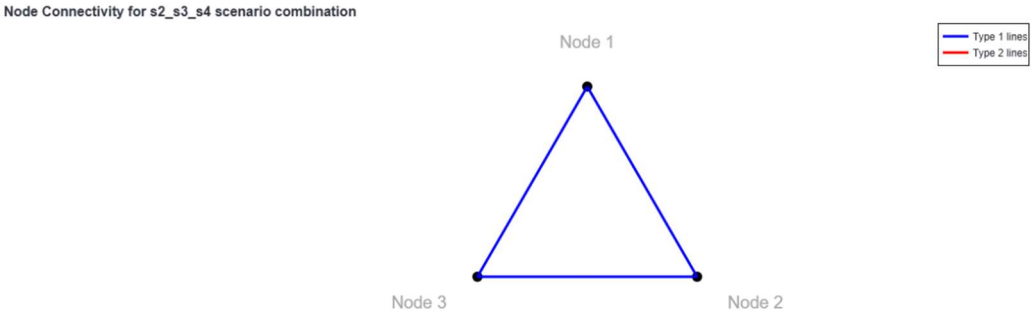


Figura 75

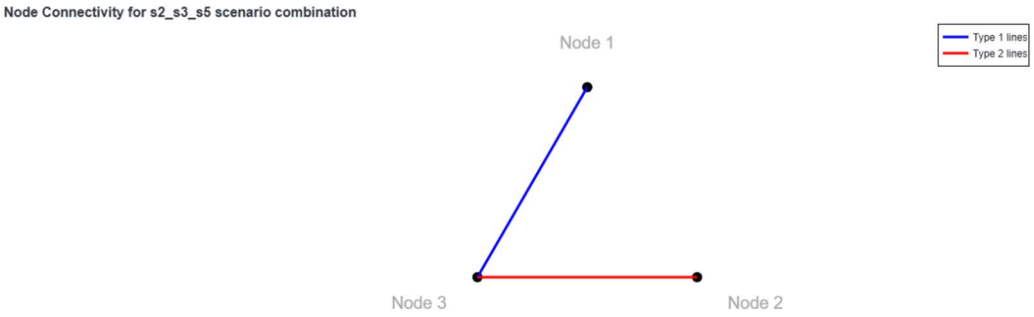


Figura 76

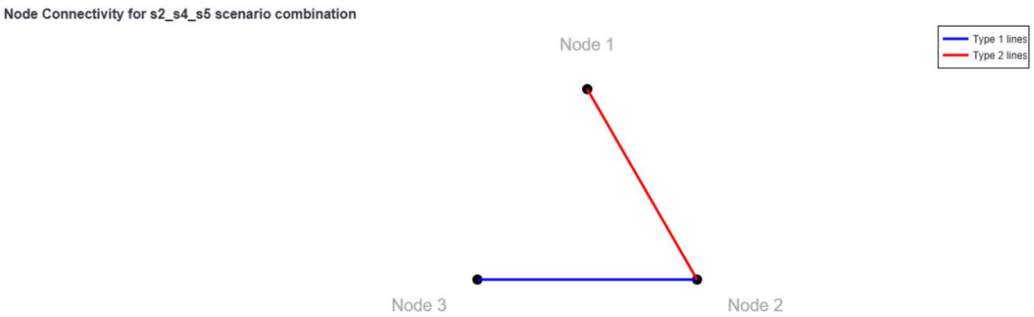


Figura 77

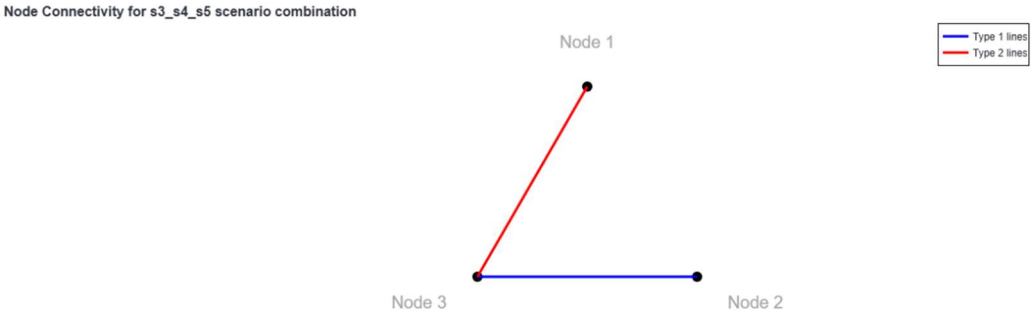


Figura 78

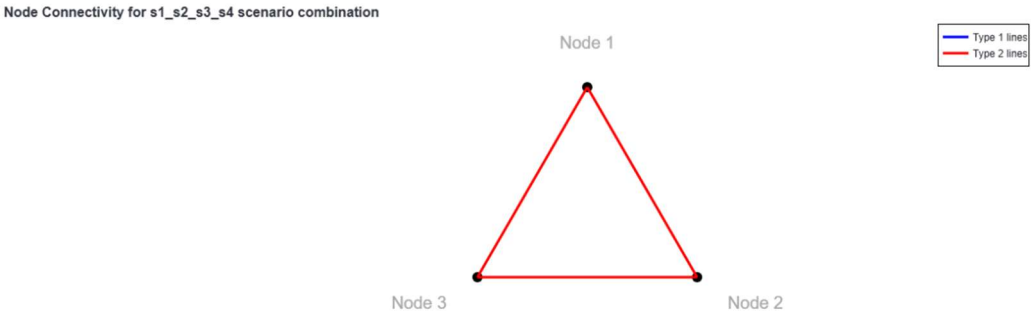


Figura 79

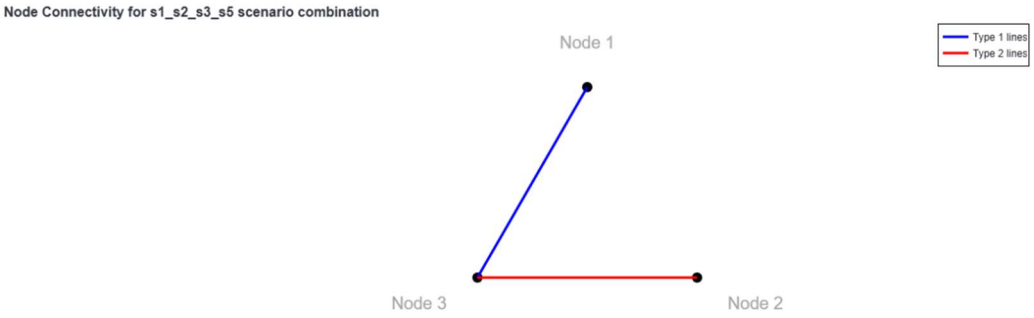


Figura 80

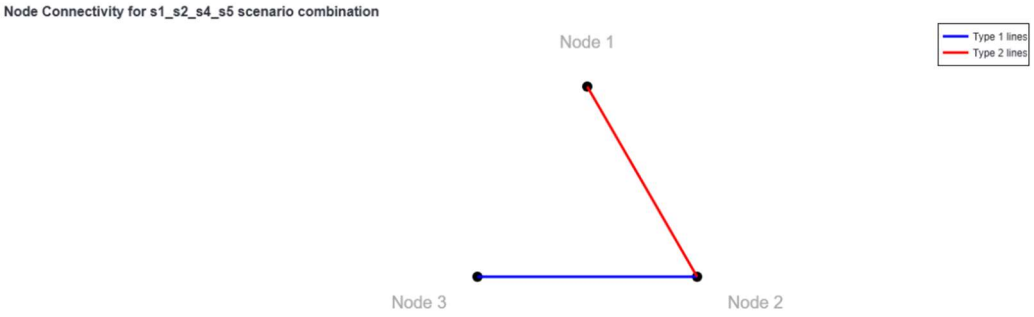


Figura 81

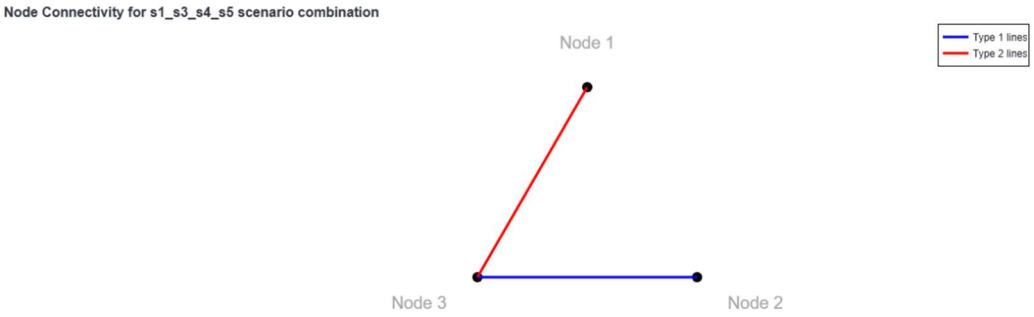


Figura 82

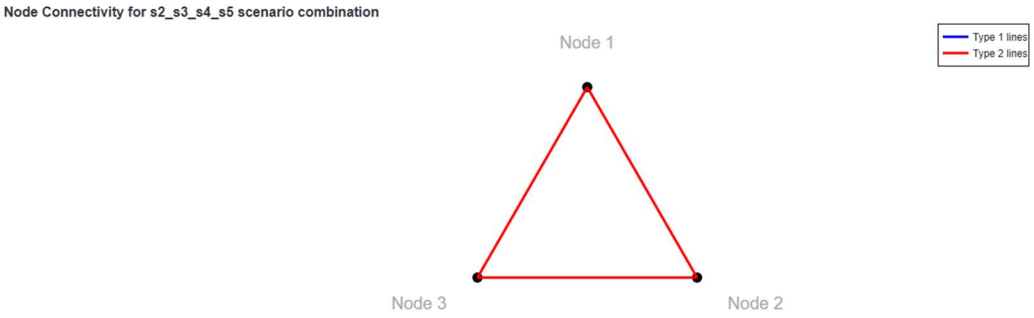


Figura 83

Mapa de calor:

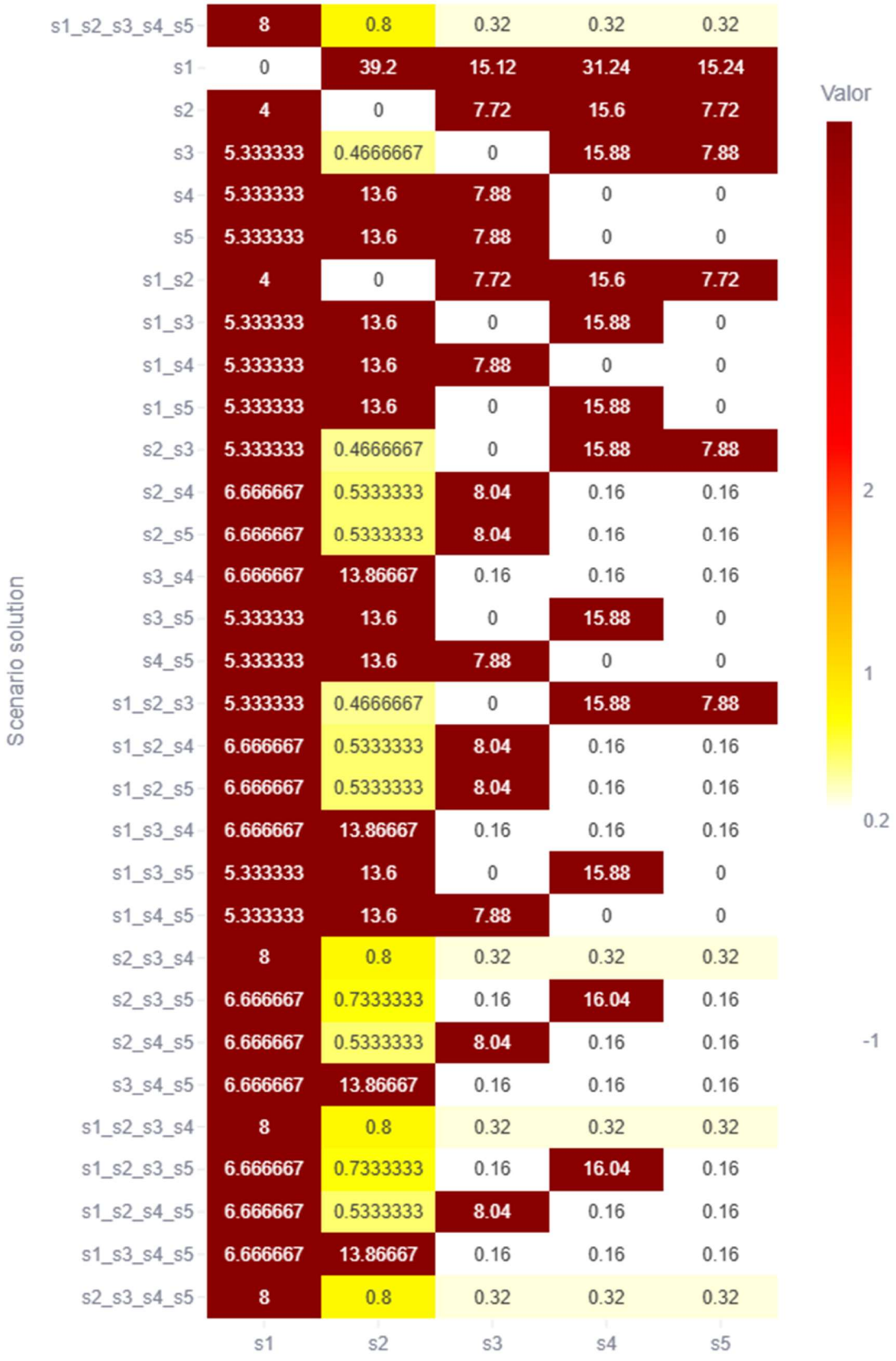

Scenario solution
s1_s2_s3_s4_s5 8 0.8 0.32 0.32 0.32
s1 0 39.2 15.12 31.24 15.24
s2 4 0 7.72 15.6 7.72
s3 5.333333 0.4666667 0 15.88 7.88
s4 5.333333 13.6 7.88 0 0
s5 5.333333 13.6 7.88 0 0
s1_s2 4 0 7.72 15.6 7.72
s1_s3 5.333333 13.6 0 15.88 0
s1_s4 5.333333 13.6 7.88 0 0
s1_s5 5.333333 13.6 0 15.88 0
s2_s3 5.333333 0.4666667 0 15.88 7.88
