UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Implementación de Modelo de Despacho Eléctrico para el Complejo Hidroeléctrico del Río Negro

Informe de Proyecto de Grado presentado por

Mathias Rodriguez Castro, Matias Ernesto Montaño Cobas, Ernesto Zampetti Garin

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Claudio Enrique Risso Montaldo
Ignacio Ramírez Paulino

Montevideo, 10 de diciembre de 2025

Agradecimientos

Agradecer siempre es importante, y en este caso resulta imprescindible. Queremos expresar nuestro más profundo agradecimiento a los tutores del proyecto, por su guía, paciencia y valiosa retroalimentación durante todo el proceso. Su apoyo y conocimiento han sido fundamentales para el desarrollo de este trabajo.

También queremos agradecer a la Facultad de Ingeniería de la Universidad de la República, por brindarnos los recursos, el entorno académico y las oportunidades necesarias para aprender y crecer como profesionales. La formación recibida y el acompañamiento institucional han sido pilares esenciales para llevar adelante este proyecto.

Finalmente, agradecemos a todas las personas que, de una u otra manera, han contribuido a que este trabajo sea posible. Su ayuda, aunque a veces silenciosa, ha dejado una huella importante en este logro.

Resumen

La presente memoria documenta el desarrollo de una herramienta computacional para la resolución del problema de Coordinación Hidrotérmica de Corto Plazo, centrado en la operación horaria del Complejo Hidroeléctrico del Río Negro. El sistema fue implementado en `C++` con una arquitectura modular y extensible, diseñada para representar de forma desacoplada agentes generadores, demandas, esquemas de despacho y *solvers*.

El trabajo se basa en una formulación matemática de la literatura, que modela en detalle la operación de las represas de Bonete, Baygorria y Palmar. La implementación se desarrolló en dos etapas: una versión inicial en `Python` para validación conceptual, seguida por una migración a `C++` con enfoque orientado a objetos.

El sistema permite instanciar agentes dinámicamente, resolver el modelo con distintos solvers y almacenar los resultados estructuradamente. Su diseño facilita la incorporación de nuevas unidades, demandas o estrategias sin modificar el núcleo. Como validación, se integró una unidad térmica rápida y un esquema de costo de agua no lineal en Bonete, comprobando la extensibilidad del sistema.

Se presentan además líneas de trabajo futuro, incluyendo agentes mixtos (como baterías), despachos con penalizaciones dinámicas y generación masiva de instancias para entrenamiento de modelos de aprendizaje automático. El resultado es una herramienta validada, flexible y reutilizable, aplicable en contextos académicos y de planificación energética.

Palabras clave: Optimización combinatoria, Coordinación Hidrotérmica de Corto Plazo, Programación Entera Mixta, Modelado energético, Planificación energética.

Índice general

Capítulo 1

Introducción

1.1. Contexto energético

La planificación y operación de sistemas eléctricos modernos constituye uno de los desafíos técnicos más relevantes de la actualidad, especialmente en contextos donde las *energías renovables no convencionales* adquieren un papel protagónico. En Uruguay, la transición energética desde mediados de los 2000 permitió una matriz diversificada y sustentable, con más del 95 % de la demanda eléctrica cubierta por fuentes renovables (Risso, Nesmachnow, y cols., 2024). Este cambio estructural ha reducido de forma significativa la dependencia de los combustibles fósiles y ha transformado profundamente las herramientas de gestión y control del sistema eléctrico nacional.

La integración masiva de fuentes eólica y solar —caracterizadas por su intermitencia y falta de control directo— ha introducido nuevos retos operativos, entre ellos la necesidad de desarrollar modelos de optimización que permitan anticipar y mitigar la variabilidad de la oferta renovable. En este marco, las centrales hidroeléctricas adquieren un rol estratégico como unidades de respaldo flexibles, capaces de compensar fluctuaciones horarias y diarias en la generación no despachable. Su operación eficiente resulta esencial para preservar la estabilidad y confiabilidad del sistema eléctrico (Risso, Nesmachnow, y cols., 2024).

