Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Civil Eléctrica

Una contribución al pre-despacho económico no lineal con indisponibilidad selectiva vía Python

Memoria de título para optar al título profesional de Ingeniero Civil eléctrico.

POR: Jason Fabián Godoy Mura

Profesor guía: Dr. Enrique López Parra

Concepción – Chile, Diciembre 2023

Agradecimientos

Queridos padres, hermanos y profesores:

Hoy, al llegar al final de este viaje académico, no puedo evitar mirar atrás y sentir una profunda gratitud. Este logro no solo es mío, sino también de aquellos que han sido mi apoyo incondicional a lo largo de este emocionante recorrido.

A mis padres, Alejandro y Eilleen, mi roca y mi faro, les agradezco por su constante aliento, sacrificio y amor incondicional. Cada paso que he dado hacia este logro ha sido posible gracias a su apoyo inquebrantable. Su confianza en mí me ha impulsado a alcanzar metas que nunca creí posibles.

A mis queridos hermanos, Nicole, Sachi y Ton, compañeros de risas y cómplices de travesuras, les agradezco por ser mi fuente de alegría y motivación.

A mis profesores, Enrique Lopez Parra (Profesor guía), Claudio Roa Sepúlveda y Rodrigo López González (Comisión) y Leonardo Palma Fanjul (Jefe de carrera), guías sabios en este viaje académico, les estoy profundamente agradecido por su dedicación y paciencia. Sus enseñanzas han sido más que lecciones en libros; han sido inspiración y luz en mi camino hacia el conocimiento. Cada desafío que me presentaron solo fortaleció mi determinación y habilidades.

En conjunto, ustedes han sido las piedras fundamentales de mi éxito. Este logro no solo representa el fin de una etapa, sino el comienzo de nuevas oportunidades y desafíos. Gracias por creer en mí, por alentarme a alcanzar las estrellas y por ser parte integral de mi historia.

Resumen

En esta memoria de título, se desarrolla un algoritmo capaz de integrarse a la automatización del cálculo eficiente de costos medios en el ámbito del pre-despacho económico (PDE). Esto lo realiza mediante la utilización del programa Python. A pesar de que el problema de PDE conceptualmente no es complejo, su solución si lo es. No obstante, la generalidad del código propuesto permite modelar un número acotado de unidades. Desde este punto de vista el “límite” solo lo impone el hardware y el mismo ambiente Python.

Con fines prácticos y de demostración de la robustez del logicial se aplica un número reducido de unidades para que las combinaciones resultantes y el hardware a utilizar no se estresen durante las pruebas. Lo previo, permite evaluar los procesos computacionales, i.e. hacerse de una idea clara hasta donde es posible llegar con el código creado.

Los datos corresponden a curvas de generación no lineales que se aproximan mediante interpolaciones para representar curvas cuadráticas. Desde este contexto, el método tradicional, visto en la literatura, hace llamado a consideraciones simples para que las unidades puedan ser representadas mediante funciones de costo lineales. De esta manera tenemos una representación más cercana a la realidad, dando una visión más realística del rendimiento y la potencia para cada unidad que se agregue al compromiso de unidades del sistema eléctrico y de sus “infinitas” combinaciones surgidas de su periodo de trabajo.

Abstract

In this thesis, an algorithm capable of integrating into the automation of efficient average cost calculation in the context of economic pre-dispatch (PDE) is developed. This is achieved using the Python program. Although the conceptual complexity of the PDE problem is not high, its solution is challenging. Despite the generality of the proposed code, it allows modeling a limited number of units. From this perspective, the "limit" is only imposed by the hardware and the Python environment itself.

For practical purposes and to demonstrate the robustness of the software, a small number of units are applied so that the resulting combinations and the hardware used are not stressed during testing. The aforementioned approach allows evaluating computational processes, i.e. gaining a clear understanding of the extent to which the created code can be taken.

The data corresponds to nonlinear generation curves that are approximated through interpolations to represent quadratic curves. From this context, the traditional method, as seen in the literature, calls for simple considerations so that units can be represented by linear cost functions. In this way, we have a representation closer to reality, providing a more realistic view of the performance and power for each unit added to the commitment of units in the electrical system and its "infinite" combinations arising from its working period.

Índice

Lista de tablas

Lista de figuras

Abreviaciones y nomenclatura

s: Segundos.

DF: Data frames (marcos de datos).

MW: 1.000.000 Watts.

MWh: 1.000.000 Watt hora.

TCIUC: Technology-Clustered Interval Unit Commitment.

UC: Unit Commitment.

MPC: Estrategia de Control Predictivo.

$/h: Dinero por hora.

Excel: Software de hojas de cálculo.

PDE: Pre-despacho económico.

PDENL: Pre-despacho económico no lineal.

CPU: Unidad central de procesamiento.

i.e: Es decir.

Capítulo 1: Introducción

1.1 Introducción general

Cada año con el avance tecnológico acelerado y las nuevas fuentes de energías en pleno auge, la matriz eléctrica chilena enfrenta nuevos desafíos. La constante expansión hacia nuevas fuentes de energía que puedan ser útiles en los años venideros, la industrialización en constante crecimiento son temas presentes tanto entre la población como entre los empresarios. El país necesita prepararse y poseer las mejores herramientas para afrontar los sucesos actuales y por venir, para lo cual se necesita trabajo en equipo y unidad a nivel país desde las distintas áreas involucradas, como: inversión empresarial, marcos regulatorios y fiscalización de parte del gobierno, la información a la población y el aporte de ideas. Estos y muchos otros objetivos y metas se establecen a lo largo de los años por lo que alcanzarlos depende única y exclusivamente de la voluntad con las que se afronten.

En este ámbito el mercado eléctrico es un gran sector a tomar en cuenta y por lo consiguiente para estudiar y proponer soluciones. El pre-despacho de unidades (un término acuñado en esta parte del globo), viene siendo estudiado y analizado cada vez más en los últimos años. Distintos aportes de alrededor del mundo ayudan a comprender y adoptar el enfoque correcto para los desafíos por venir en esta área. Por otro lado, la demanda de potencia con los años crece cada vez más por lo que disponer de un modelo que simplifique las cosas y una metodología adecuada para observar los costos asociados de las generadoras que suplen esta demanda es una ayuda de enorme valor.

No obstante, el modelo y las técnicas actuales no son perfectos, siempre pueden mejorar y someterse a cambios. Así bajo este contexto los profesionales de

ingeniería civil eléctrica son un agente clave para la mejora y solución de estos diversos problemas.

Por lo tanto, observando y estudiando los problemas asociados al PDE, se puede racionalizar y entender fácilmente (como se anticipó) la metodología “iterativa” usada para analizar los costos asociados del problema bajo estudio. Sin embargo, la repetición no es “fácilmente” reducible ya que implica optimizar una serie, bastante extensa, de unidades generadoras, haciendo que el tiempo requerido aumente significativamente, a lo cual se puede agregar una cantidad de información innecesaria al momento de observarlo. En consecuencia, para solucionar este problema se plantea, en esta tesis, una automatización para reducir los extensos tiempos de formulación “manual” y una optimización de soluciones para solo ver los datos esenciales e importantes al momento de analizarlos y estudiarlos.

1.2 Planteamiento del problema

La necesidad de una automatización y optimización del PDE es una obligación al momento de considerar varias unidades en el parque de generación. Se sabe que el tiempo requerido y la información innecesaria crece exponencialmente volviendo prácticamente imposible realizarlo manualmente. En este trabajo, para ejemplificar lo anterior se presentan casos con distintas unidades las cuales están representadas en una base de datos en el programa Excel que simula la producción de distintas unidades generadoras mediante su potencia y costos asociados. Aplicando la metodología estudiada en la Universidad de Concepción, esta arroja combinaciones posibles de producción para distintas demandas, no obstante, la cantidad de información en pantalla procedente de esta metodología se vuelve bastante extensa y redundante por lo que, la solución óptima se vuelve poco clara y difícil. Este informe permite avanzar en la solución del problema proponiendo un algoritmo capaz de optimizar y automatizar el esfuerzo computacional intrínseco.

1.3 Objetivos

1.3.1 Objetivo general

- El objetivo principal de este estudio es contribuir a la solución del PDE no lineal y en paralelo evaluar el potencial del utilitario Python para tal fin. Esto es, concebir un algoritmo computacionalmente eficiente para los cálculos de valores medios para la toma decisiones temporales centrados en el compromiso de unidades o centrales. Por lo tanto, se estudian diferentes estrategias para formular un logicial que posibilite tratar rápidamente las dificultades que implican el procesamiento de múltiples unidades de un sistema eléctrico. Al final del trabajo investigativo, se hace llamado a una serie de aplicaciones (casos) donde se observa la bondad de dicho algoritmo. El fin práctico, de lo anteriormente mencionado, es tratar de procesar y conseguir los datos justos y necesarios evitando la sobrecarga de información computacional al momento de resolver el tema.

1.3.2 Objetivos específicos

- Construir y evaluar casos “clásicos” de 3 y 4 unidades.
- Ver cómo se comporta el código y la automatización en diversos casos donde el número de unidades es grande.
- Implementar y analizar la indisponibilidad de unidades para distintos casos.
- Evaluar el esfuerzo computacional del algoritmo en situaciones de estrés.

1.4 Alcances y limitaciones

- La metodología y la automatización usada solo serán válidas para unidades generadoras que puedan ser representadas mediante curvas cuadráticas. Si el grado aumenta o la curva dista mucho de los estándares que se utilizan en esta MT deberá reconstruirse la solución.

- Las pruebas se realizan en ambiente Python.
- Las unidades representadas en la base de datos en Excel no pueden estar incompletas, las unidades a tratar siempre deben ser distintas entre sí y poseer la misma cantidad filas y columnas.
- La computadora usada en cuestión también limita la obtención de datos y la velocidad con la que trabaja el código en “Python”, en este caso se trabajó con un Intel i5 – 10150 de décima generación, con 16GB de RAM y un disco duro solido NVMe M.2 de 256 GB.

1.5 Temario y metodología

En el capítulo 1 se explica el problema que convoca este trabajo y reconoce las limitaciones de este. Se da una justificación del “por qué” del problema y cuáles son las problemáticas que se quieren resolver, además se establecieron los objetivos por alcanzar, en resumen, se establecieron las bases para la resolución del problema.

Para el capítulo 2 se estudia la teoría detrás de la problemática y los aportes dados en los últimos años. Se menciona el uso de Python para el trabajo y luego se presenta la metodología que se encuentra disponible en la literatura para problemas de esta índole, dicha metodología son los pilares fundamentales de lo realizado a lo largo del código. Además, se deja entrever la problemática de la combinatoria al ir agregando unidades.

En el capítulo 3 se pone sobre la mesa la manera en la que se trabajarán las unidades de producción. Ellas son representadas mediante una base de datos codificada. Este escenario se compone de la información de 3 casos de prueba. Por otro lado, se tiene una explicación del macro algoritmo utilizado en la automatización y optimización de dichos casos y posteriores aplicaciones.

En el capítulo 4 se tienen aplicaciones que fueron sometidas al algoritmo visto en el capítulo anterior, dicho algoritmo está basado en la metodología propuesta de

pre-despacho económico no lineal (PDENL). Los resultados, además, se grafican y comparan con la estrategia conceptual de PDENL. Además, se agregaron pequeñas partes de código para facilitar el análisis de datos y complementar las soluciones obtenidas. Por otro lado, se hizo hincapié en la indisponibilidad de unidades y se estudiaron diversos casos para ver cómo se comportaba el código y finalmente se estudió el tiempo de ejecución para diversas cantidades de unidades en pos de tener una idea clara sobre los límites del algoritmo creado y del programa Python en cuestión.

Finalmente, en el capítulo 5 se habla de manera general de la optimización y las aplicaciones que se realizaron, se señalaron los problemas en los que incurrió la optimización al ser meramente investigativa y no práctica, donde se indicaron algunas soluciones para poder enfocarla de una manera más realista. Se mencionaron ideas, que, si bien aumentan la complejidad de la solución, a cambio brindan una mejor eficiencia y rapidez de esta, vis a vis de lo conocido en el ámbito de la solución numérica del PDE clásico.

Capítulo 2: Marco teórico y estado del arte

2.1 Introducción

En sistemas eléctricos de generación el problema de PDE es uno de lo más urgentes. Este busca determinar de antemano la entrada y salida de centrales generadoras al parque de producción durante un intervalo de tiempo, lo que garantiza un suministro constante de energía.

Dicha programación de generadores, en general, considera aspectos y restricciones tanto técnicas como económicas (start-up, running, shutdown, banking, etc.). No obstante, esta memoria se concentra en el aspecto más importante desde el punto de vista de la optimización económica no lineal del PDE, i.e. el que dice relación con el tratamiento de los costos no lineales

económicos de operación térmica y que en la literatura ad-hoc es tratado de forma simplificada o “frágil”.

En los últimos años los aportes a este tema han crecido en demasía, por lo que la información abunda. El pre-despacho al ser un tema inherente del despacho puede ser representado y modelado de diversas formas según se requiera. Sin embargo, para “una solución excelente” implica inmiscuirse en un universo de posibilidades. Para escoger el camino a seguir y proponer un nuevo método, se investigaron los aportes realizados en los últimos años. Las referencias, detalladas en la bibliografía y estado del arte, fueron las más relevantes para la solución planteada.

2.1.1 Pre-despacho de intervalos y agrupados por tecnología clústeres [1]

Este modelo de optimización llamado "TCIUC” fue diseñado para estudiar el impacto de diferentes métodos de pronóstico en la generación de energía renovable a corto plazo. Utiliza el concepto de optimización por intervalos para representar la incertidumbre en la generación renovable. En vez de utilizar variables binarias para indicar el encendido o apagado de cada generador, utiliza variables enteras para representar el número de unidades disponibles para cada grupo de tecnología térmica, lo que resulta en una representación eficiente desde el punto de vista computacional de la incertidumbre renovable, gracias al bajo número de escenarios requeridos y al uso de variables enteras. No obstante, el modelo no abarca todos los detalles técnicos del “UC” y no incorpora una representación más sofisticada de la incertidumbre, aun así, la reducción en los tiempos de computación lo hace útil para realizar múltiples estudios manteniendo restricciones técnicas relevantes.

2.1.2 Pre-despacho modelo para un Sistema Eléctrico de Potencia [2]

Se busca minimizar los costos de producción, considerando las restricciones asociadas a la generación, transmisión y demanda en toda la red eléctrica. La formulación matemática del modelo resulta en un problema de programación lineal, donde la solución se obtiene mediante una herramienta computacional de libre acceso, la cual consiste en el análisis de distintos casos mediante la utilización de la aplicación Solver de GNU Octave. En este artículo, se hace llamado a tres casos para validar el modelo, abordando situaciones como restricciones de capacidad, metas de generación y desconexión de líneas, en conclusión, busca optimizar la operación del sistema eléctrico, equilibrando eficiencia y cumplimiento de restricciones.