s2_s4 6.666667 0.5333333 8.04 0.16 0.16
s2_s5 6.666667 0.5333333 8.04 0.16 0.16
s3_s4 6.666667 13.86667 0.16 0.16 0.16
s3_s5 5.333333 13.6 0 15.88 0
s4_s5 5.333333 13.6 7.88 0 0
s1_s2_s3 5.333333 0.4666667 0 15.88 7.88
s1_s2_s4 6.666667 0.5333333 8.04 0.16 0.16
s1_s2_s5 6.666667 0.5333333 8.04 0.16 0.16
s1_s3_s4 6.666667 13.86667 0.16 0.16 0.16
s1_s3_s5 5.333333 13.6 0 15.88 0
s1_s4_s5 5.333333 13.6 7.88 0 0
s2_s3_s4 8 0.8 0.32 0.32 0.32
s2_s3_s5 6.666667 0.7333333 0.16 16.04 0.16
s2_s4_s5 6.666667 0.5333333 8.04 0.16 0.16
s3_s4_s5 6.666667 13.86667 0.16 0.16 0.16
s1_s2_s3_s4 8 0.8 0.32 0.32 0.32
s1_s2_s3_s5 6.666667 0.7333333 0.16 16.04 0.16
s1_s2_s4_s5 6.666667 0.5333333 8.04 0.16 0.16
s1_s3_s4_s5 6.666667 13.86667 0.16 0.16 0.16
s2_s3_s4_s5 8 0.8 0.32 0.32 0.32
s1 s2 s3 s4 s5
Valor
2
1
0.2
-1

Figura 84

Gráfico de barras de los conflictos entre combinaciones (por escenarios agrupados) y escenarios:

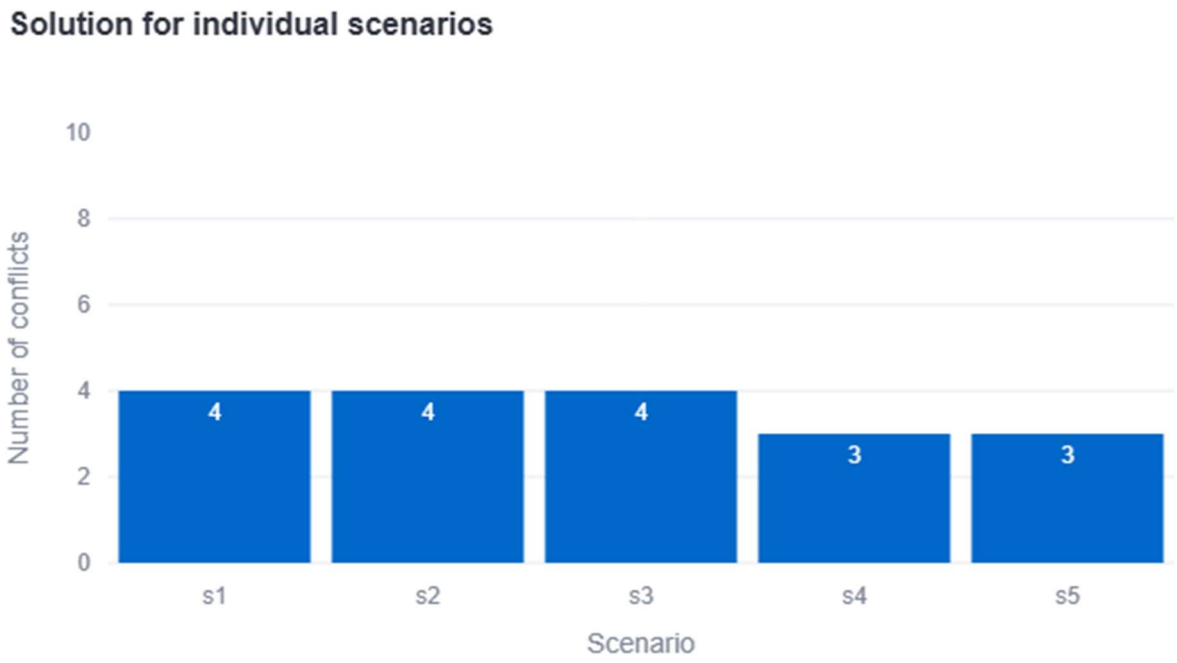


Figura 85

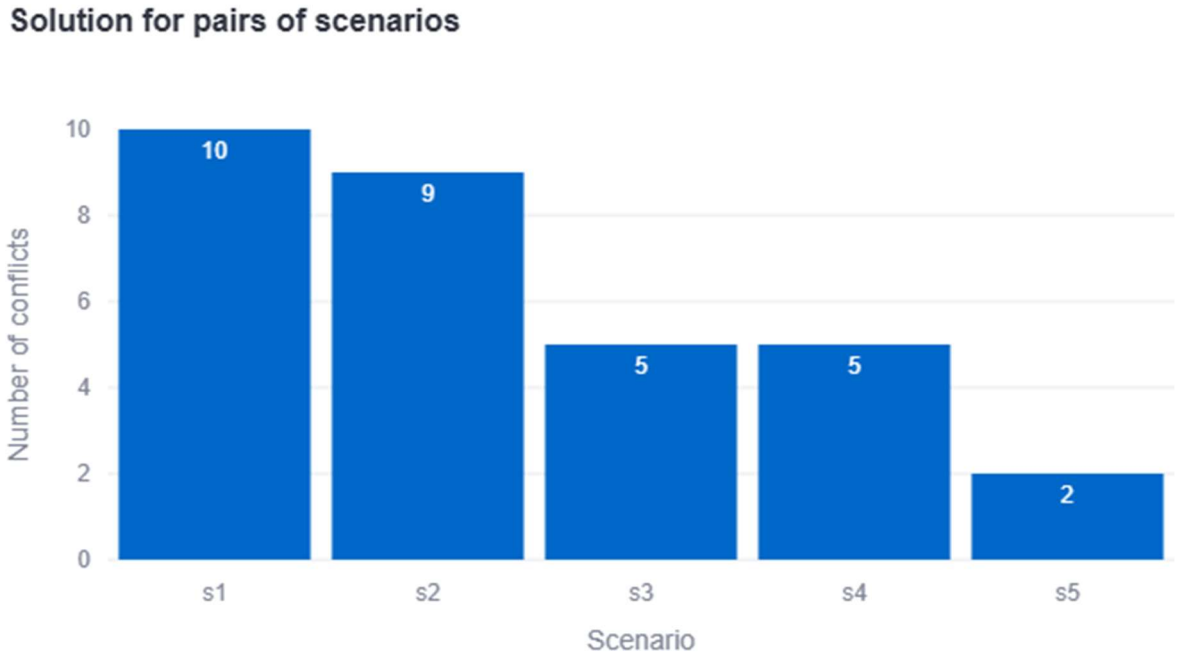


Figura 86

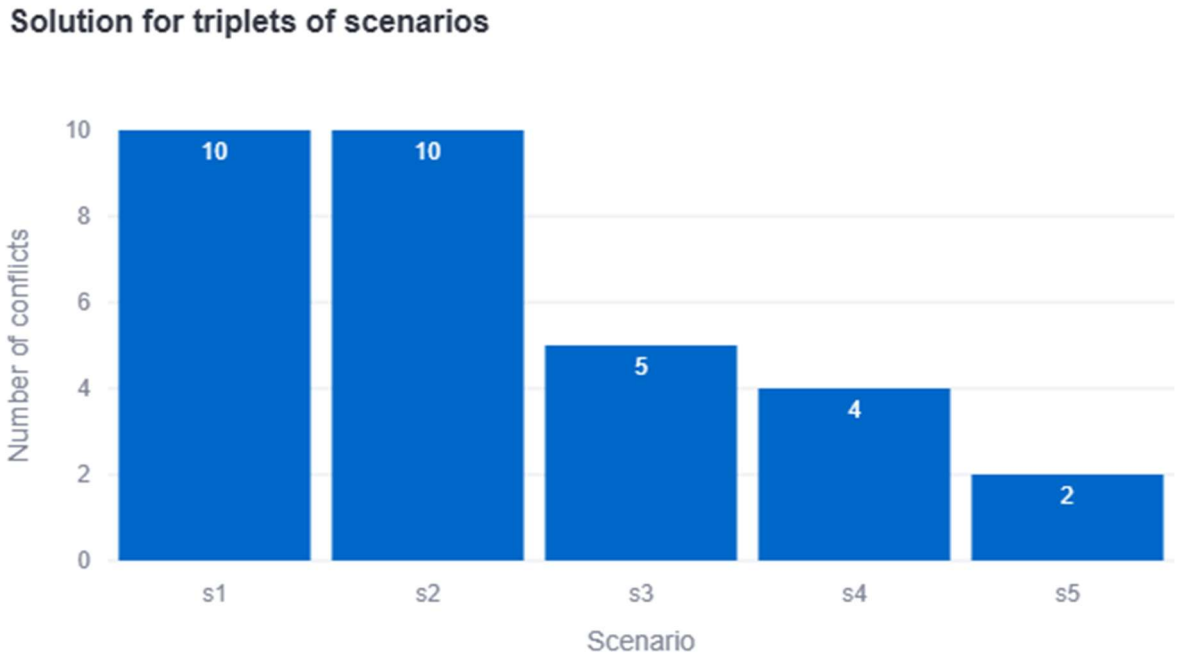