El *Complejo Hidroeléctrico del Río Negro*, conformado por las represas de Rincón del Bonete, Baygorria y Palmar, es uno de los pilares fundamentales de la estrategia energética nacional. Aunque la potencia combinada de estas represas representa solo el 40 % del total hidroeléctrico del país (considerando únicamente el 50 % de Salto Grande que corresponde a Uruguay), la capacidad de almacenamiento del embalse de Bonete otorga al complejo una notable versatilidad operativa. El tiempo de vaciado del lago de Bonete supera ampliamente al de Salto Grande, lo que convierte a Bonete en el principal acumulador de energía del país. Cuando el lago está a cota plena y en situaciones de sequía, la

energía acumulada y utilizada eficientemente en el conjunto de represas sobre el Río Negro puede alcanzar hasta el 90 % del total hidroeléctrico nacional. Este comportamiento coordinado permite almacenar energía potencial y liberarla según las necesidades del sistema, posicionando al complejo como un elemento clave para el despacho eficiente de energía a corto plazo (Olivera, 2024).

La operación del Complejo Hidroeléctrico del Río Negro forma parte del proceso de *planificación operativa* del sistema eléctrico nacional, que abarca horizontes temporales que van desde minutos hasta varios años. En el corto plazo —usualmente entre 48 y 480 horas— se plantea el problema de *despacho y compromiso de unidades*, cuyo objetivo es determinar, para cada hora, qué centrales deben operar y cuánta energía debe generar cada una. El propósito es satisfacer la demanda al menor costo posible, respetando las *restricciones técnicas y operativas* de cada unidad generadora (Risso, Nesmachnow, y cols., 2024).

Este proceso depende de información proveniente de múltiples fuentes: costos de generación, pronósticos de demanda, aportes hídricos esperados, disponibilidad de energía renovable no controlable, intercambios internacionales y cronogramas de mantenimiento. El resultado es un plan operativo que combina generación hidroeléctrica, térmica y renovable, coordinando los distintos recursos para alcanzar un equilibrio económico y técnico del sistema. Para ello, se emplean *modelos de optimización* que representan con precisión las restricciones físicas e hidráulicas de cada central, así como las interdependencias entre sus decisiones de turbinación, vertido y almacenamiento (Risso, Nesmachnow, y cols., 2024; Risso, Cabrera, Porteiro, y Ibarburu, 2024).

En horizontes de mediano y largo plazo —que abarcan desde semanas hasta varios años— la planificación adquiere un enfoque estocástico. El objetivo en este nivel es asignar un *valor estratégico* a los recursos almacenados, particularmente al agua en los embalses, considerando distintos escenarios hidrológicos y de precios futuros. Estos modelos permiten cuantificar el *valor de oportunidad* del recurso hídrico y su influencia sobre las decisiones operativas de corto plazo (Risso, Nesmachnow, y cols., 2024).

La interacción entre los distintos horizontes de planificación sigue una estructura jerárquica bien definida: los modelos de largo plazo determinan las inversiones óptimas y la expansión de la capacidad instalada; los de mediano plazo estiman el valor del agua y establecen políticas de uso eficiente de los embalses; y los modelos de corto plazo, como el que se desarrolla en este trabajo, optimizan la operación horaria del sistema considerando las condiciones reales de generación, demanda e intercambio energético (Risso, Cabrera, y cols., 2024). Esta integración garantiza una gestión coherente de los recursos hidroeléctricos y una operación confiable del sistema bajo distintos escenarios de oferta y demanda.

El presente trabajo se enfoca en el horizonte de corto plazo, tomando como

condiciones de borde los costos de oportunidad del agua embalsada derivados de los modelos de mediano y largo plazo. Estos modelos alimentan al despacho de corto plazo mediante funciones de valor aproximadas que reflejan el costo esperado de las decisiones operativas actuales sobre el desempeño futuro del sistema. Se incorpora un elevado nivel de detalle operativo, modelando las restricciones técnicas de cada agente generador y su coordinación, con el objetivo de proporcionar una herramienta de apoyo para la toma de decisiones del operador del sistema.