2.1.3 Pre-despacho mediante estrategia de control predictivo MPC [3]

Este presenta una alternativa a la política tradicional de despacho de energía, que incluye un controlador predictivo en la capa de pre-despacho horario. Este enfoque permite considerar mediciones previas, como recursos eólicos, con el objetivo de mejorar el rendimiento de la planificación. Se propone una “MPC” aplicada al despacho horario en la capa media de la estructura de despacho eléctrico. Se centra en combinar la estimación a corto plazo de los recursos con la oferta de generación para lograr un pre-despacho preciso. La MPC se basa en un modelo dinámico, este utiliza un proceso de optimización para calcular señales de control futuras, considerando tanto objetivos como restricciones.

2.2 El uso de Python

Python es un programa de alto nivel, interpretado que ofrece una versatilidad y demás ventajas que la hace posicionarse como una de las más populares para

la resolución de distintos problemas. Sus usos se extienden a diversas áreas siendo las principales desarrollo web, inteligencia artificial, scripting y automatización, siendo esta última la que interesa en este trabajo. No obstante, debido a la extensa cantidad de datos y cálculos que se tiene en el problema de PDE provoca que se dificulte su uso en este ámbito ya que su principal enfoque no va por este camino. Esto hace que la gran mayoría de los estudios relacionados al PDE decida pasar por alto Python y se enfoquen en programas más adecuados para la labor tales como PSS/E, DIgSILENT PowerFactory, etc. Sin embargo, al notar este creciente auge y diversificación de usos que posee Python, se decide indagar e investigar sus capacidades y desempeño para un problema de la índole de la PDE.

2.3 Metodología y formulación

En esta memoria el modelo de PDENL se inspiró, podría decirse, en la manera tradicional vista en la literatura. Consecuentemente, el punto de partida (modelo conceptual) de esta investigación se detalla en el capítulo 7 del libro "*Redes eléctricas, Tercera parte" del autor Jacinto Viqueira, Editorial Representaciones y servicios de ingeniería SA, 1985.* Dicha aproximación se detalla más adelante en este informe.

No obstante, lo previo, están bastante bien difundidos los diversos modelos y métodos de solución del PDE que se sustentan en determinar la salida de potencia de cada unidad generadora que minimizan el costo total de combustible que se necesitará para alimentar la carga del sistema, i.e. aquellos soportados en el despacho económico lineal [4]. En esta pesquisa se considera la distribución más económica de salidas para unidades generadoras conectadas a una barra común, teniendo en cuenta sus límites térmicos, el balance de la generación de activos con la demanda media real y la confiabilidad (incertidumbre de unidades vía la indisponibilidad) del parque de producción. Todo esto último con el fin de evaluar la performance de un "Utilitario/Lenguaje", hoy cada día más aplicado.

En la figura 2.1 se puede observar la curva entrada-salida de combustible en función de la potencia de salida, donde dicha función es cuadrática. El análisis de regresión y consecuentemente los valores interpolados varían considerablemente. Este aspecto, no es atendido suficientemente en la metodología tradicional donde es muy común trabajar con funciones lineales para simplificar el análisis. Sobre este punto es necesario dejar establecido que los modelos de Fuel, casi únicamente usados son: lineales, cuadráticos, cúbicos y exponenciales. Dentro de este listado los más empleados son los dos primeros, dependiendo de su aplicación en el control y la planificación de SEP.

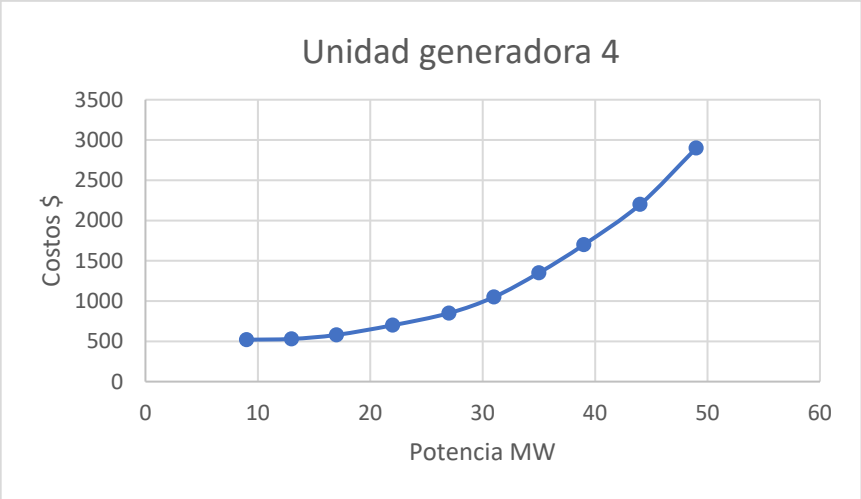

Figura 2.1. Unidad generadora 4

Una vez definida la tendencia de los costos de las unidades generadoras se procede a calcular los costos específicos o también llamados “costos medios”. Estos, en general, se obtienen a partir de la razón F/P de las unidades. Los costos de combustible de entrada al dividirlo por la salida de energía (por ejemplo, MWh), da la proporción de calor expresada en $/MWh. El recíproco de esta tasa es la eficiencia de combustible (esto último es central en el pre-despacho).

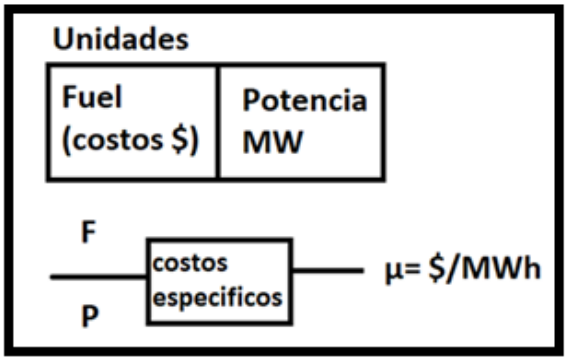

Figura 2.2. Representación costos específicos

$$\mu i = \frac{Fi}{Pi} \tag{2.1}$$

Por lo tanto, siguiendo lo planteado en la figura 2.2 y utilizando la ecuación 2.1, se obtienen los costos medios de las unidades para cada par de datos de combustible-potencia (F, P). Utilizando estos nuevos “puntos de rendimientos” o costos medios de la unidad, se obtiene la gráfica 2.3.

De la figura 2.3 se puede observar como el rendimiento de la unidad 4 crece paulatinamente hasta alcanzar su óptimo entre 20 y 30 MW, posteriormente decrece y la unidad se vuelve menos eficiente al aumentar la potencia de salida.

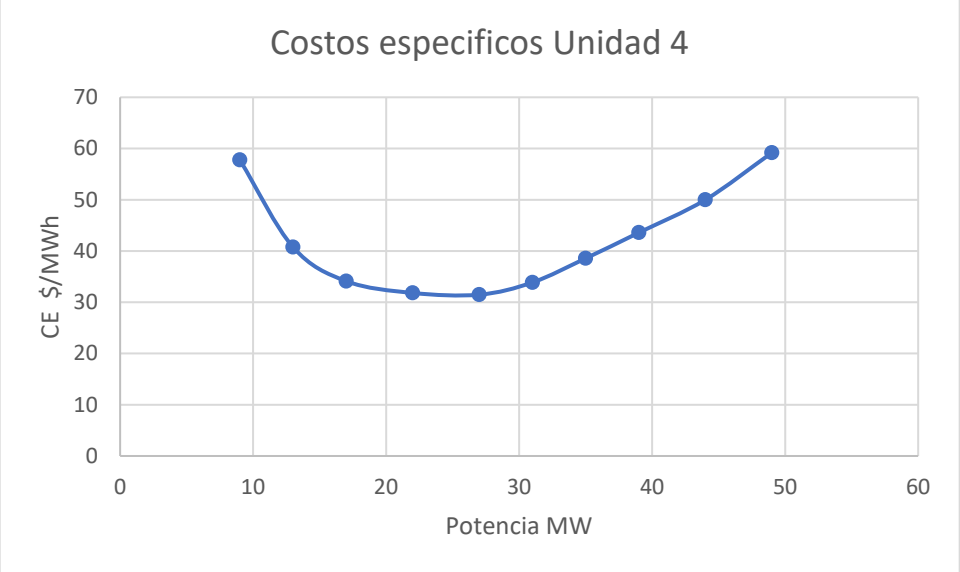

Figura 2.3. Costos específicos unidad generadora 4

2.4 Combinaciones posibles

Esta metodología, al tratar dos o más unidades generadoras, hace aparecer combinaciones que deben ser tratadas análogamente provocando que el conjunto de datos involucrados crezca exponencialmente. Además, por otra parte, si se tienen dos unidades trabajando a la vez, la potencia de salida total obedece a la superposición de las generaciones individuales según la carga total. Este comportamiento no está presente en lo referente a los costos medios. Estos se "componen" dando nuevos valores de costos medios y por consiguiente nuevas curvas de rendimiento, las cuales pueden ser mejor o peor al compararlas con otras combinaciones de unidades generadoras. Por lo tanto, se construyen múltiples combinaciones (de costos medios) para luego comparar las distintas curvas de rendimiento.

De lo anterior, se puede evidenciar que para una unidad generadora se tiene solo una curva de rendimiento posible. Sin embargo, para dos unidades las combinaciones crecen hasta dar tres curvas de rendimiento: Luego para tres

unidades generadoras tenemos siete posibles combinaciones y para cuatro unidades crece hasta quince combinaciones.

Tabla 2.1. Combinaciones según la cantidad de unidades

Número de unidades	Combinaciones posibles (curvas de rendimiento)
1	1
2	3
3	7
4	15
10	1023

Por lo tanto, se puede ver que la problemática es evidente. Al aplicar la metodología tradicional de manera "manual" para 1023 combinaciones, el manejo de información numérica se vuelve inviable debido al abrumador conjunto de datos y el trabajo que este requiere.

Capítulo 3: Soluciones propuestas y automatización

3.1 Bases de datos

3.1.1 Caso 1. Tres unidades

Las unidades generadoras están formadas de la siguiente manera, donde la primera columna representa la salida de potencia, en MW y la siguiente representa la entrada de combustible o el costo asociado al mismo, en $/h.

Tabla 3.1. Datos de las unidades generadoras caso 1

Unidad 1		**Unidad 2**		**Unidad 3**	
MW	$/h	MW	$/h	MW	$/h
10	590	12	500	5	600
20	810	16	550	10	625
28	1040	19	610	15	680
34,75	1280	22	720	20	760

41,25	1540	25	900	25	860
48,25	1840	27	1050	30	1000
55	2210	30	1350	35	1200
64	2700	32	1600	38	1350
73	3310	34	1900	42	1600
80	3810	36	2300	45	1900

Luego al graficar las unidades, como se puede ver a continuación, estas son representadas de forma cuadrática y no lineal como se les trata a menudo en la metodología tradicional. Por lo tanto, teniendo presente lo dicho en el inciso 1.4 de los “alcances y límites”, los aspectos metodológicos explicados en el capítulo anterior, se puede proseguir con la búsqueda de un tratamiento óptimo de la información de las unidades para el PDENL.

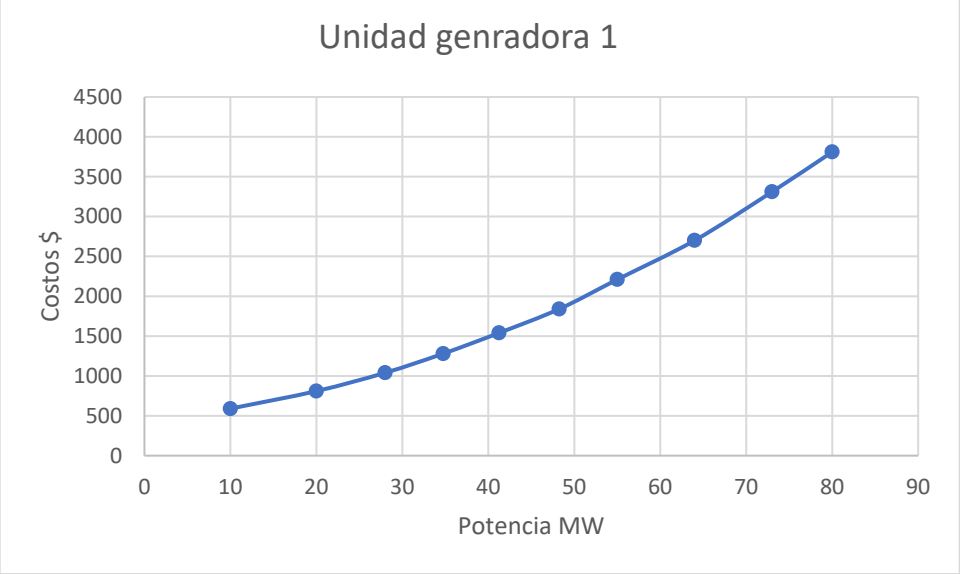

Figura 3.1. Unidad generadora 1

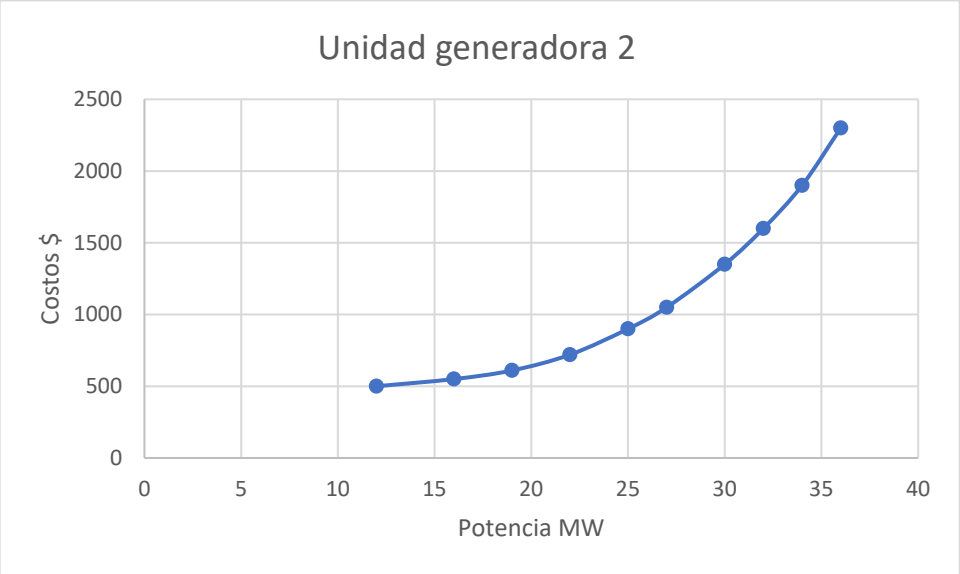

Figura 3.2. Unidad generadora 2

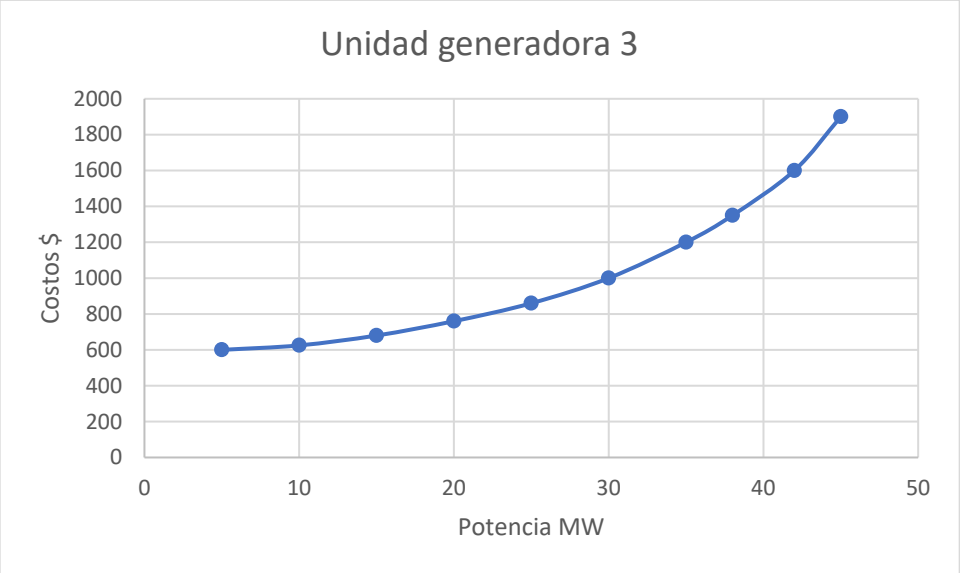

Figura 3.3. Unidad generadora 3

3.1.2 Caso 2. Cuatro unidades

Para este caso, el parque de generación se forma con las tres unidades descritas anteriormente, más una cuarta unidad. Esta ya fue mencionada en el inciso 2.1 y detallada en la figura 2.3. Los datos de esta nueva unidad generadora son los siguientes.