Figura 87

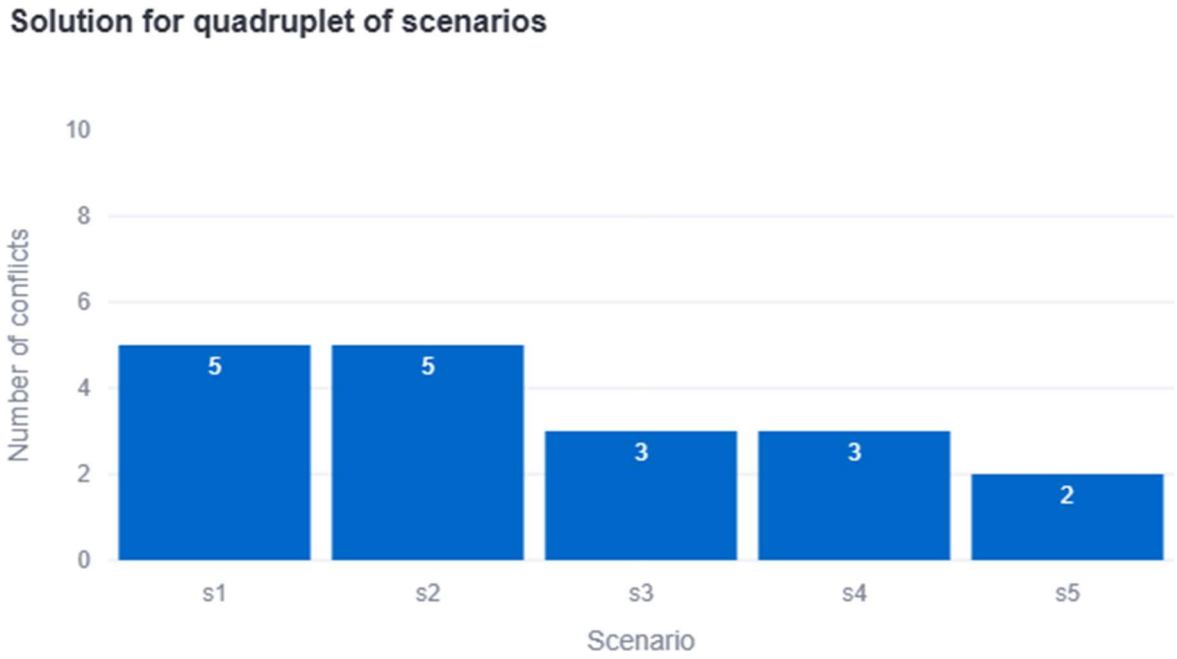


Figura 88

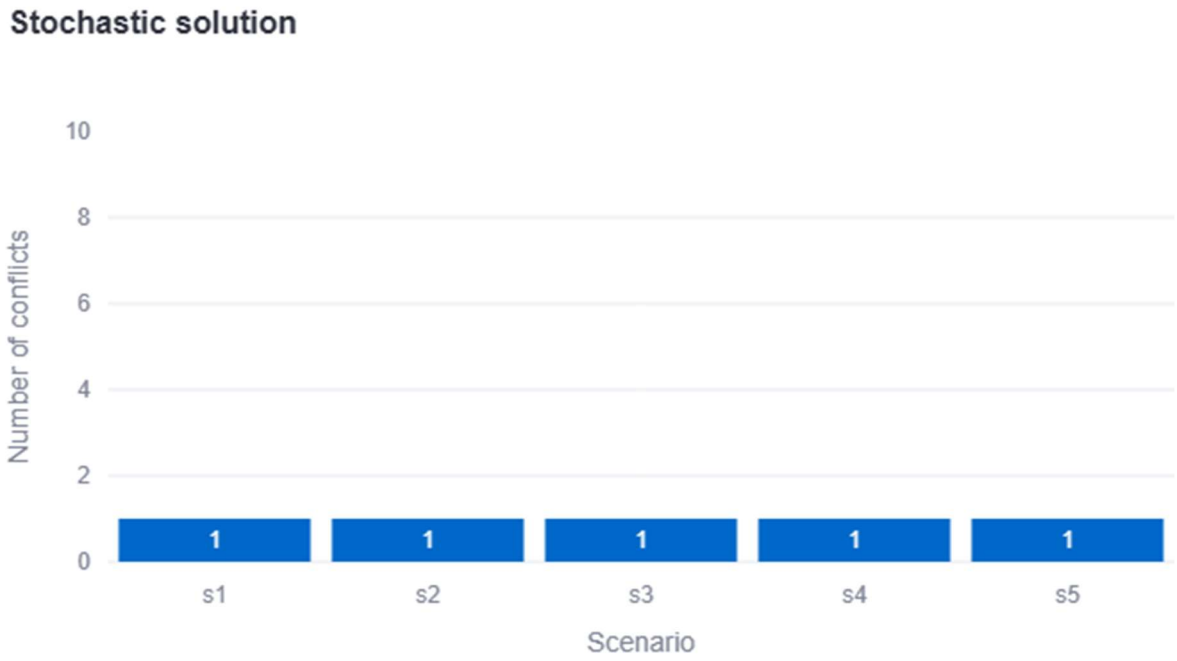


Figura 89

Resolución de conflictos:

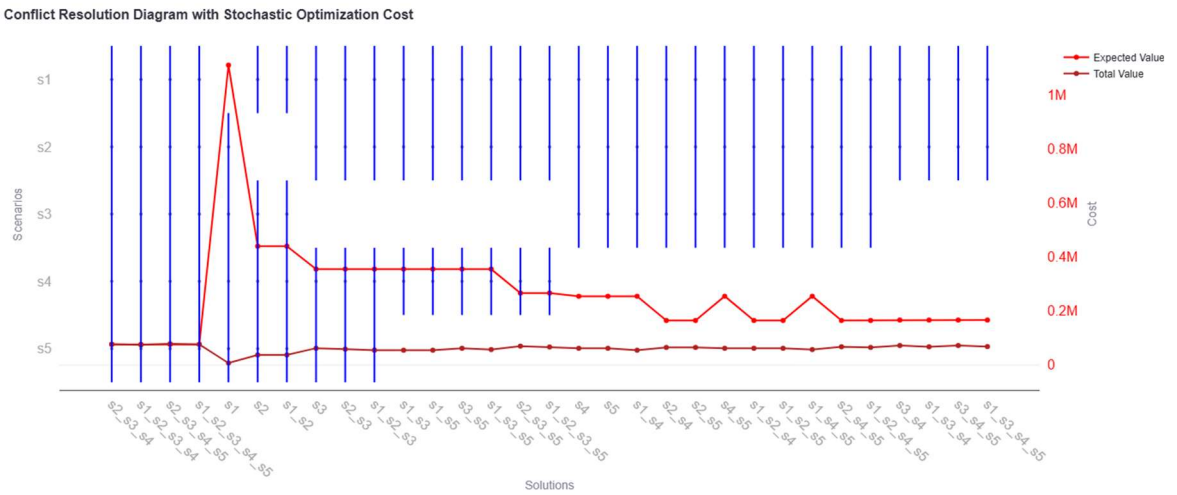


Figura 90

Apéndice: Gráficos avanzados

i. Problema 1

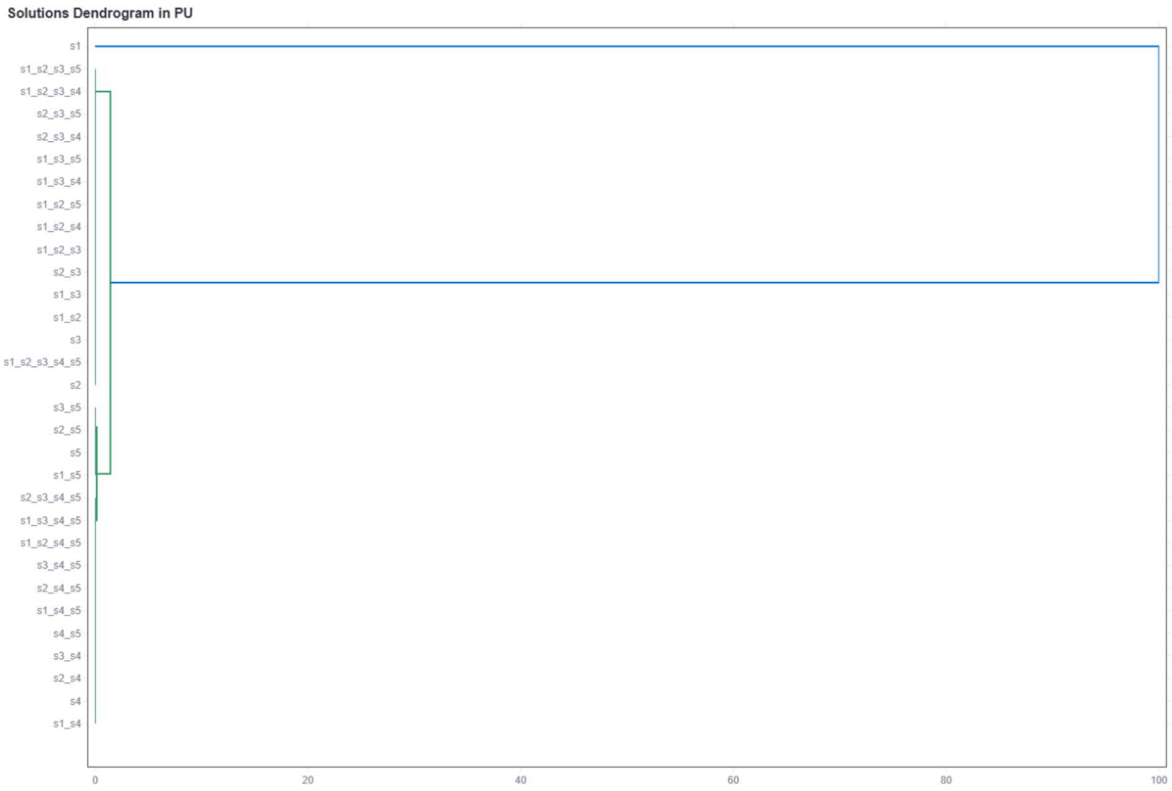


Figura 91

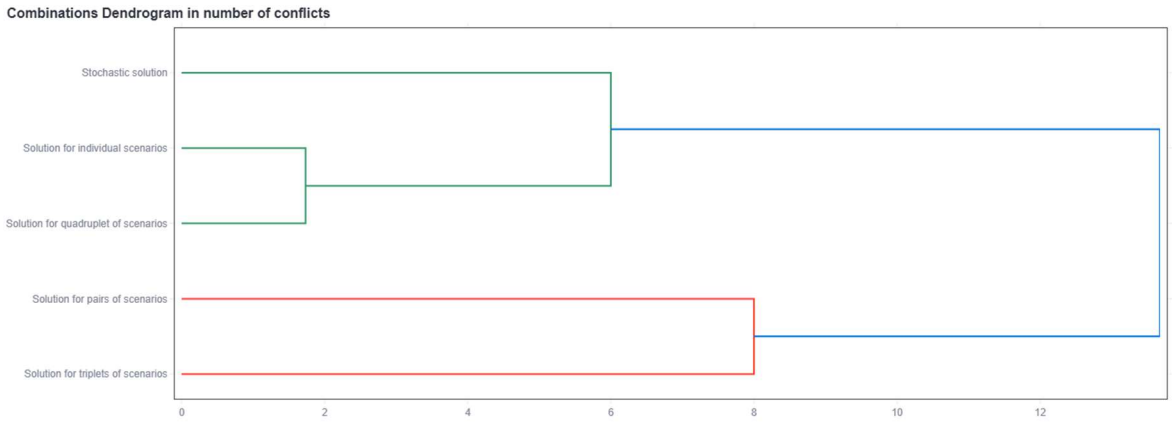


Figura 92

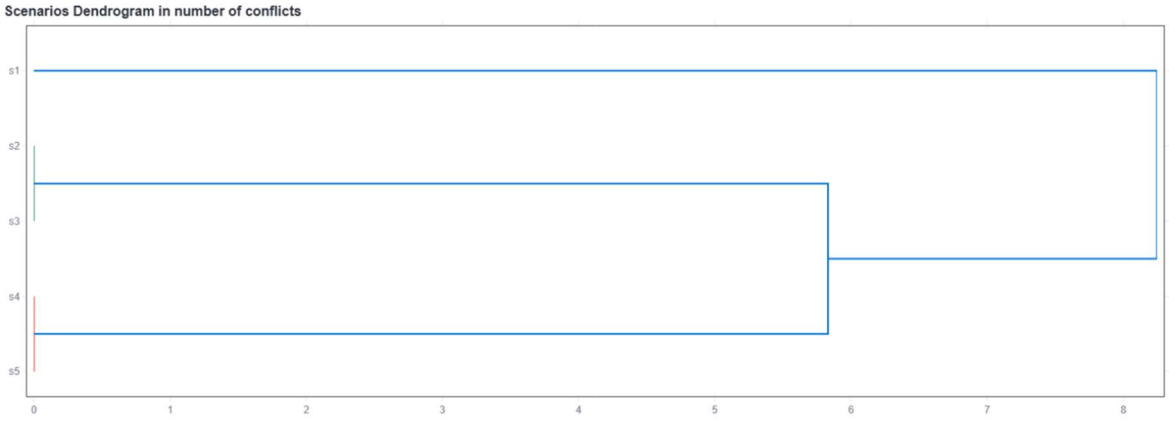


Figura 93

Figura 94: MDS Embedding 2D

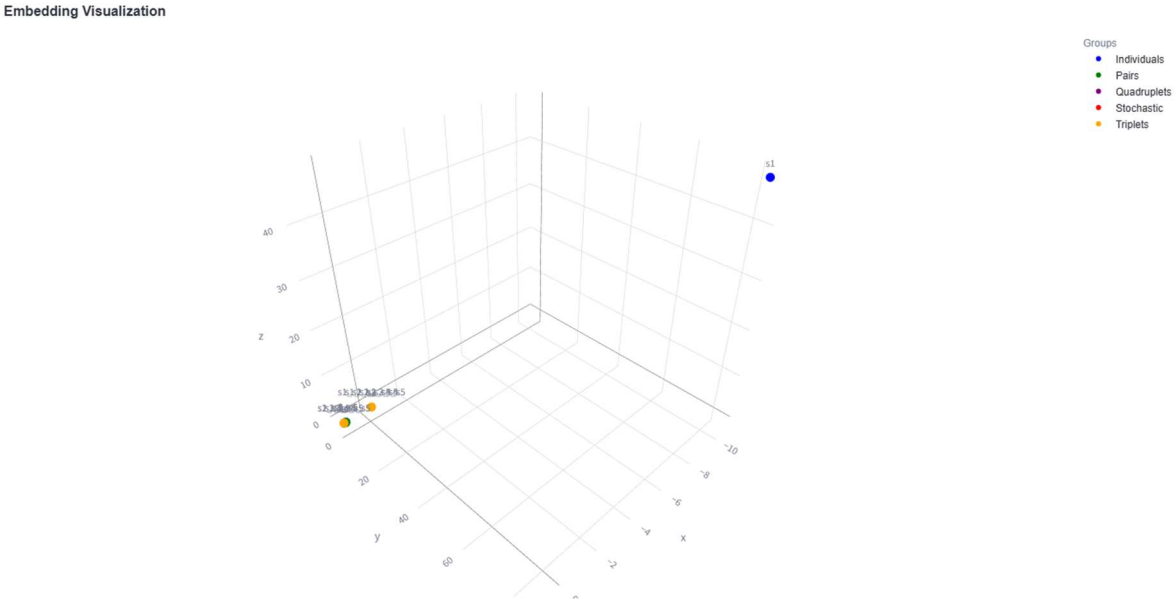


Figura 95: MDS Embedding 3D

Figura 96: PCA Embedding 2D

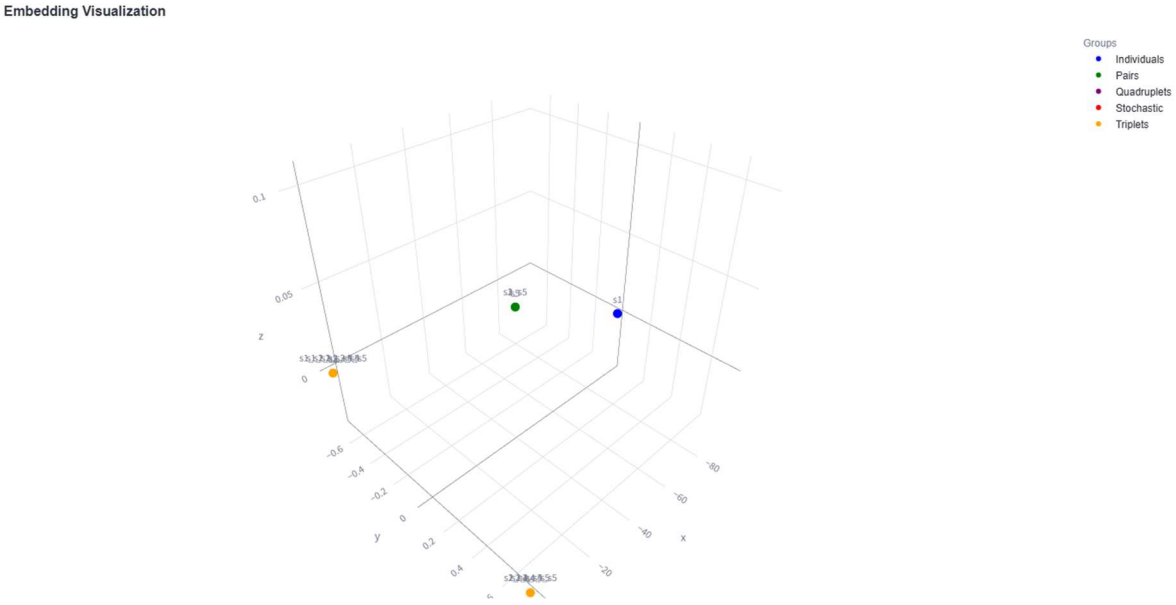


Figura 97: PCA Embedding 3D

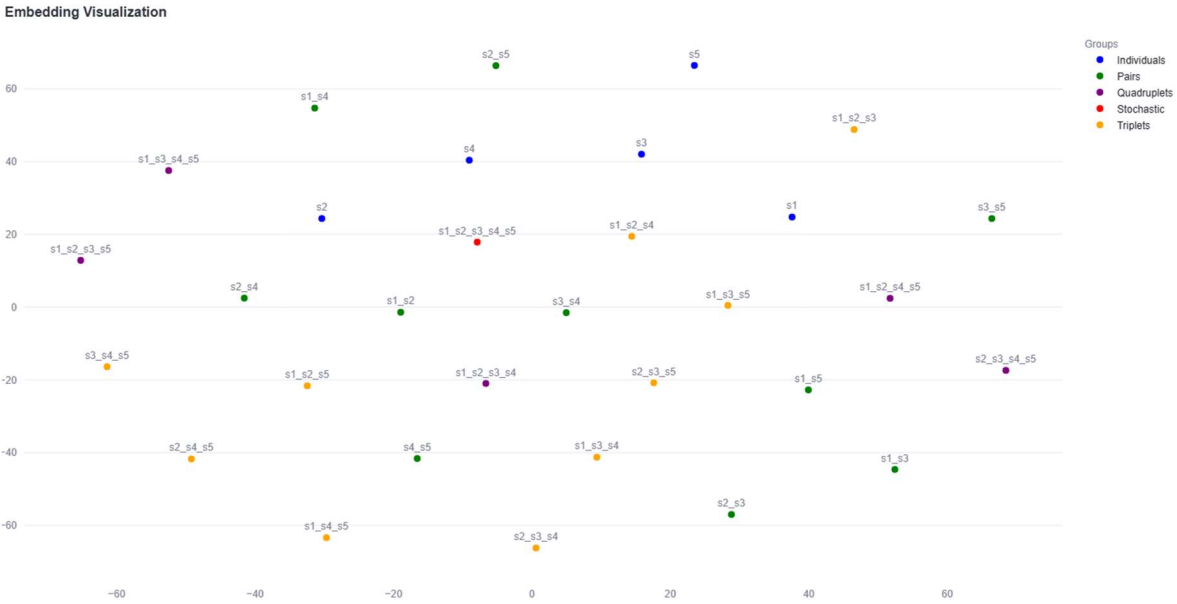


Figura 98: t-SNE Embedding 2D

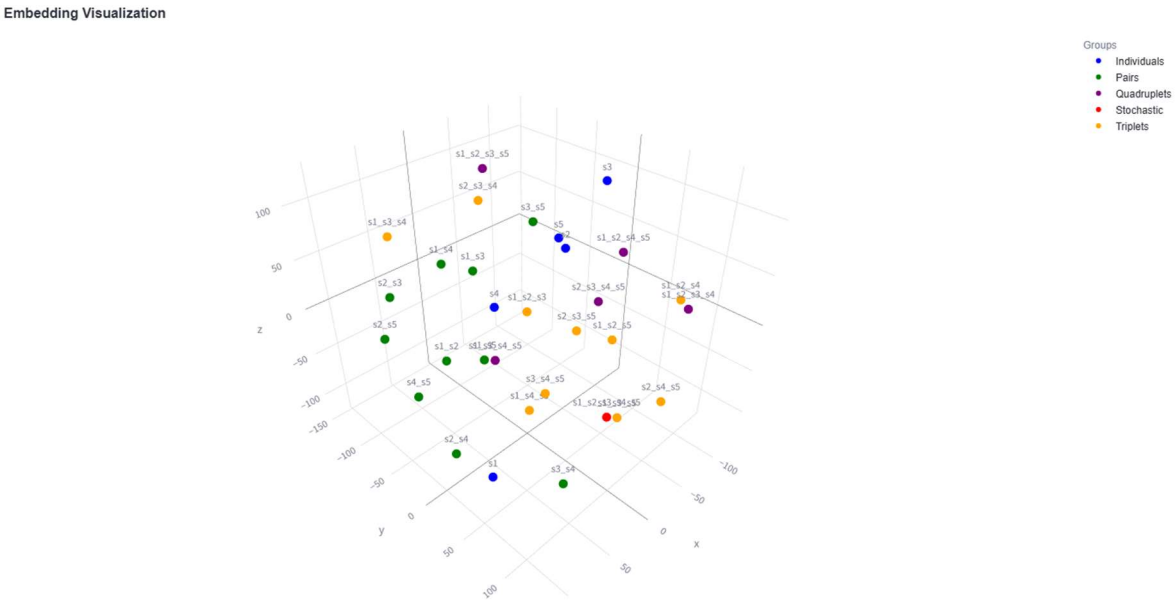


Figura 99: t-SNE Embedding 3D

ii. Problema 2

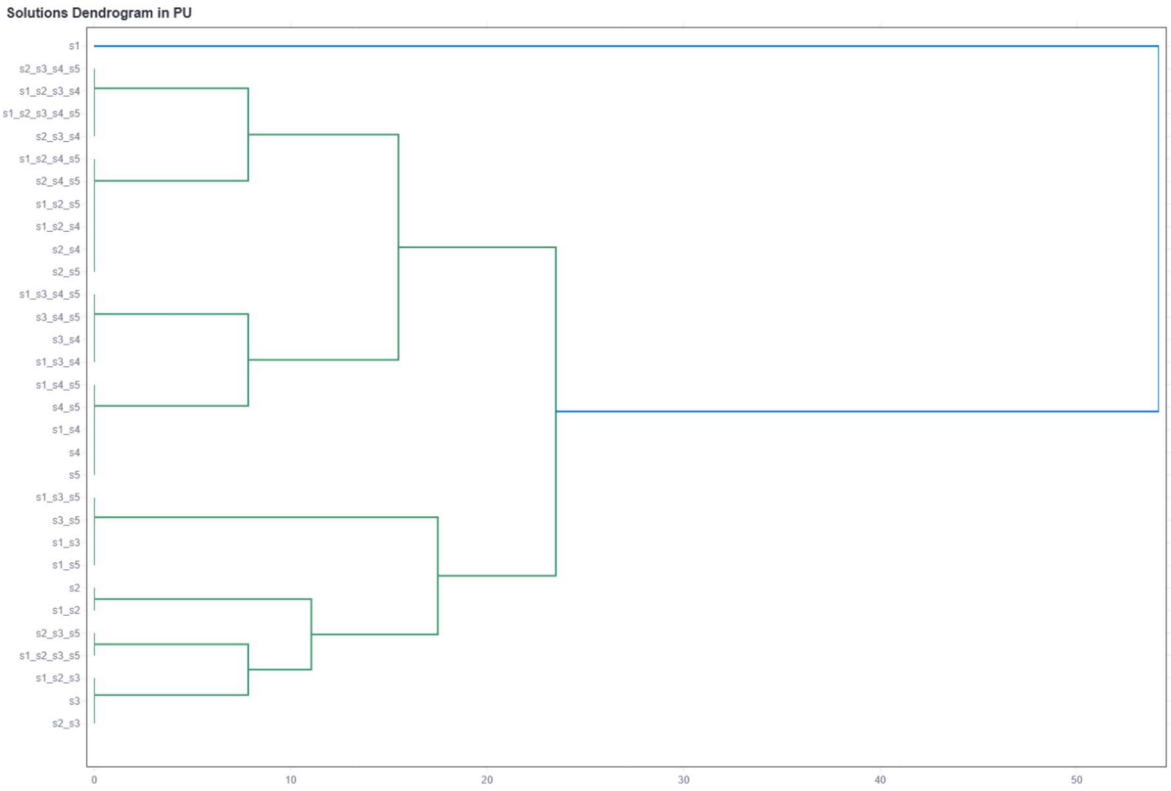


Figura 100

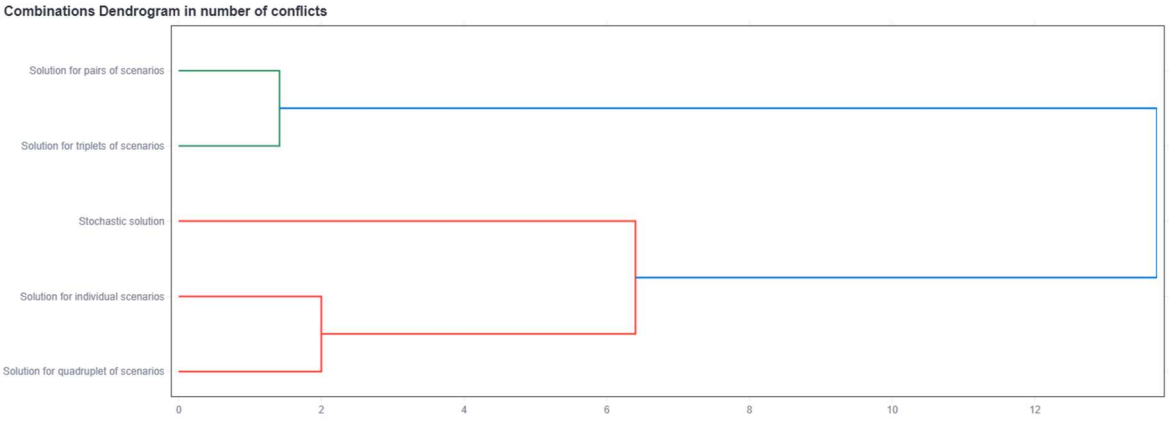


Figura 101