1.2. Motivación y planteo del problema

La motivación principal de este trabajo radica en la necesidad de disponer de herramientas computacionales robustas, flexibles y escalables que permitan representar con precisión el funcionamiento horario del sistema hidroeléctrico en cascada, respetando sus restricciones físicas, hidrológicas y operativas. En particular, se apunta a superar las limitaciones de modelos simplificados o difíciles de adaptar, mediante una implementación moderna y modular de un *modelo de optimización combinatoria*.

En este contexto, investigaciones recientes —como las desarrolladas en el marco del simulador energético SimSEE (SimSEE, 2024)— han desarrollado modelos de despacho horario que integran múltiples unidades hidroeléctricas, restricciones técnicas complejas y criterios de eficiencia económica.

La fuerte no linealidad de la generación hidroeléctrica y la incertidumbre inherente a datos críticos —como la demanda, los aportes hídricos, los precios de combustibles y la generación renovable variable— han promovido el uso de métodos basados en Programación Dinámica Estocástica (Stochastic Dynamic Programming, SDP) (Bellman, 1957a, 1957b; van der Wal, 1980), especialmente en los horizontes de planificación de mediano y largo plazo. Estos métodos permiten representar la incertidumbre mediante escenarios y asignar valores estratégicos a los recursos almacenados.

En la *planificación de corto plazo*, el enfoque se centra en definir un plan optimizado de gestión para los días por venir, incorporando un elevado nivel de detalle en la modelización de las restricciones operativas. El sistema eléctrico requiere un control continuo que determine, entre otros aspectos, cuáles unidades generadoras deben estar operativas y cómo debe coordinarse su entrada y salida del sistema para satisfacer la demanda de manera eficiente y confiable (Olivera, 2024).

Dentro de los problemas de despacho a corto plazo, el que mejor se ajusta al caso uruguayo es el problema de Coordinación Hidrotérmica de Corto Plazo (Short-Term Hydrothermal Coordination, STHTC) (Farhat y El-Hawary, 2009). Este problema busca determinar el cronograma óptimo de generación de energía

para sistemas mixtos hidro-térmicos en un horizonte breve (generalmente de uno a siete días), minimizando los costos de producción térmica mientras se satisface la demanda y se garantiza el uso eficiente de los recursos hídricos limitados. El STHTC es particularmente relevante para Uruguay, país sin recursos de combustibles fósiles y líder mundial en integración de energías renovables, donde más del 95 % de la demanda eléctrica se satisface típicamente mediante fuentes renovables. Entre las fuentes despachables y controlables, la generación hidroeléctrica tiene un peso superlativo frente a otras renovables controlables como la biomasa, siendo la primera considerablemente más grande en capacidad y participación en la matriz energética (Risso, Nesmachnow, y cols., 2024; Risso y cols., 2025).

Históricamente, como se analizará en el Capítulo 2 (*Revisión de antecedentes*), el problema STHTC se ha abordado mediante modelos de Programación Matemática (MP). Una de sus principales ventajas es que permiten trabajar con variables continuas —como el volumen de agua almacenado en cada embalse— sin necesidad de discretizarlas. Esto resulta especialmente valioso cuando se busca reducir la cantidad de niveles discretos para acelerar los algoritmos de Programación Dinámica (DP) o Programación Dinámica Estocástica (SDP). En contraste, aunque la MP maneja naturalmente variables continuas, el marco conceptual de Bellman para DP/SDP exige un conjunto finito y discreto de estados, lo cual incrementa la complejidad computacional cuando dicho espacio es amplio.

Volviendo a los modelos de Programación Matemática (MP) para el corto plazo, los enfoques tradicionales explotaban el hecho de que el horizonte de planificación abarcaba pocos días, lo que posibilitaba:

1. Linealizar las producciones hidráulicas en torno a sus puntos de operación inicial;

2. Considerar los datos de la instancia —demanda, aportes hidrológicos y costos— como conocidos y determinísticos con hasta 72 horas de anticipación (Risso, Nesmachnow, y cols., 2024).