Tabla 3.2. Datos de la unidad generadora 4 agregada para el caso 2

Unidad 4	**Unidad 4**
MW	$/h
9	520
13	530
17	580
22	700
27	850
31	1050
35	1350
39	1700
44	2200
49	2900

Una vez dicho esto, se sabe que esta nueva unidad generadora junto a las anteriormente mencionadas cumple las condiciones impuestas, por lo que se puede proceder al algoritmo y automatización pertinentes para el análisis.

3.1.3 Caso 3. Diez unidades

Como último caso, se tienen las mismas unidades mencionadas anteriormente, pero se le agregan 6 unidades extras para su respectivo análisis y así observar el comportamiento de la automatización en el terreno de los miles de combinaciones donde la carga computacional empieza a ser un factor relevante.

Tabla 3.3. Datos de las unidades generadoras agregadas para el caso 3

Unidad 5		**Unidad 6**		**Unidad 7**		**Unidad 8**		**Unidad 9**		**Unidad 10**	
MW	$/h	MW	$/h	MW	$/h	MW	$/h	MW	$/h	MW	$/h
10	530	11	540	12	550	13	560	14	570	15	580
14	540	15	550	16	560	17	570	18	580	19	590
18	590	19	600	20	610	21	620	22	630	23	640
23	710	24	720	25	730	26	740	27	750	28	760
28	860	29	870	30	880	31	890	32	900	33	910
32	1060	33	1070	34	1080	35	1090	36	1100	37	1110
36	1360	37	1370	38	1380	39	1390	40	1400	41	1410
40	1710	41	1720	42	1730	43	1740	44	1750	45	1760
45	2210	46	2230	47	2240	48	2250	49	2260	50	2270
50	2950	51	3000	52	3050	53	3100	54	3150	55	3200

Siguiendo la misma narrativa de los anteriores casos, todas las unidades cumplen con las condiciones impuestas en anteriores ítems.

3.2 Propuesta algorítmica para Python

El algoritmo que se utiliza a lo largo de la automatización y posterior optimización del PDE, crea un conjunto de datos en el ambiente de Python de manera tal de poder utilizarlos y maniobrarlos a voluntad según se requiera. El cambio de ambiente desde Excel *hasta* Python cubre una necesidad de simplificación y sencillez, debido a que ambos programas son fáciles de usar y sencillos de entender. Sin embargo, en última instancia, la base de datos contenida en Excel obedece a la idea de poseer una versatilidad y globalidad al programar, ya que dicho software (Excel), desarrollado por Microsoft, es ampliamente conocido y por ende se comprometen más posibilidades de integración de herramientas.

En la primera parte del algoritmo, se cargan los datos desde Excel a Python. Dichos datos corresponden a las bases de datos vistas en el capítulo anterior los cuales representan las distintas unidades generadoras. Posterior a esto se crean dos *data-frames* (“marcos” de datos) abreviados “DF”. Estos representan, mediante columnas, el fuel y la potencia de cada unidad generadora.

```python
# Inicia la medición de tiempo
inicio_tiempo = time.time()

# Llamado al excel base para obtener el DataFrame
ruta_archivo_excel = "generadoras.xlsx"
df = pd.read_excel(ruta_archivo_excel)

# Obtener las columnas pares e impares
df_potencia = df.iloc[:,::2]
df_fuel = df.iloc[:,1::2]

# Crear una lista de nombres de columna para ambos dataframes
num_columnas = len(df_potencia.columns)
nombres_columnas = [f'{i}' for i in range(1, num_columnas + 1)]

# Asignar los nuevos nombres de columna a los dataframes
df_potencia.columns = nombres_columnas
df_fuel.columns = nombres_columnas

```

Figura 3.4. Creación de los dataframes a partir de la base de datos en Excel

A continuación, el algoritmo posee una sección controlada por el usuario, la cual tiene por objetivo indisponer alguna unidad generadora o en caso contrario continuar normalmente con el PDE. Luego de este el algoritmo entra en un ciclo combinatorio el cual va creando las distintas combinaciones entre unidades generadoras, el fin de dicho apartado es tener todas las combinaciones posibles entre unidades sin que se repitan, estas combinaciones se van almacenando en sus respectivos DF de fuel o potencia.

Con las combinaciones entre unidades conformadas, se procede a crear un nuevo DF donde se almacenan los costos específicos o costos medios que se calculan a partir de los DF de fuel y potencia. Una vez realizado el cálculo y guardado de los costos específicos se da paso a una visualización gráfica. Esto es, se grafican los datos almacenados en los DF de potencia y costos específicos, dando a lugar la representación de las unidades generadoras junto a sus combinaciones en su totalidad.

A punto seguido, el algoritmo profundiza en los valores mínimos i.e. identificarlos y destacarlos. Para lograr dicho objetivo se crea un nuevo DF, pero a diferencia de los anteriores es dual. Esto implica que unifica los datos tanto de la potencia como de los costos específicos. Este DF es sometido a una interpolación con el fin de crear curvas continuas las cuales van a representar a las unidades generadoras.

Luego, se identifican los valores mínimos presentes en las curvas continuas y mediante un color se representa su procedencia (unidad generadora de la que proviene dicho valor). Al visualizar estos valores gráficamente se posee una única curva con distintos colores. Esta curva, consecuencia de la optimización de la primera parte del algoritmo, muestra los valores mínimos de los costos específicos en función de la potencia.

A modo de complementar el análisis de datos, el algoritmo crea un arreglo de intervalos de potencia los cuales muestran los rangos donde trabajan las distintas curvas optimizadas. Estas se muestran en la consola de Python, además de almacenarse en un archivo Excel.

En la figura 3.5 se puede observar a rasgos generales el paso a paso del macro algoritmo descrito previamente. Por cierto, este resultado debe paralelizar con la serie de tiempo de la demanda, para dar respuesta a la estrategia de pre-despacho y participación de las unidades.

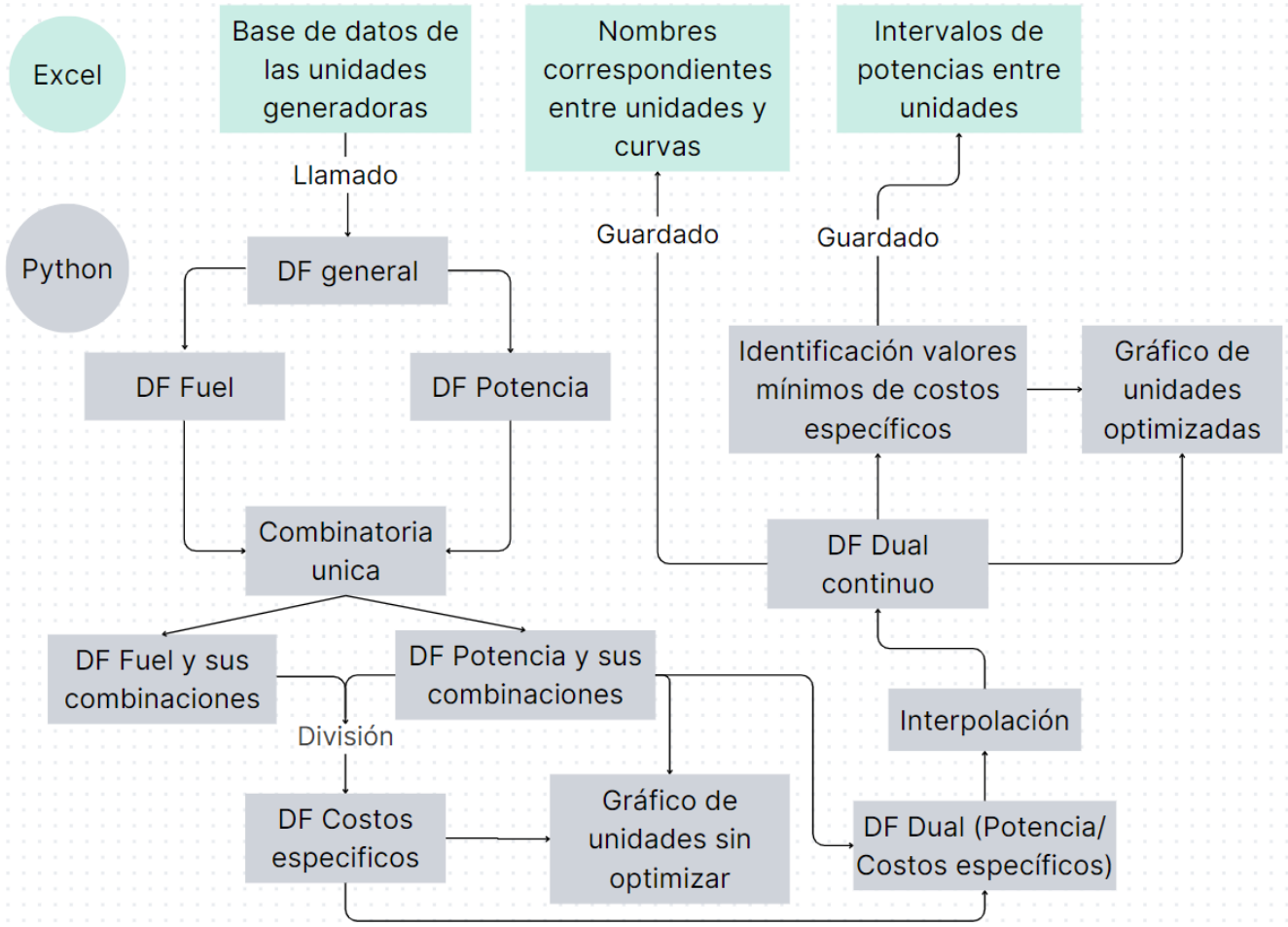

Figura 3.5. Mapa conceptual macro algoritmo

Capítulo 4: Aplicaciones

4.1 Aplicación 1. Tres unidades generadoras

Del capítulo 2.3 se sabe que las combinaciones posibles para tres unidades son 7 (véase tabla 2.1). Por lo tanto, se tiene una noción de la cantidad de curvas esperadas una vez iniciado el código hecho en Python.

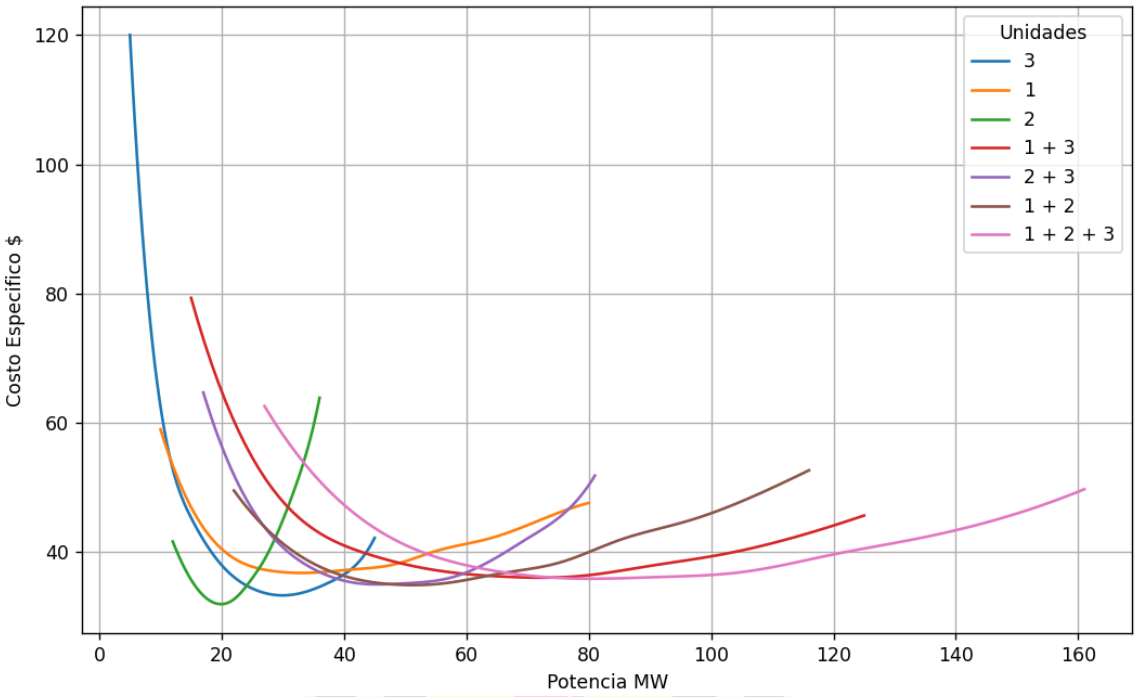

Figura 4.1. Unidades generadoras y combinaciones para 3 unidades

En la figura 4.1, se pueden apreciar todas las combinaciones posibles y sus curvas en detalle, notándose la gran cantidad de información innecesaria en pantalla, recordando lo visto en el “capítulo 2” respecto a los rendimientos, se sabe que la curva más “baja” representa la de mejor rendimiento y por lo tanto la que suplirá la demanda en ese intervalo. Sin embargo, este apartado del algoritmo representa a cabalidad cómo se verían las unidades y sus combinaciones en su totalidad sin ser optimizadas.

En la siguiente figura se tiene una interpolación de puntos donde se tendrá el costo específico de cada curva en función de la potencia, no obstante, para evitar la saturación de información en pantalla, mediante el algoritmo se selecciona siempre el costo más bajo entre las curvas y se graficará para su análisis.