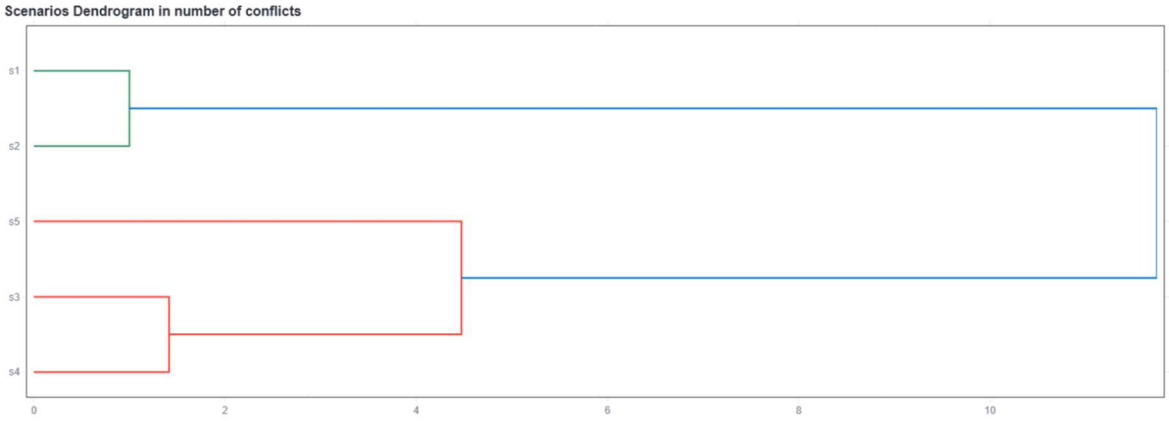


Figura 102

Figura 103: MDS Embedding 2D

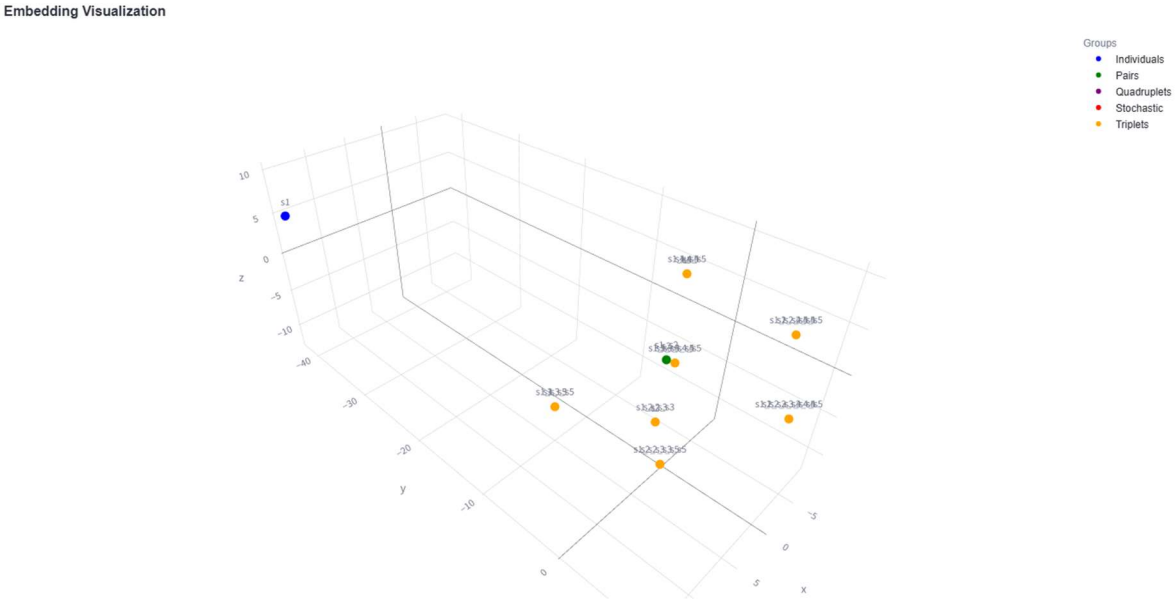


Figura 104: MDS Embedding 3D

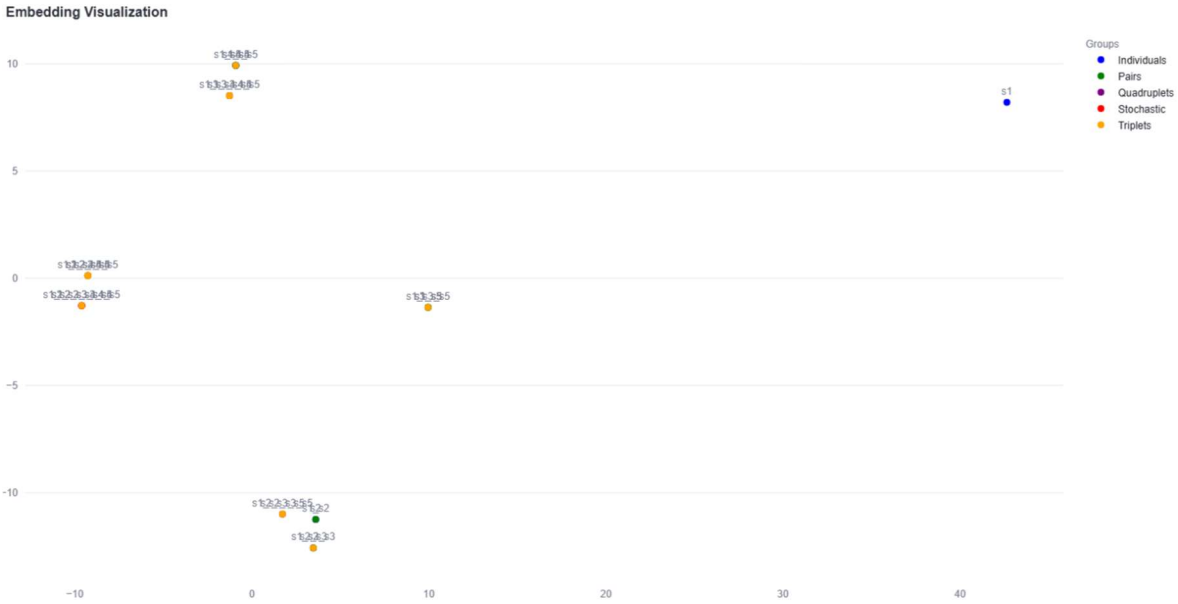


Figura 105: PCA Embedding 2D

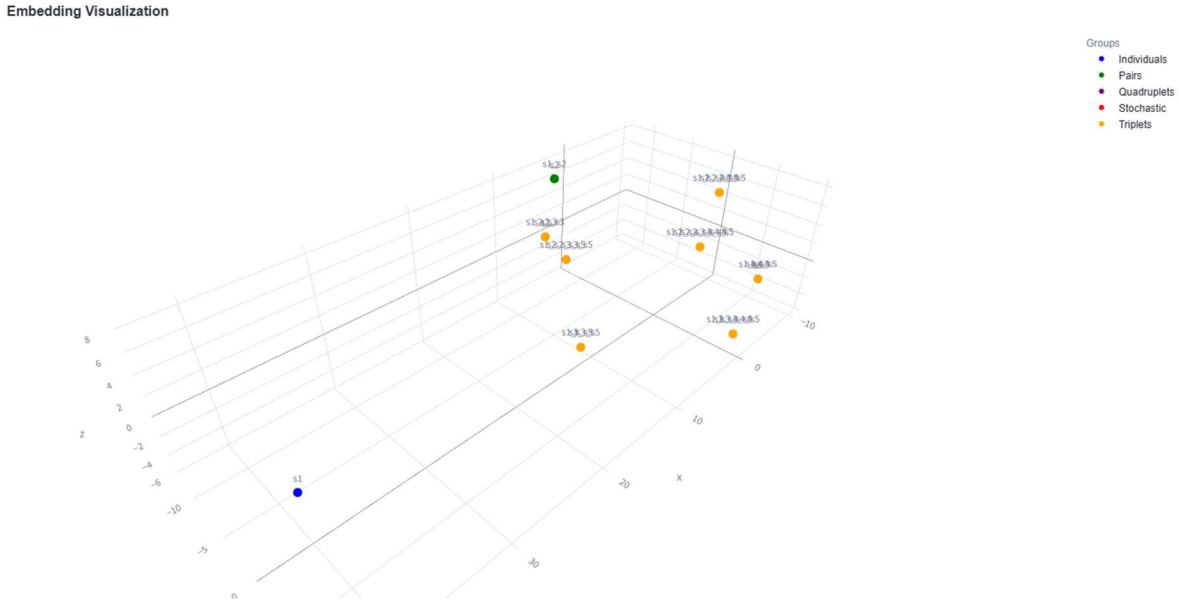


Figura 106: PCA Embedding 3D

Figura 107: t-SNE Embedding 2D

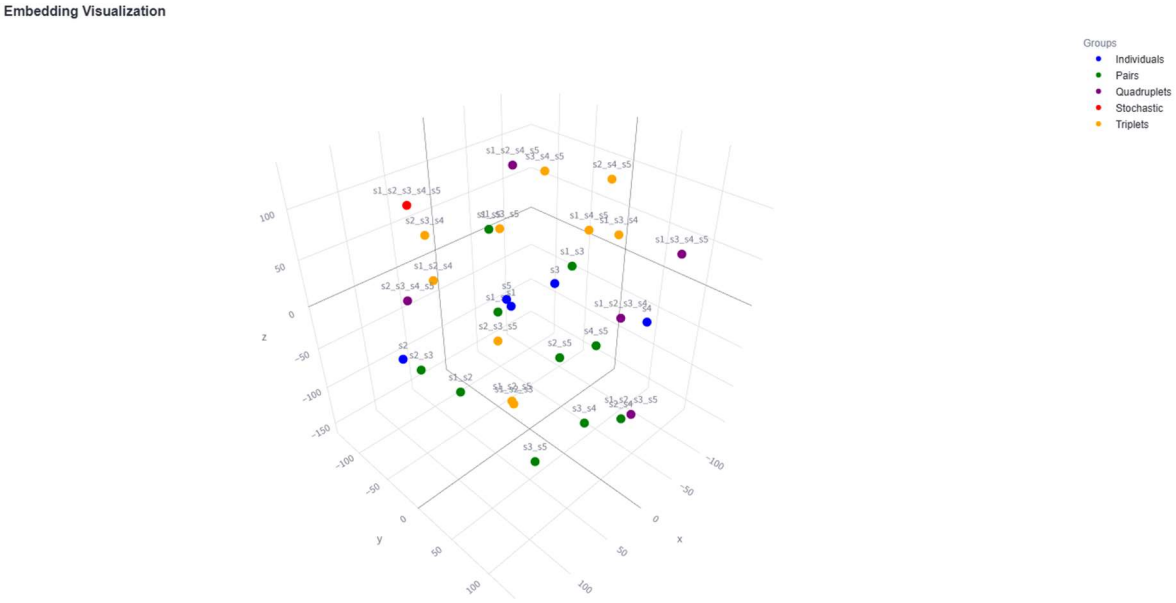


Figura 108: t-SNE Embedding 3D

Recursos

[1] G. B. Dantzig, Linear Programming Under Uncertainty, Management Science, vol. 1, no. 3–4, pp. 197–206, Apr. 1955.

[2] E. M. L. Beale, On Minimizing a Convex Function Subject to Linear Inequalities, Journal of the Royal Statistical Society. Series B (Methodological), vol. 17, no. 2, pp. 173–184, 1955.