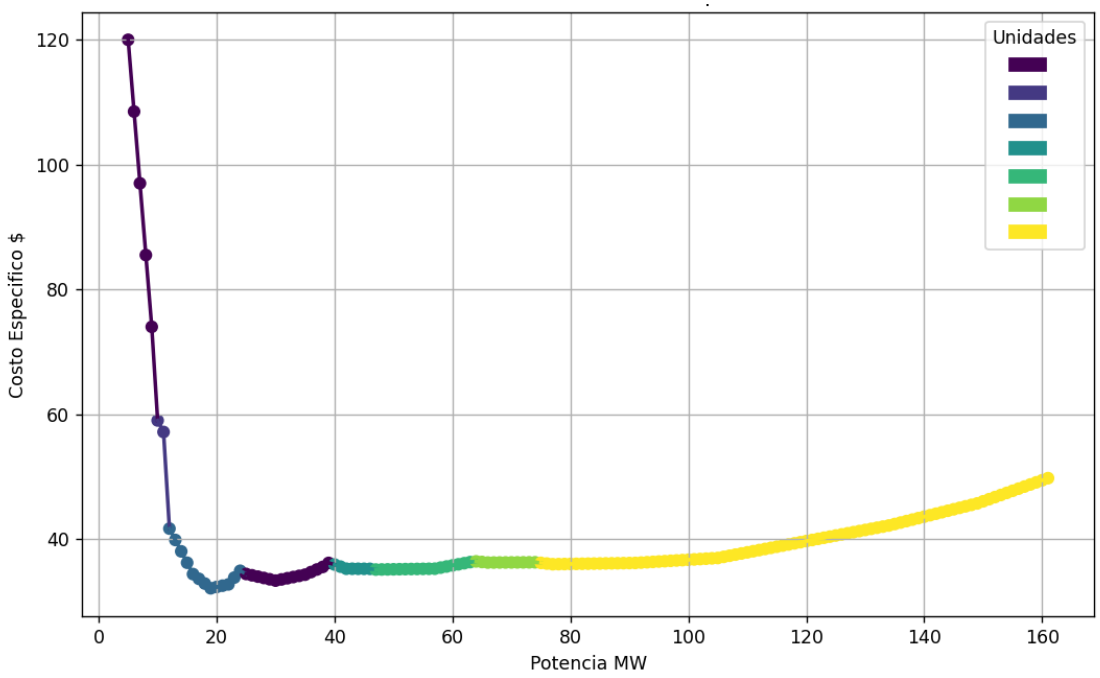

Figura 4.2. Valores mínimos del Costo específico-Python para 3 unidades

De la figura 4.2 se puede notar que hay 7 combinaciones que entran en funcionamiento, las cuales son representadas por colores. Sin embargo, hay casos notorios donde funcionan en rangos muy acotados, también los costos asociados no varían mucho hasta recién entrando en potencias elevadas.

Haciendo una comparación de figuras, se puede observar en la figura 4.2 y 4.3 cómo los valores mínimos calzan, corroborando que el algoritmo está seleccionando los valores óptimos.

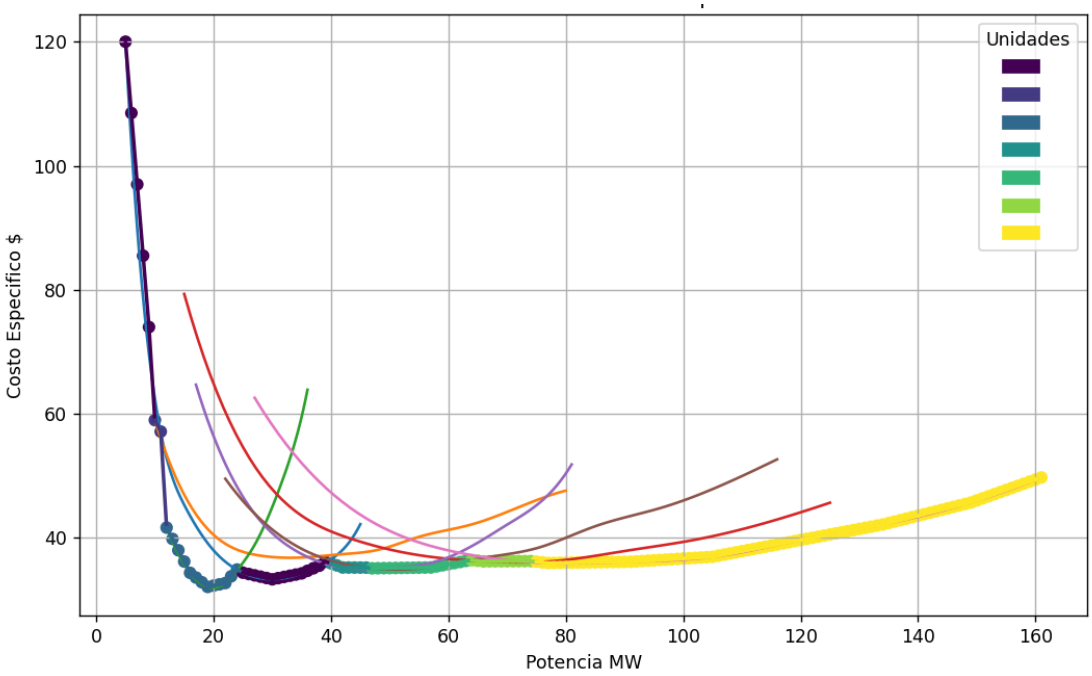

Figura 4.3. Valores mínimos Cmed-Python vis a vis las 7 curvas originales

Como una aplicación extra del algoritmo se tiene al detalle donde corresponde cada MW de las curvas, lo que ayuda a identificar los intervalos de potencia de cada unidad y donde se produce el cambio para cada una de estas.

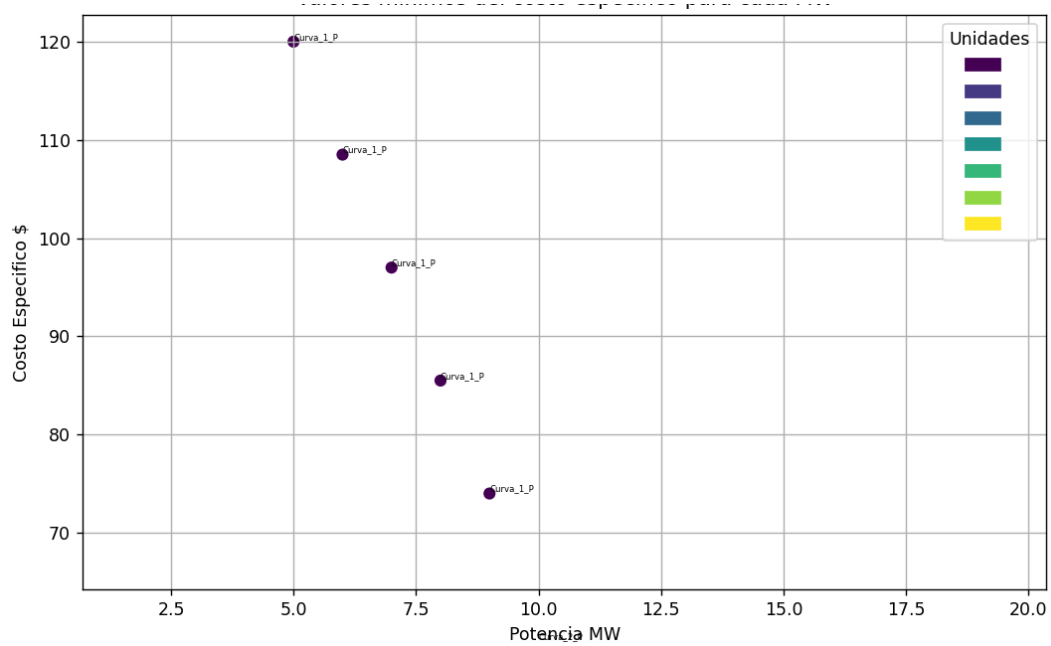

Figura 4.4. MW al detalle para curva 1

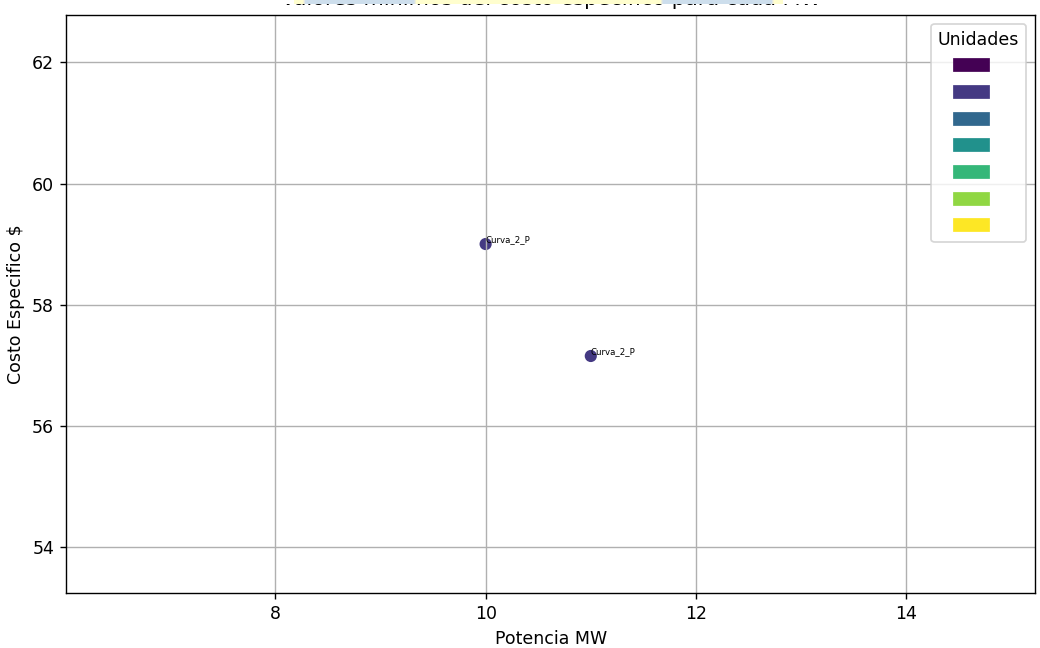

Figura 4.5. MW al detalle para curva 2

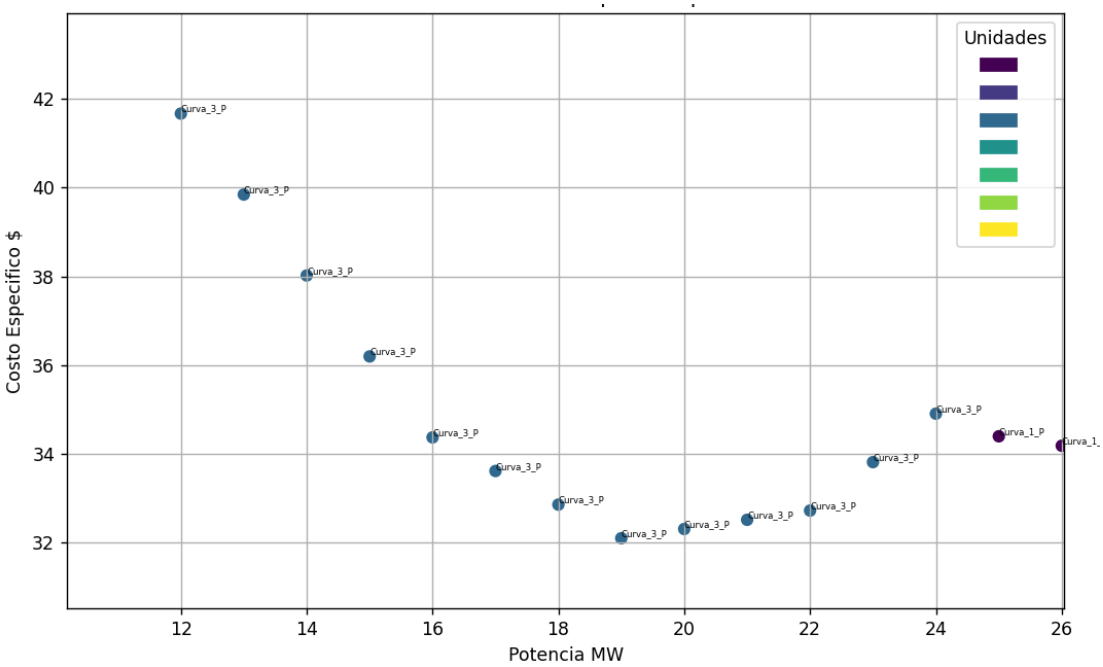

Figura 4.6. MW al detalle para curva 3

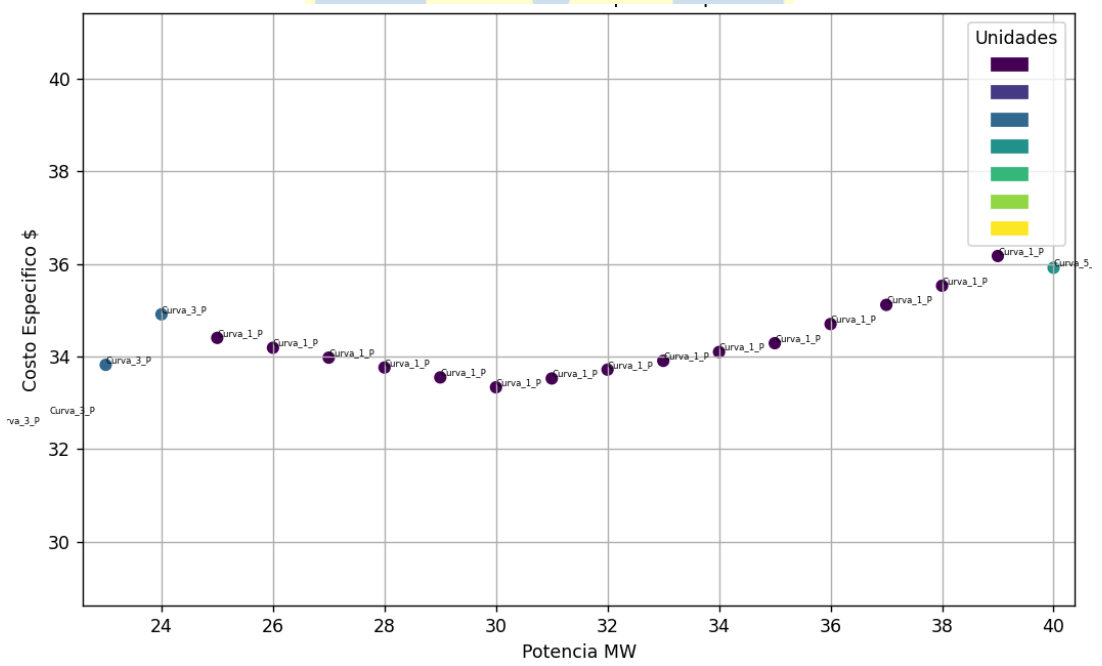

Figura 4.7. Segundo intervalo de MW al detalle para curva 1

Siguiendo las ideas anteriores se puede analizar y observar minuciosamente cada MW de salida correspondiente a las curvas, por lo tanto, los intervalos en los cuales se tiene la unidad generadora óptima se pueden identificar a simple vista. Sin embargo, siguiendo el proceso de automatización se procede a que el código también realice esta parte evitando la formulación “manual” de dichos intervalos. Estos son visibles en la consola de Python y en un archivo Excel llamado “intervalos_potencia” a continuación se puede ver la tabulación correspondiente.

Tabla 4.1. Intervalos de potencia de las unidades para el caso 1

Curvas de trabajo	Intervalo Inferior MW	Intervalo superior MW
Curva 1	5	9
Curva 2	10	11
Curva 3	12	24
Curva 1	25	39
Curva 5	40	46
Curva 6	47	63
Curva 4	64	74
Curva 7	75	161

Las curvas están ordenadas de menor a mayor según la potencia de salida al igual que las unidades generadoras representadas en la figura 4.1 donde se podían ver todas las combinaciones. El número de curva corresponde al orden de unidades visto en las leyendas de la figura 4.1, para representar lo dicho anteriormente se muestra el siguiente cuadro resumen a modo de ejemplo.

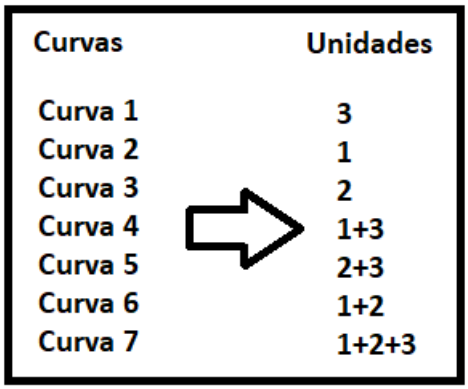

Figura 4.8. Correspondencia entre curvas y unidades caso 1

Sin embargo, para futuros casos donde la cantidad de curvas sea difícil de identificar a simple vista, el algoritmo posee un apartado donde guarda en un archivo Excel llamado “nombres_unidadesycurvas” los nombres de las curvas y las unidades correspondientes a estas.

Finalmente se puede observar cómo este algoritmo simplifica el análisis de datos para el PDE de unidades al automatizar la metodología y reducir los datos innecesarios en pantalla, además se posee una mayor exactitud a la hora de verificar los intervalos entre las curvas ya que el mismo programa puede tabular y mostrar en pantalla dichos intervalos de potencia lo que ayuda a tener una idea general del rango en el cual trabajan las unidades

4.2 Aplicación 2. Cuatro unidades generadoras

Para este caso, al agregar una nueva unidad al caso anterior, se tienen 15 combinaciones posibles. Por lo tanto, de acuerdo con el algoritmo propuesto, se agrega esta unidad generadora 4 (véase tabla 3.2) como un par de columnas en Excel desde donde el programa fuente en Python recopila los datos. Como en el caso anterior se aplica la misma algorítmica obteniendo los siguientes resultados.

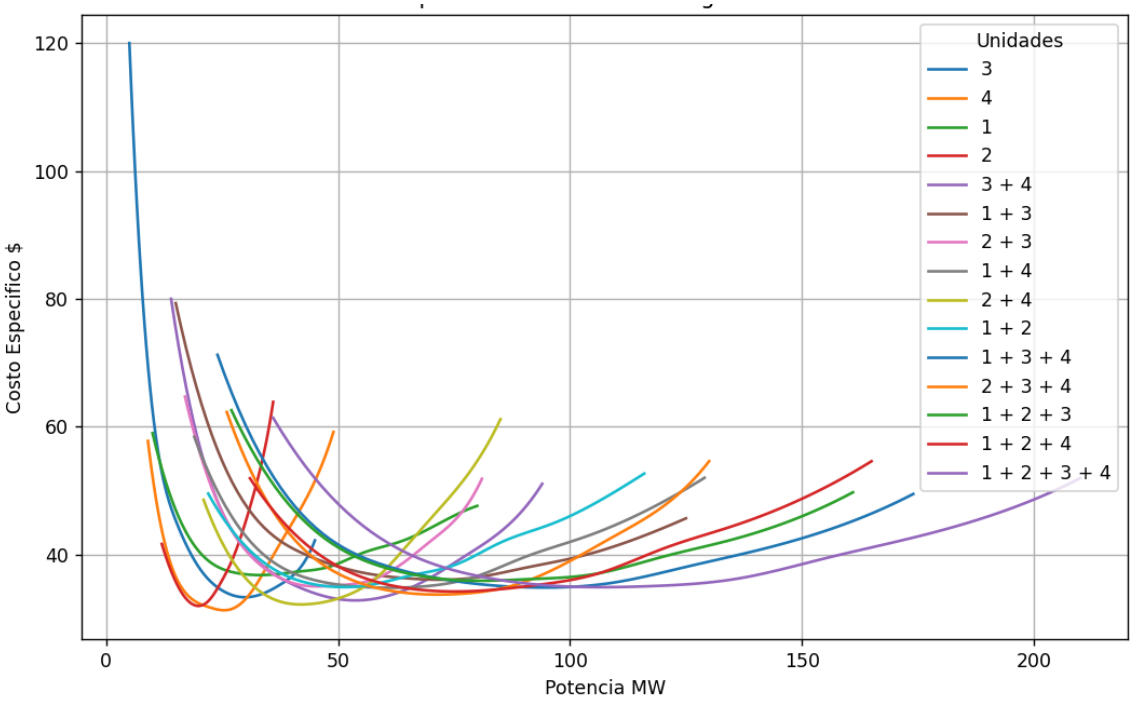

Figura 4.9. Unidades generadoras y combinaciones para 4 unidades

De la figura 4.9 se puede verificar de las 15 combinaciones presentes, que la información en pantalla y en particular, los puntos mínimos se vuelven difíciles de observar. En consecuencia, siguiendo las mismas ideas ya establecidas previamente, se procede a la siguiente parte del logicial.

Observando la figura 4.10, se puede notar que hay 9 combinaciones representadas por distintos colores. Esto significa que de las 15 combinaciones posibles solo 9 de estas deben asociarse. Para verificar dicha información se compara este resultado con aquellos de la curva original.

Observando la figura 4.11 se corrobora que hay combinaciones que no participan de PDENL del parque en ningún momento. Por lo tanto, se guardan solo las combinaciones de unidades comprometidas por el algoritmo.

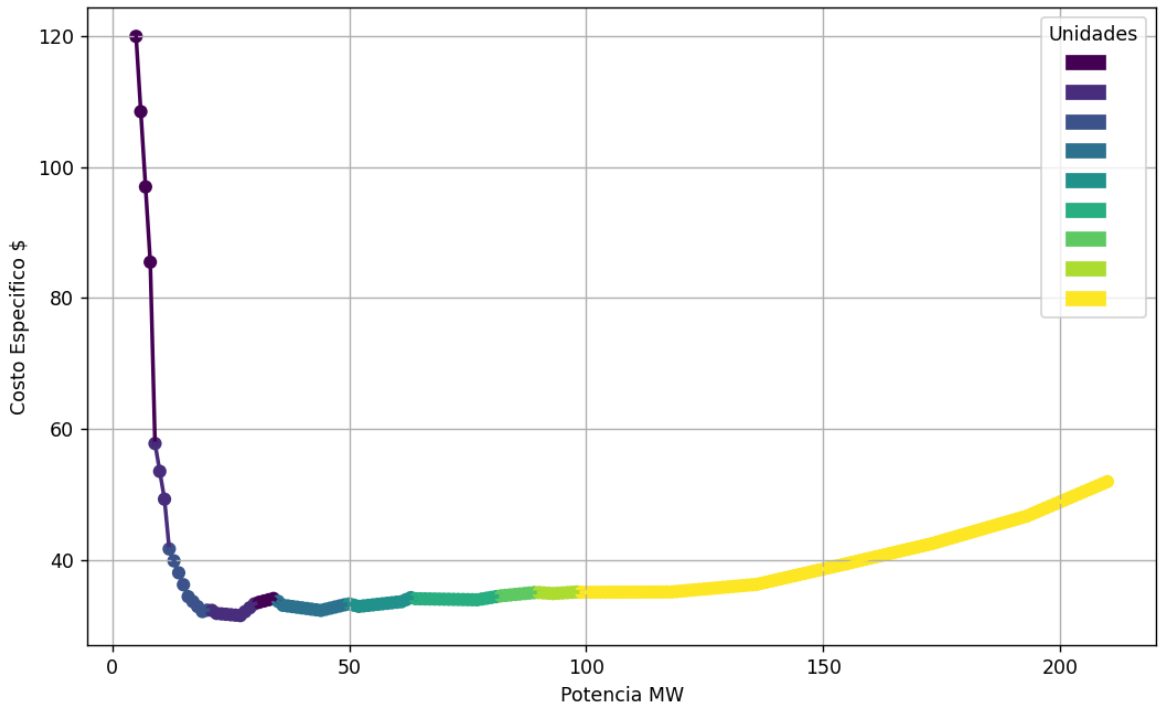

Figura 4.10. Valores mínimos del Costo específico-Python para 4 unidades

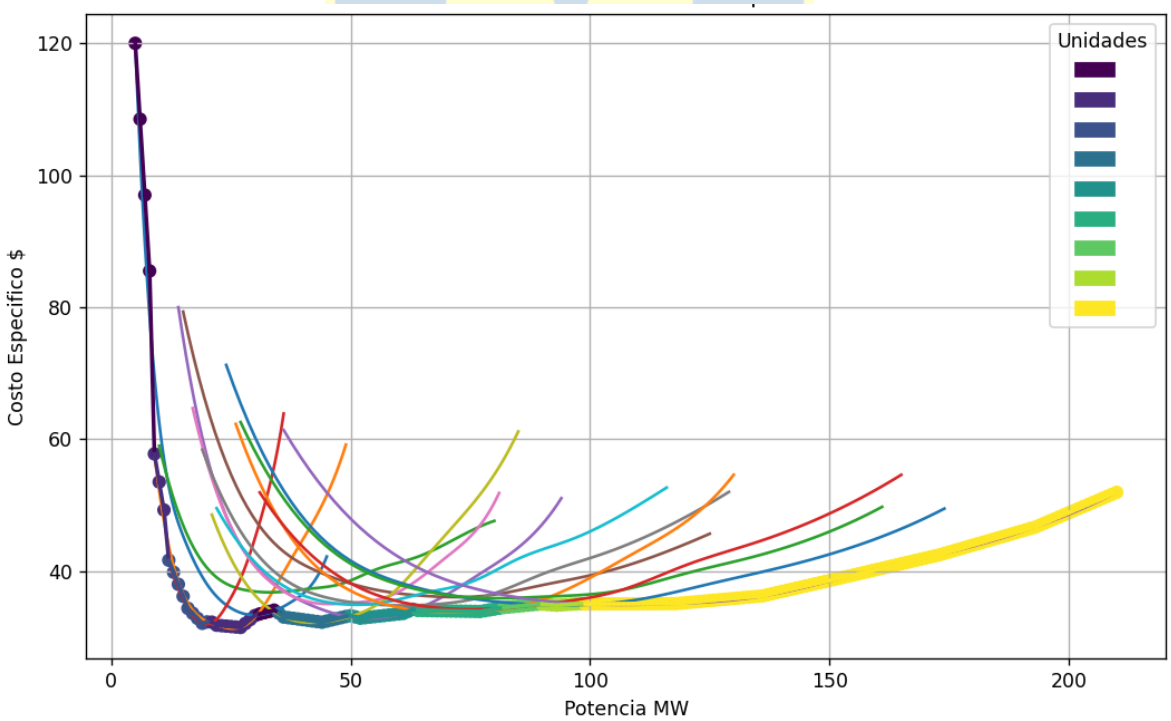

Figura 4.11. Valores mínimos Cmed-Python vis a vis las 15 curvas originales

Tabla 4.2. Intervalos de potencia de las unidades para el caso 2

Curvas de trabajo	Intervalo inferior MW	Intervalo superior MW
Curva 1	5	8
Curva 2	9	11
Curva 4	12	20
Curva 2	21	30
Curva 1	31	34
Curva 9	35	49
Curva 5	50	63
Curva 12	64	81
Curva 14	82	89
Curva 11	90	98
Curva 15	99	210

Tabla 4.3. Correspondencia entre unidades y curvas caso 2

Unidades	Curvas
3	Curva 1
4	Curva 2
2	Curva 4
3 + 4	Curva 5
2 + 4	Curva 9
1 + 3 + 4	Curva 11
2 + 3 + 4	Curva 12
1 + 2 + 4	Curva 14
1 + 2 + 3 + 4	Curva 15

.

Igual que en el caso 1, vemos cómo el universo de posibilidades se reduce significativamente lo cual ayuda muy eficientemente a analizar y simplificar tanto el problema como el esfuerzo computacional (tiempo de CPU y almacenamiento de datos). En este escenario de cuatro unidades, al agregar una unidad se puede evidenciar como los datos aumentan y el problema se “ensucia”. Como se vio reflejado en la figura 4.9 (el uso de todas las combinaciones) lo vuelve difícil desde cualquier punto de vista. Por tanto, se hace evidente que la automatización

está cumpliendo muy eficientemente, el objetivo de optimizar la algorítmica del tratamiento de las unidades para un mejor análisis de datos de PDE.

4.3 Aplicación 3. Diez unidades generadoras

Para este caso se agregaron 6 unidades, de costo de fuel no lineales, extras a modo de prueba. Como en los casos anteriores seguimos la misma metodología la cual ya fue vista en el inciso anterior.

Las combinaciones posibles crecen hasta 1023, lo que genera cierto tiempo de espera en el programa al ejecutarlo debido a la gran cantidad de información procesada, la cual se puede ver reflejada en las curvas originales antes de la optimización. Esto se observa en la siguiente figura.