[3] A. Ramos and S. Cerisola, Apuntes de "Optimización estocástica", Instituto de Investigación Tecnológica (IIT), Universidad Pontificia Comillas. Disponible online: http://www.iit.comillas.edu

[4] A. Ramos, P. Sánchez, J. M. Ferrer, and S. Wogrin, Apuntes de "Teoría de la decisión" en "Modelos matemáticos de técnicas específicas de optimización", IIT-Comillas. Disponible online: http://www.iit.comillas.edu

[5] A. Ramos, P. Sánchez, and S. Wogrin, Apuntes de "Decisión Multicriteria" en "Mixed Integer Linear modeling", IIT-Comillas. Disponible online: http://www.iit.comillas.edu

[6] Python Software Foundation, Python Language Reference, version 3.x. Disponible: https://www.python.org/

[7] GAMS Development Corporation, GAMS – A User's Guide. Disponible: https://www.gams.com/

[8] Streamlit Inc., Streamlit: The fastest way to build and share data apps. Disponible: https://streamlit.io/

[9] IEEE, IEEE Editorial Style Manual, IEEE, 2023. [Online]. Disponible: https://journal.ieeeauthorcenter.ieee.org

[10] OpenAI, ChatGPT – Modelo de lenguaje IA como apoyo auxiliar para codificación y traducción, 2024. [Online]. Disponible: https://openai.com/chatgpt y otros modelos de lenguaje IA

[11] M. Álvarez Calvo, *Apuntes de Teoría de la Decisión* en la asignatura *Investigación Operativa* impartida por la Dra. Sara Lumbreras Sancho en ICAI, Universidad Pontificia Comillas, 2024/25.

[12] S. Lumbreras Sancho, "Resolución progresiva de conflictos inter-escenario", algoritmo no publicado, IIT, Universidad Pontificia Comillas, Madrid, Spain, 2024.