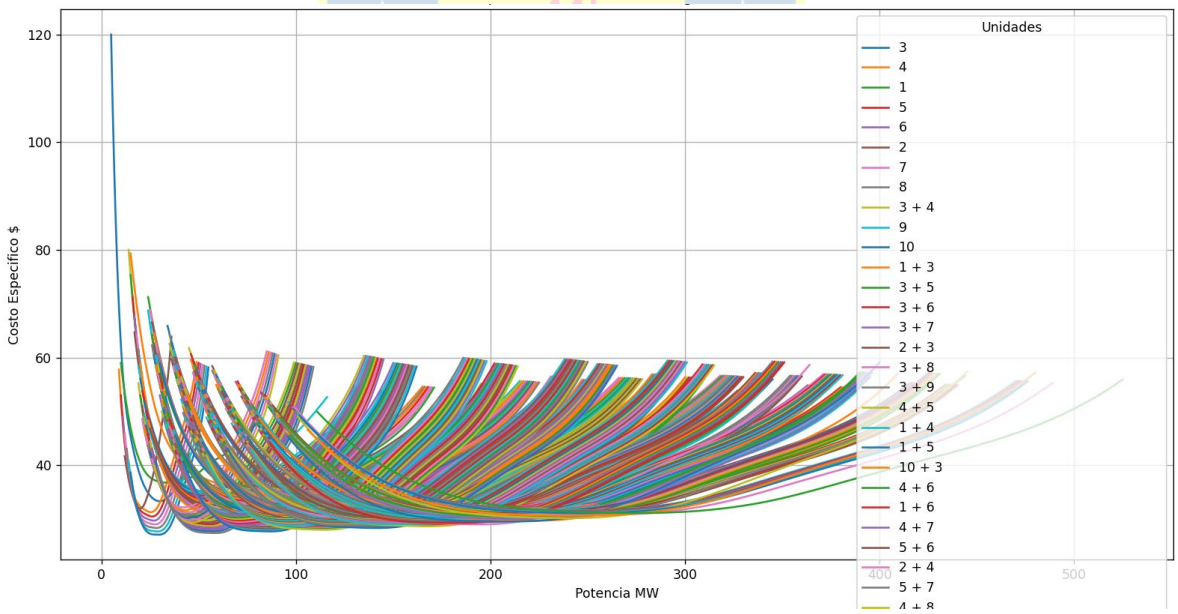

Figura 4.12. Unidades generadoras y combinaciones para 10 unidades

Podemos evidenciar claramente la problemática surgida al no optimizar el proceso de búsqueda de las soluciones. La figura 4.12 nos muestra cómo se ven 10 unidades y sus combinaciones al no ser tratadas por el algoritmo, el análisis de datos se vuelve inviable para este caso.

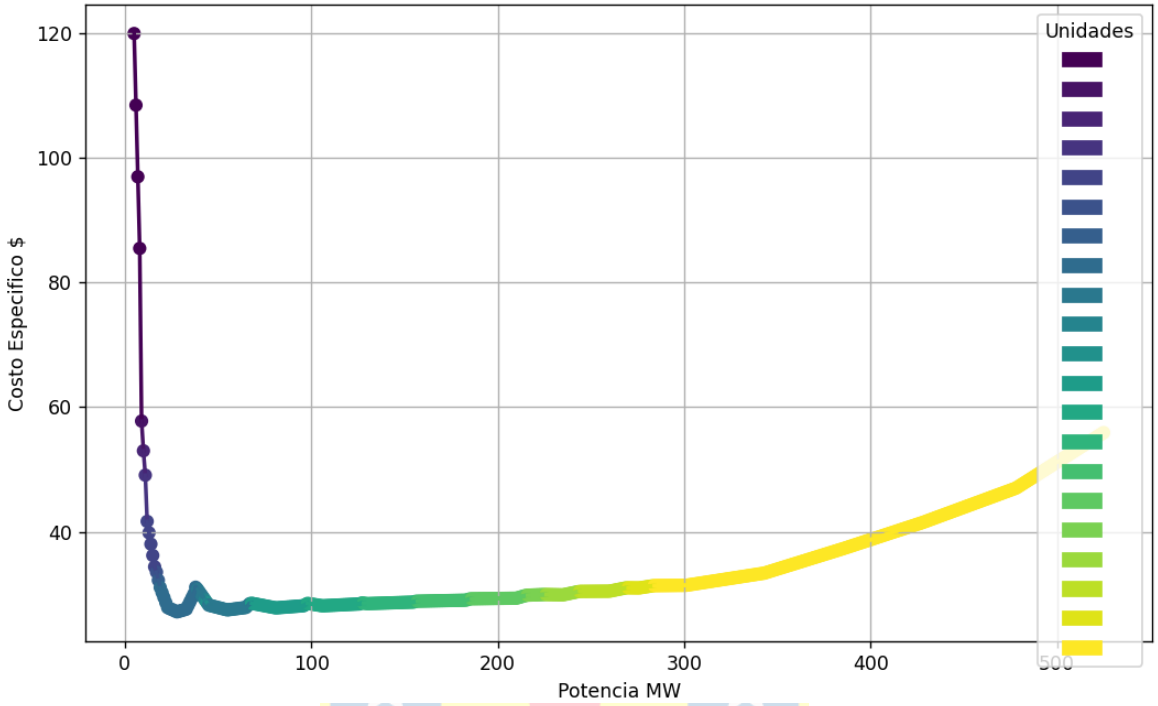

Figura 4.13. Valores mínimos del Costo específico-Python para 10 unidades

En este gráfico de la figura 4.13, podemos observar cómo lucen los datos al ser optimizados por el algoritmo, la paleta de colores, como en los casos anteriores, muestra cuando entra en funcionamiento cada unidad. Se puede concluir que de las 1023 combinaciones solo algunas de estas entran a trabajar en el sistema eléctrico como era de esperar. Además, la abrumadora cantidad de datos se vio mermada casi en su totalidad para los análisis y conclusiones pertinentes. Por otro lado, complementando con la automatización realizada para los intervalos de potencia, se sabe a partir de esta que solo 21 combinaciones deben ser consideradas (ver figura que sigue).

Tabla 4.4. Intervalos de potencia de las unidades para el caso 3

Curvas de trabajo	Intervalo Inferior MW	Intervalo superior MW
Curva 1	5	8
Curva 2	9	9
Curva 4	10	10
Curva 5	11	11

Curva 6	12	16
Curva 8	17	17
Curva 10	18	18
Curva 11	19	38
Curva 78	39	42
Curva 64	43	43
Curva 71	44	44
Curva 78	45	67
Curva 241	68	97
Curva 479	98	127
Curva 721	128	156
Curva 879	157	185
Curva 957	186	215
Curva 1011	216	225
Curva 992	226	243
Curva 1019	244	269
Curva 1017	270	283
Curva 1023	284	525

Con la tabulación al detalle conocemos en su totalidad las curvas y las ventanas asociadas al PDENL. Se tienen 21 combinaciones que entran en funcionamiento, algunas apenas de 1 MW lo cual las hace inviable colaborar con el PDE. Sin embargo, para la base investigativa del problema muestra el nivel de precisión que posee el algoritmo. Para ayudar con la identificación a que unidad pertenece cada curva, accedemos al apartado mencionado en el caso de tres unidades ítem 4.1 el cual guarda en un archivo Excel nombrado “nombres_unidadesycurvas” los respectivos nombres de cada curva, de aquí podemos identificar la unidad que entró en funcionamiento.

Por lo tanto, de la tabla 4.5 se tienen identificadas las curvas y unidades que entraron al proceso. De la tabla 4.4 se obtienen las ventanas de potencia de estas curvas. De la figura 4.13 se puede observar la optimización que se esperaba. En resumen, la automatización planteada y vista en los casos de tres y cuatro unidades está funcionando de igual manera para el caso de diez unidades, donde

las combinaciones y las unidades asociadas crecieron exponencialmente entrando en el terreno de la alta carga computacional.

Tabla 4.5. Correspondencia entre unidades y curvas caso 3

Unidades	Curvas
3	Curva 1
4	Curva 2
5	Curva 4
6	Curva 5
2	Curva 6
8	Curva 8
3+4	Curva 10
10	Curva 11
2+3+7	Curva 78
2+3+5	Curva 64
3+5+8	Curva 71
1+10+3+7	Curva 241
3+5+7+8+9	Curva 479
1+2+4+5+6+8	Curva 721
1+10+3+4+5+7+9	Curva 879
1+10+2+3+4+5+6+7	Curva 957
10+2+4+5+6+7+8+9	Curva 1011
1+10+2+3+4+6+8+9	Curva 992
10+2+3+4+5+6+7+8+9	Curva 1019
1+10+2+3+4+5+6+8+9	Curva 1017
1+10+2+3+4+5+6+7+8+9	Curva 1023

4.4 Aplicación 4. Indisponibilidad de unidades

Como bien se sabe, no es raro que algunas de las unidades generadoras se encuentren indisponibles por distintos motivos. Por lo tanto, para observar el comportamiento de las distintas unidades, se añade este apartado para tratar dicho aspecto.

El código en este apartado requiere del *ingreso de las unidades a indisponer*. Por este camino, las unidades declaradas indisponibles no son consideradas por el algoritmo, esta consideración (como se espera) reduce, en particular, el tiempo

de CPU. Luego de seleccionar la(s) unidad(es) a indisponer mediante, el código sigue el paso a paso descrito en los ítems anteriores. A modo de ejemplificar y explicar este escenario se usan los datos de los casos declarados previamente. En particular, el caso a tratar consiste en aquel de 10 unidades generadoras que llega a tener 1023 combinaciones o estados posibles.

Caso 1. Como se puede observar en la figura 4.14 se indispusieron dos unidades. Estas corresponden a las unidades 1 y 3. Bajo esta premisa se ensayan distintas "semillas" de indisponibilidades con el fin de revisar tanto el aspecto práctico como la robustez del modelo.

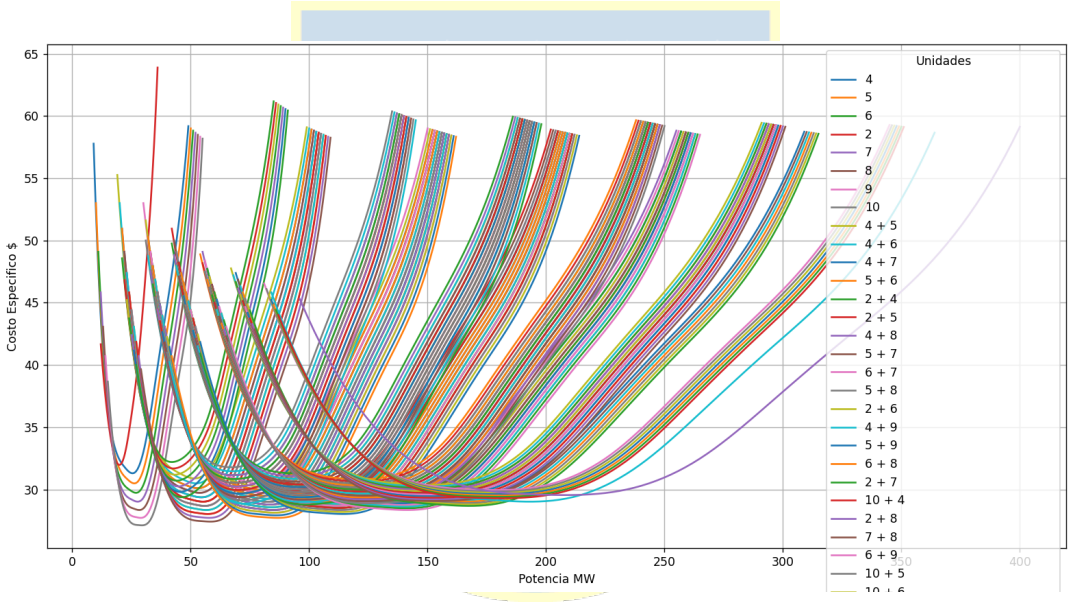

Figura 4.14. Combinaciones de 10 unidades con las unidades 1 y 3 indisponibles

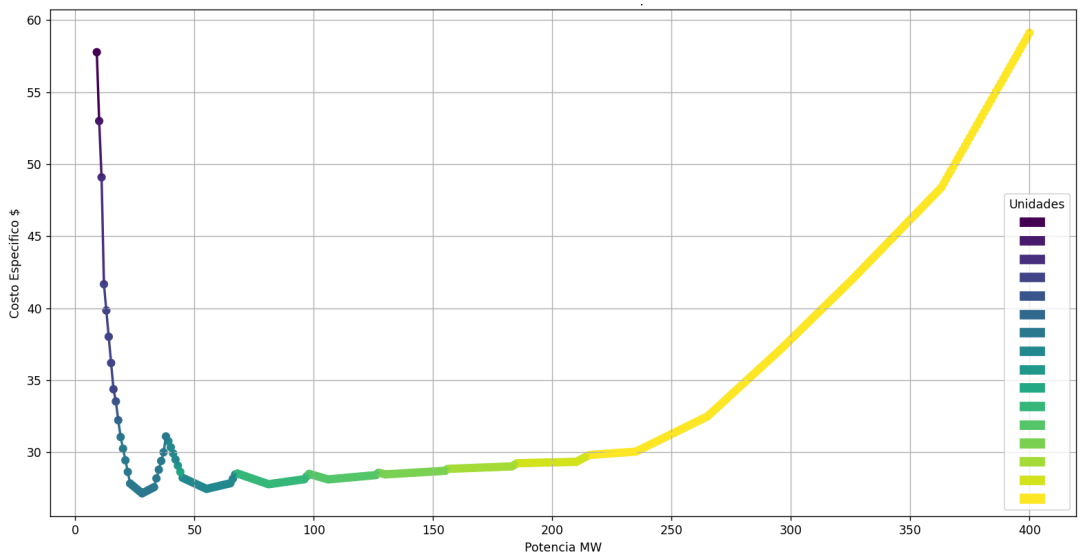

Figura 4.15. Optimización de 10 unidades con las unidades 1 y 3 indisponibles

Tabla 4.6. Intervalos óptimos de PDENL del parque de 10 unidades con las unidades 1 y 3 indisponibles

Curvas de trabajo	Intervalo Inferior MW	Intervalo superior MW
Curva 1	9	9
Curva 2	10	10
Curva 3	11	11
Curva 4	12	16
Curva 6	17	17
Curva 7	18	18
Curva 8	19	38
Curva 36	39	42
Curva 32	43	43
Curva 35	44	44
Curva 36	45	67
Curva 92	68	97
Curva 163	98	127
Curva 217	128	156
Curva 242	157	185
Curva 250	186	215
Curva 255	216	400

Tabla 4.7. Correspondencia entre unidades y curvas para 10 unidades con las unidades 1 y 3 indisponibles.

Unidades	Curvas
4	Curva 1
5	Curva 2
6	Curva 3
2	Curva 4
8	Curva 6
9	Curva 7
10	Curva 8
10 + 7	Curva 32
10 + 8	Curva 35
10 + 9	Curva 36
10 + 8 + 9	Curva 92
10 + 7 + 8 + 9	Curva 163
10 + 6 + 7 + 8 + 9	Curva 217
10 + 5 + 6 + 7 + 8 + 9	Curva 242
10 + 4 + 5 + 6 + 7 + 8 + 9	Curva 250
10 + 2 + 4 + 5 + 6 + 7 + 8 + 9	Curva 255

La automatización como era de esperar funciona correctamente. Las unidades disponibles son optimizadas (ver figura 4.15) y su respectivo análisis es detallado en las tablas 4.6 y 4.7. A partir de lo anterior se muestra como los costos medios se disparan a medida que la potencia de salida supera el umbral de los 250 MW. Este ensayo deja en evidencia que las unidades 1 y 3, estabilizan y mantienen bajos los costos medios. Además, como se vio en las aplicaciones anteriores, solo 17 combinaciones están asociadas y “entraron” a trabajar. En definitiva, este primer caso, corrobora que el código trabaja sin problemas, resguardando las condiciones de optimización y entrega las tablas complementarias para el análisis del problema. Por lo tanto, para los siguientes casos se omitirán las tablas y solo se enfocará en constatar, gráficamente, cómo va variando el resultado óptimo al indisponer distintos grupos de unidades.

Caso 2. Para el caso representado en las figuras 4.16 y 4.17 se puede observar cierta similitud con lo mencionado anteriormente. Al tener indisponibles las

unidades impares (1, 3, 5, 7, 9) se observa que los costos medios se escapan a partir de cierta potencia de salida. En este caso, a diferencia del anterior, el umbral donde se disparan los costos se encuentra aproximadamente en los 140 MW.

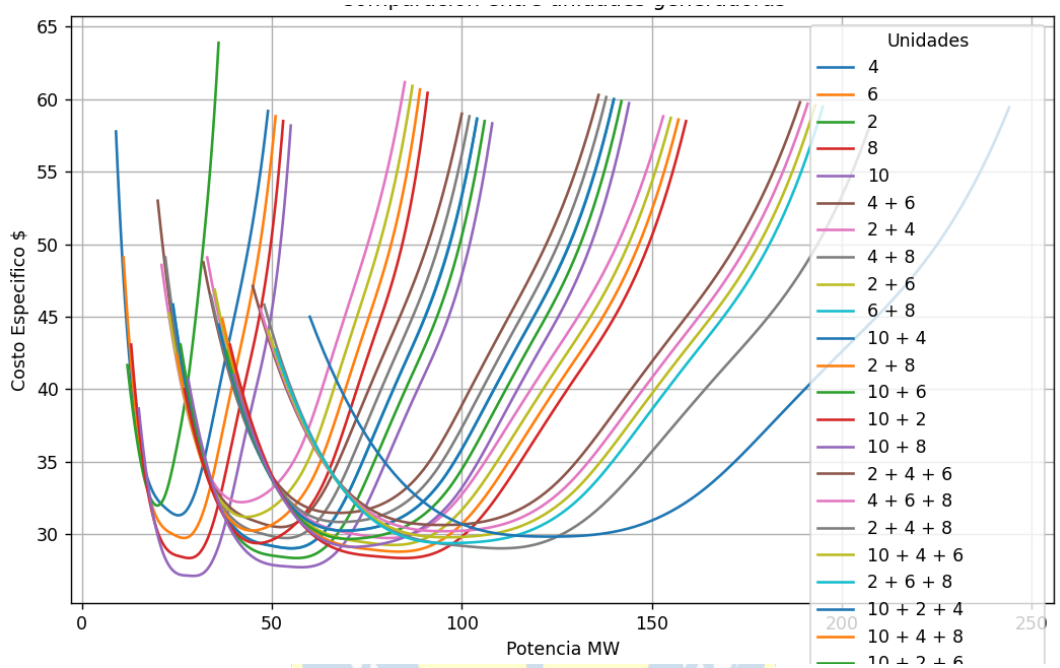

Figura 4.16. Combinaciones de 10 unidades con las unidades 1, 3, 5, 7, 9 indisponibles

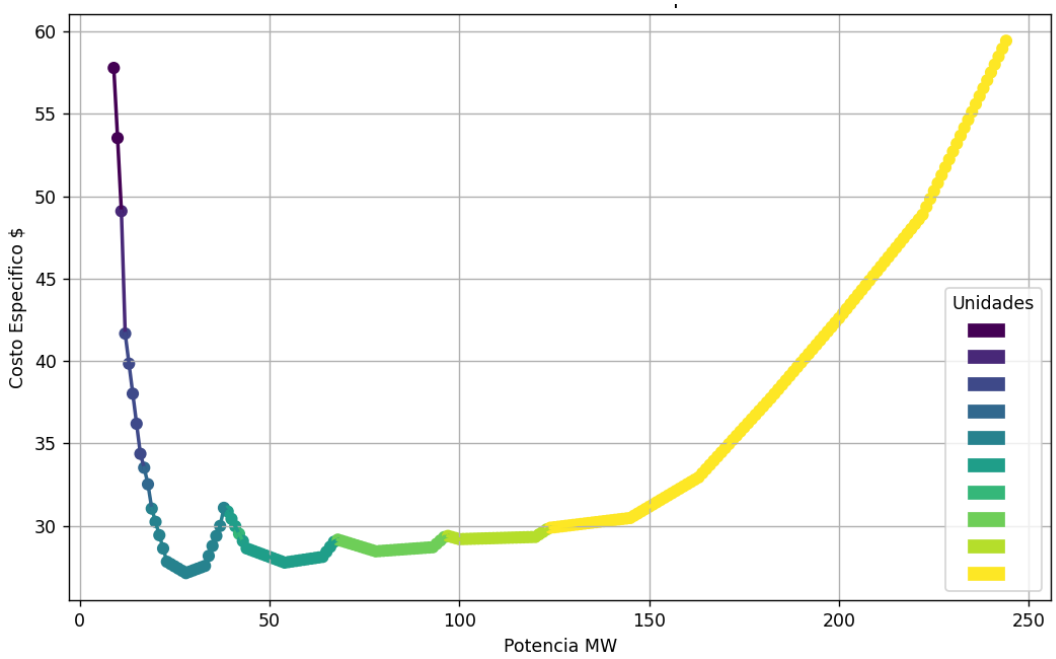

Figura 4.17. Optimización de 10 unidades con las unidades 1, 3, 5, 7, 9 indisponibles

Caso 3. En este último caso visto en las figuras 4.18 y 4.19 el patrón se repite, la indisponibilidad de las unidades generadoras 3, 5 y 8 provocó un aumento brusco de los costos medios al ir aumentando la potencia, esto sucede a partir de los 230 MW. A través de esos casos se puede ver que la indisponibilidad de unidades no afecta el desempeño del código, i.e. La optimización sigue entregando las mejores curvas de rendimiento independiente de las unidades que se hallen indisponibles.

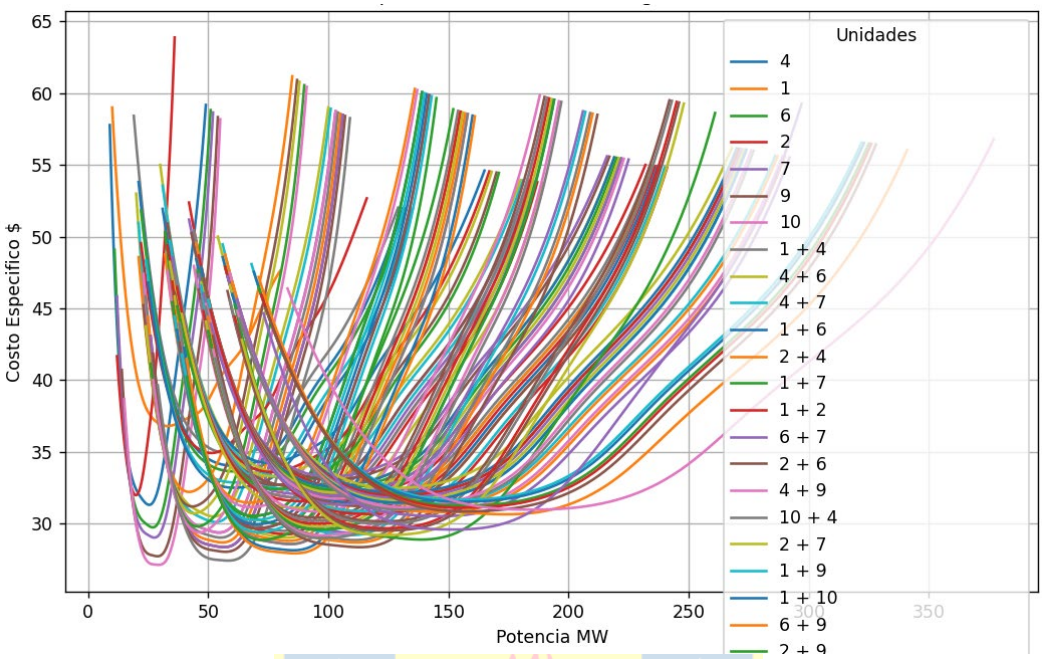

Figura 4.18. Combinaciones de 10 unidades con las unidades 3, 5, 8 indisponibles

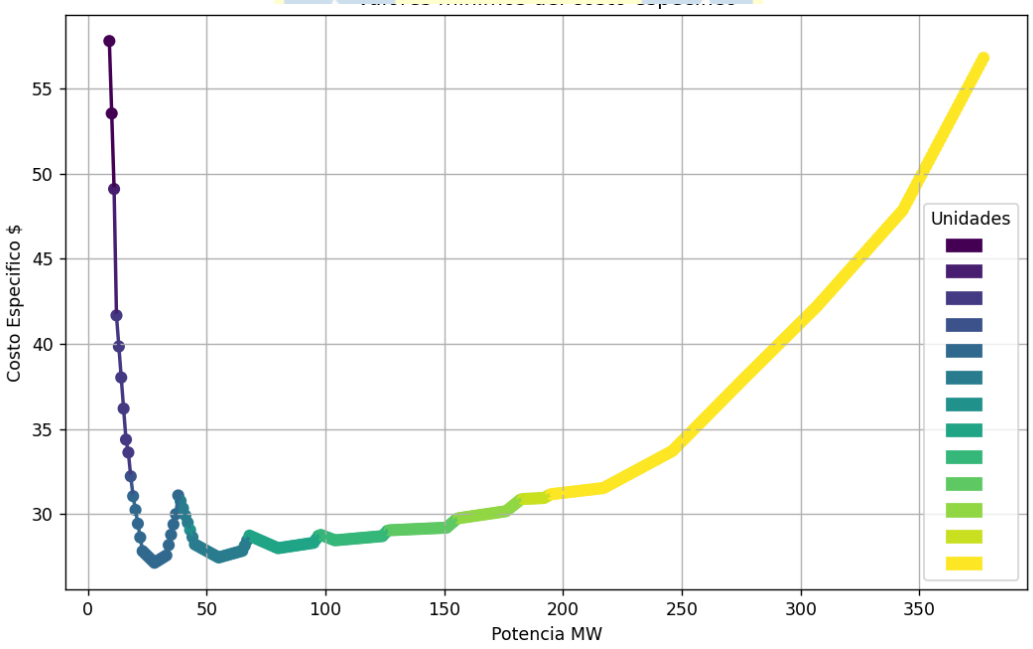

Figura 4.19. Optimización de 10 unidades con las unidades 3, 5, 8 indisponibles

En resumen, se puede decir que la automatización funciona correctamente en aplicaciones con aleatoriedades. En general y a manera de conclusión, la optimización arrojada para los casos con distintos grupos de unidades indisponibles siempre representa la mejor opción de PDENL.

4.5 Esfuerzo computacional

Retomando lo que se vio en las primeras aplicaciones, se puede observar que el tiempo de respuesta del algoritmo empezó a incrementar proporcionalmente con la cantidad de unidades a tratar. Para la aplicación 1 y 2 que consta de 3 unidades y 4 unidades, respectivamente, el tiempo fue bastante rápido, i.e. en el orden de los segundos. Posterior a esto la aplicación 3 que consta de 10 unidades el tiempo de respuesta superó el minuto. Este resultado muestra, que la plataforma (Código/Python) está completamente ligada al proceso combinatorio por el cual debe pasar el algoritmo. En los primeros casos se procesan apenas 7 y 15 combinaciones, mientras que el último alcanza las 1023 combinaciones.

El creciente tiempo CPU (segundos a minutos) catapultó un análisis ad-hoc. Por lo tanto, se investigó el tiempo y esfuerzo requerido para efectuar la optimización de distintas unidades donde las combinaciones vayan *in crescendo*. Para lograr este cometido se agregó una línea de medición de tiempo en la ejecución del código.

En primera instancia, como se puede ver en la siguiente tabla, se midió el tiempo de ejecución para distintos escenarios donde la labor combinatoria cada vez es más extensa. Se procedió desde la más básica que consiste en 2 unidades y 3 combinaciones, hasta la más compleja que consta de 10 unidades y 1023 combinaciones, respectivamente (véase tabla 2.1).

Tabla 4.8. Tiempo de ejecución para distintas unidades.

Unidades	Tiempo (s)
1, 2	0.93
1, 2, 3	1.19
1, 2, 3, 4	1.65
1, 2, 3, 4, 5	2.64
1, 2, 3, 4, 5, 6	5.05
1, 2, 3, 4, 5, 6, 7	10.21
1, 2, 3, 4, 5, 6, 7, 8	22.41
1, 2, 3, 4, 5, 6, 7, 8, 9	47.70
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	109.97

Se pudo reconocer que en la medida que el tiempo de ejecución del código fue aumentando paulatinamente, la carga combinatoria que esta conlleva también fue creciendo. En cifras globales el tiempo de CPU fue creciendo casi al doble en cada ocasión. En breve, para estas primeras unidades vistas en la tabla 4.8 las combinaciones son pocas, por lo que la ejecución del código sigue siendo rápido manteniéndose en el orden de segundos sin superar el minuto. Sin embargo, para 10 unidades entramos en el terreno de los miles de combinaciones lo que conlleva un cambio en el tiempo de ejecución pasando al orden de los minutos. Por lo tanto, recordándose que se tiene como limitante el hardware y el ambiente, que para este caso es Python, se indagará hasta donde es posible efectuar una automatización y posterior optimización sin entrar en periodos extensos de espera. Cabe recalcar que para dicho cometido se omitirá el análisis y gráfica de las unidades en pos de enfocarse puramente en la labor combinatoria y el esfuerzo computacional requerido para dichos casos que se componen de miles.

Tabla 4.9. Tiempo de ejecución para más de 10 unidades.

Unidades	Tiempo (s)
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	243.21
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	614.15
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	1488.01
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	3677.23
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	10193.22

De la tabla 4.9 se puede observar como el tiempo de ejecución del código crece exponencialmente. En este apartado se puede identificar esta nueva problemática surgida a raíz de optimizar más de 10 unidades. Al agregar una nueva unidad las combinaciones crecen enormemente, siendo 32767 combinaciones posibles para 15 unidades, lo que sugiere que para fines prácticos el límite se establece en 14 unidades ya que más allá de eso, el tiempo de ejecución y el esfuerzo computacional requerido se vuelve muy extenso de optimizar. Por lo tanto, mientras se tengan de limitantes los aspectos técnicos mencionados en el ítem 1.4 *"alcances y límites"* los tiempos de procesamientos digital variarán en gran medida, según los casos evaluados y registrados en las tablas.

Capítulo 5: Conclusiones

Como se vio reflejado anteriormente en los casos y sus posteriores aplicaciones se sabe que la estrategia de optimización que supera los problemas combinatorios de la PDENL funciona bien. Sin embargo, esta acarrea nuevos problemas, los cuales son para tomar en cuenta. Por ejemplo, lo relacionado al esfuerzo computacional empieza a crecer exponencialmente lo que implica una limitante al procesar casos con un número significativo de unidades de producción. Por lo tanto, una completa y minuciosa revisión del código, junto a una reinterpretación de algunos procesos, podría llevar a una disminución de los tiempos de CPU y posiblemente a esfuerzos de almacenamiento más pequeños.

La elección de utilizar el programa "Python" fue en primera instancia debido a su sencillez y enfoque minimalista al programar, i.e la sintaxis de Python es fácilmente entendible y legible, lo que reduce en gran medida la cantidad de código necesario. Por otro lado, la versatilidad que brinda deja abierta las puertas para seguir explorando soluciones y herramientas para la optimización. No obstante, también posee desventajas, ya que al ser un lenguaje interpretado es

bastante más lento. Claramente, no puede competir con lenguajes compilados como C++ o Fortran [5]. Esto se vio reflejado en la problemática del ítem 4.5, donde el tiempo de ejecución del código empezó a incrementar enormemente. Por lo tanto, en vez de una reinterpretación de procesos u optimización del algoritmo, un cambio de lenguaje a C++, C o Java, sería beneficioso para el problema. La compilación permite trabajar nivel de código de máquina, lo que implica ejecuciones más performantes para la labor combinatoria en la que incurre la optimización de unidades.

Por otro lado, se sabe que la base de datos en Excel puede llegar a ser el eje principal de la automatización y ejecución del código, en un futuro no muy lejano, debido a que actualmente Microsoft se encuentra en una fase beta de integración entre Python y Excel [6]. Por lo tanto, al unificarse dichos ambientes se potencializa más eficiencia computacional, al no cruzar datos entre distintos programas “no emparentados”.

En otro aspecto de la automatización se vio como la optimización lleva, muy eficientemente, a las curvas de mejor rendimiento, independiente de su intervalo de trabajo, al centrarse en los costos de la energía primaria y evitando otros costos tales como el de partida y/o de parada de generadores. Lo previo, implica que se centre en un enfoque puramente de eficiencia expresada en las funciones entrada/salida de cada unidad. Esta hipótesis puede ser levantada, sin grandes dificultades, en investigaciones posteriores. Evidentemente, al considerar dichos costos que fueron ignorados en esta investigación, si bien aumentará la complejidad de la investigación y la búsqueda de soluciones futuras, llevará también a que los modelos se potencien y superen situaciones reales actuales y próximas. En contrapartida, el modelo propuesto suma y resuelve el tratamiento de las incertidumbres reales presentes en control de un parque de generación.

Finalmente, al trabajar con unidades generadoras representadas mediante funciones cuadráticas nos da una mirada más amplia y cercana del rendimiento

que pueden alcanzar dichas fuentes. Debido a que las formas lineales que se presentan en la literatura ya son reconocidas como una aproximación de las unidades generadoras. Lo anterior puede vislumbrarse cualitativamente, puesto que al reconocer combinatorias cada vez de mayor orden hace que los resultados se acerquen en gran medida a los valores reales. En otras palabras, al aumentar el grado en la caracterización de las funciones de costo y abandonar la linealidad del análisis, se consigue un enfoque más cercano al actual de sus operaciones. Consecuentemente, ayuda a la planeación y definir de mejor manera las unidades que pueden, aleatoriamente, rendir mejor en un intervalo específico.

Bibliografía

[1] Córdova Sota, Samuel Alejandro. (2017). *Un esquema eficiente de pronóstico-optimización para el predespacho intra-diario bajo alta penetración eólica y solar (pp 55-58)* [Memoria de título, Pontificia Universidad Católica de Chile]. Repositorio institucional – Pontificia Universidad Católica de Chile.

[2] Duarte Quiñonez, Alcaraz Varela, Arce Encina. (2022). *Modelo de Pre-Despacho para un Sistema Eléctrico de Potencia* [Articulo científico, Facultad Politécnica UNE]. Repositorio institucional – FPUNE Scientific.

[3] Levieux, Ocampo Martínez, Inthamoussou, De Battista. (7-9 de noviembre 2018). *Power schedule for a hydro-wind system based on a predictive algorithm* [Discurso principal]. Conferencia en control automático, Buenos Aires, Argentina. Repositorio internacional – IEEE Xplore.

[4] Grainer, Stevenson Jr. (1996). Operación económica de sistemas de potencia, *Análisis de sistemas de potencia* (1ra edición, pp 499-503). McGRAW-HILL.

[5] freeCodeCamp.org. (2022, 23 de septiembre). Lenguajes de programación interpretados vs compilados: ¿Cuál es la diferencia?. *freeCodeCamp página web.* https://www.freecodecamp.org/espanol/news/lenguajes-compilados-vs-interpretados/

[6] Palacios, José. (2023, 23 de agosto). Python en Excel: una nueva funcionalidad que revoluciona el análisis y la visualización de datos. *Microsofters página web.* https://microsofters.com/188785/python-en-excel-una-nueva-funcionalidad-que-revoluciona-el-analisis-y-la-visualizacion-de-datos/

Anexo

A. Base de datos en Excel

A	B	C	D	E	F	G	H	I	J
Unidad 1	Unidad 1	Unidad 2	Unidad 2	Unidad 3	Unidad 3	Unidad 4	Unidad 4	Unidad 5	Unidad 5
10	590	12	500	5	600	9	520	10	530
20	810	16	550	10	625	13	530	14	540
28	1040	19	610	15	680	17	580	18	590
34,75	1280	22	720	20	760	22	700	23	710
41,25	1540	25	900	25	860	27	850	28	860
48,25	1840	27	1050	30	1000	31	1050	32	1060
55	2210	30	1350	35	1200	35	1350	36	1360
64	2700	32	1600	38	1350	39	1700	40	1710
73	3310	34	1900	42	1600	44	2200	45	2210
80	3810	36	2300	45	1900	49	2900	50	2950

Unidad 6	Unidad 6	Unidad 7	Unidad 7	Unidad 8	Unidad 8	Unidad 9	Unidad 9	Unidad 10	Unidad 10
11	540	12	550	13	560	14	570	15	580
15	550	16	560	17	570	18	580	19	590
19	600	20	610	21	620	22	630	23	640
24	720	25	730	26	740	27	750	28	760
29	870	30	880	31	890	32	900	33	910
33	1070	34	1080	35	1090	36	1100	37	1110
37	1370	38	1380	39	1390	40	1400	41	1410
41	1720	42	1730	43	1740	44	1750	45	1760
46	2230	47	2240	48	2250	49	2260	50	2270
51	3000	52	3050	53	3100	54	3150	55	3200

Unidad 11	Unidad 11	Unidad 12	Unidad 12	Unidad 13	Unidad 13	Unidad 14	Unidad 14	Unidad 15	Unidad 15
16	590	17	600	18	610	19	620	20	630
20	600	21	610	22	620	23	630	24	640
24	650	25	660	26	670	27	680	28	690
29	770	30	780	31	790	32	800	33	810
34	920	35	930	36	940	37	950	38	960
38	1120	39	1130	40	1140	41	1150	42	1160
42	1420	43	1430	44	1440	45	1450	46	1460
46	1770	47	1780	48	1790	49	1800	50	1810
51	2280	52	2290	53	2300	54	2310	55	2320
56	3250	57	3300	58	3350	59	3400	60	3450

B. Bibliotecas usadas en Python.

```python
import sys
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import itertools
from scipy.interpolate import interp1d
from tabulate import tabulate
import time
```

C. Tablas de potencias generadas en excel.

Caso 3 unidades

numeros_x	nombres_curvas_minimas
5	Curva_1_P
9	Curva_1_P
10	Curva_2_P
11	Curva_2_P
12	Curva_3_P
24	Curva_3_P
25	Curva_1_P
39	Curva_1_P
40	Curva_5_P
46	Curva_5_P
47	Curva_6_P
63	Curva_6_P
64	Curva_4_P
74	Curva_4_P
75	Curva_7_P
161	Curva_7_P

Caso 4 unidades

numeros_x	nombres_curvas_minimas
5	Curva_1_P
8	Curva_1_P
9	Curva_2_P
11	Curva_2_P
12	Curva_4_P
20	Curva_4_P
21	Curva_2_P
30	Curva_2_P
31	Curva_1_P
34	Curva_1_P
35	Curva_9_P
49	Curva_9_P
50	Curva_5_P
63	Curva_5_P
64	Curva_12_P
81	Curva_12_P
82	Curva_14_P
89	Curva_14_P
90	Curva_11_P
98	Curva_11_P
99	Curva_15_P
210	Curva_15_P

Caso 10 unidades

numeros_x	nombres_curvas_minimas
5	Curva_1_P
8	Curva_1_P
9	Curva_2_P
9	Curva_2_P
10	Curva_4_P
10	Curva_4_P
11	Curva_5_P
11	Curva_5_P
12	Curva_6_P
16	Curva_6_P
17	Curva_8_P
17	Curva_8_P
18	Curva_9_P
18	Curva_9_P
19	Curva_11_P
38	Curva_11_P
39	Curva_76_P
42	Curva_76_P
43	Curva_61_P
43	Curva_61_P
44	Curva_70_P
44	Curva_70_P
45	Curva_76_P
67	Curva_76_P
68	Curva_243_P
97	Curva_243_P

Cont.

98	Curva_491_P
127	Curva_491_P
128	Curva_711_P
156	Curva_711_P
157	Curva_881_P
185	Curva_881_P
186	Curva_958_P
215	Curva_958_P
216	Curva_1012_P
225	Curva_1012_P
226	Curva_994_P
243	Curva_994_P
244	Curva_1019_P
269	Curva_1019_P
270	Curva_1017_P
283	Curva_1017_P
284	Curva_1023_P
525	Curva_1023_P

Caso 10 unidades con las unidades 1 y 3 indisponibles.

numeros_x	nombres_curvas_minimas
9	Curva_1_P
9	Curva_1_P
10	Curva_2_P
10	Curva_2_P
11	Curva_3_P
11	Curva_3_P
12	Curva_4_P
16	Curva_4_P
17	Curva_6_P
17	Curva_6_P
18	Curva_7_P
18	Curva_7_P
19	Curva_8_P
38	Curva_8_P
39	Curva_36_P
42	Curva_36_P
43	Curva_33_P
43	Curva_33_P
44	Curva_35_P
44	Curva_35_P
45	Curva_36_P
67	Curva_36_P
68	Curva_94_P
97	Curva_94_P
98	Curva_163_P
127	Curva_163_P

Cont.

128	Curva_216_P
156	Curva_216_P
157	Curva_242_P
185	Curva_242_P
186	Curva_251_P
215	Curva_251_P
216	Curva_255_P
400	Curva_255_P

D. Nombres entre curvas y unidades en Excel

Caso 3 unidades

Nombres_Unidades	Nombres_Curvas
3	Curva_1_P
1	Curva_2_P
2	Curva_3_P
1 + 3	Curva_4_P
2 + 3	Curva_5_P
1 + 2	Curva_6_P
1 + 2 + 3	Curva_7_P

Caso 4 unidades

Nombres_Unidades	Nombres_Curvas
3	Curva_1_P
4	Curva_2_P
1	Curva_3_P
2	Curva_4_P
3 + 4	Curva_5_P
1 + 3	Curva_6_P
2 + 3	Curva_7_P
1 + 4	Curva_8_P
2 + 4	Curva_9_P
1 + 2	Curva_10_P
1 + 3 + 4	Curva_11_P
2 + 3 + 4	Curva_12_P
1 + 2 + 3	Curva_13_P
1 + 2 + 4	Curva_14_P
1 + 2 + 3 + 4	Curva_15_P

Caso 10 unidades

Nombres_Unidades	Nombres_Curvas
3	Curva_1_P
4	Curva_2_P
1	Curva_3_P
5	Curva_4_P
6	Curva_5_P
2	Curva_6_P
7	Curva_7_P
8	Curva_8_P
9	Curva_9_P
3 + 4	Curva_10_P
10	Curva_11_P
3 + 5	Curva_12_P
1 + 3	Curva_13_P
3 + 6	Curva_14_P
3 + 7	Curva_15_P
2 + 3	Curva_16_P
3 + 8	Curva_17_P
4 + 5	Curva_18_P
3 + 9	Curva_19_P
1 + 4	Curva_20_P
4 + 6	Curva_21_P
10 + 3	Curva_22_P
1 + 5	Curva_23_P

Cont.

1 + 10 + 2 + 4 + 6 + 7 + 8 + 9	Curva_1011_P
10 + 2 + 4 + 5 + 6 + 7 + 8 + 9	Curva_1012_P
1 + 10 + 2 + 5 + 6 + 7 + 8 + 9	Curva_1013_P
1 + 10 + 2 + 3 + 4 + 5 + 6 + 7 + 8	Curva_1014_P
1 + 10 + 2 + 3 + 4 + 5 + 6 + 7 + 9	Curva_1015_P
1 + 10 + 2 + 3 + 4 + 5 + 6 + 8 + 9	Curva_1016_P
1 + 10 + 3 + 4 + 5 + 6 + 7 + 8 + 9	Curva_1017_P
1 + 10 + 2 + 3 + 4 + 5 + 7 + 8 + 9	Curva_1018_P
10 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9	Curva_1019_P
1 + 10 + 2 + 3 + 4 + 6 + 7 + 8 + 9	Curva_1020_P
1 + 10 + 2 + 3 + 5 + 6 + 7 + 8 + 9	Curva_1021_P
1 + 10 + 2 + 4 + 5 + 6 + 7 + 8 + 9	Curva_1022_P
1 + 10 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9	Curva_1023_P

*Debido a que son más de 1000 nombres, se ha omitido incluirlos todos.

Caso 10 unidades con las unidades 1 y 3 indisponibles.

Nombres_Unidades	Nombres_Curvas
4	Curva_1_P
5	Curva_2_P
6	Curva_3_P
2	Curva_4_P
7	Curva_5_P
8	Curva_6_P
9	Curva_7_P
10	Curva_8_P
4 + 5	Curva_9_P
4 + 6	Curva_10_P
5 + 6	Curva_11_P
2 + 4	Curva_12_P
4 + 7	Curva_13_P
5 + 7	Curva_14_P
2 + 5	Curva_15_P
4 + 8	Curva_16_P
6 + 7	Curva_17_P
4 + 9	Curva_18_P

Cont.

10 + 2 + 4 + 7 + 8 + 9	Curva_244_P
10 + 2 + 5 + 7 + 8 + 9	Curva_245_P
10 + 2 + 6 + 7 + 8 + 9	Curva_246_P
2 + 4 + 5 + 6 + 7 + 8 + 9	Curva_247_P
10 + 2 + 4 + 5 + 6 + 7 + 8	Curva_248_P
10 + 2 + 4 + 5 + 6 + 7 + 9	Curva_249_P
10 + 2 + 4 + 5 + 6 + 8 + 9	Curva_250_P
10 + 4 + 5 + 6 + 7 + 8 + 9	Curva_251_P
10 + 2 + 4 + 5 + 7 + 8 + 9	Curva_252_P
10 + 2 + 4 + 6 + 7 + 8 + 9	Curva_253_P
10 + 2 + 5 + 6 + 7 + 8 + 9	Curva_254_P
10 + 2 + 4 + 5 + 6 + 7 + 8 + 9	Curva_255_P

*Debido a que son más de 200 nombres, se ha omitido incluirlos todos.