Previo a 2005, la generación eléctrica uruguaya era predominantemente controlable, y tanto los aportes hidrológicos como la demanda nacional podían ser pronosticados con alta precisión en ese horizonte temporal, lo que permitía que los sistemas de planificación de corto plazo fueran esencialmente determinísticos.

Sin embargo, una de las premisas que sostenían el uso de modelos de programación lineal en el corto plazo se vio afectada con la introducción a gran escala de energía eólica y, más recientemente, de energía solar, transformando el problema de despacho en uno de naturaleza estocástica. En el momento en que se comenzó a desarrollar SimSEE, la matriz energética uruguaya estaba cambiando drásticamente con la incorporación de una gran cantidad de generación renovable no controlable. Este cambio trajo consigo la necesidad de integrar

estas nuevas fuentes en los modelos de despacho y optimización existentes —o crear nuevos— que habían sido diseñados para un sistema mayoritariamente hidro-térmico (Olivera, 2024). Con la integración de fuentes eólicas, solares y de biomasa, estas fuentes renovables son priorizadas y suministran electricidad siempre que no excedan la demanda. Las unidades controlables deben entonces satisfacer la *demanda residual*: la diferencia entre la demanda total y la generación renovable no convencional (Risso, Nesmachnow, y cols., 2024). Esta formulación mediante demanda residual resulta particularmente conveniente para el mercado eléctrico uruguayo, dado que la obligación de recibir y despachar la producción no convencional en la red permite enmascarar la presencia explícita de estas fuentes en el modelo de optimización, simplificando su representación. De este punto en adelante, y dado el carácter determinístico del modelo desarrollado en este trabajo, no se volverá a elaborar sobre las fuentes eólica, solar o de biomasa. Actualmente, el problema se resuelve mediante SDP con herramientas como SimSEE, pero este enfoque enfrenta desafíos de escalabilidad que no pueden sostenerse indefinidamente.

Un problema inherente a la SDP es la llamada *maldición de la dimensionalidad* o *maldición de Bellman*, donde el espacio de estados sobre el que se trabaja tiende a crecer exponencialmente, volviendo al problema intratable en términos computacionales (Olivera, 2024; Risso, Nesmachnow, y cols., 2024; Risso y cols., 2025). Este fenómeno dificulta la incorporación de nuevos componentes al sistema cuando estos introducen estados adicionales. Ejemplos de tales componentes incluyen:

- ➢ Gestión de sistemas de almacenamiento de energía en baterías (Battery Energy Storage Systems, BESS);
- ➢ Planificación de demandas gestionables, como contratos donde el operador dispone de ventanas de tiempo para carga de vehículos eléctricos;
- ➢ Uso y gestión de contratos complejos de corto plazo, como intercambios energéticos internacionales;
- ➢ Operación de unidades con ciclos de arranque (commitments) complejos, como centrales de ciclo combinado.

Otro aspecto, más sutil pero igualmente relevante, es que la Programación Dinámica es un método directo: un algoritmo que resuelve recursivamente cada etapa del problema hasta completar el horizonte temporal, momento en el cual recién se obtiene la solución óptima. Antes de llegar a ese punto, no se dispone de soluciones intermedias factibles ni de información operativa útil. Esta característica —que convierte a DP/SDP en un enfoque de "todo o nada"—, combinada con el crecimiento exponencial del tiempo de ejecución con respecto al número de estados, genera un compromiso inevitable entre la cantidad de componentes a incluir en el modelo (la dimensión del vector de estados, o maldición de la dimensionalidad) y el nivel de refinamiento en la *discretización* de dichos estados. Agregar detalle o incorporar componentes adicionales incrementa de forma significativa la complejidad computacional, pudiendo volver el

problema intratable. Además, este enfoque aumenta el riesgo de que la operación del sistema eléctrico uruguayo dependa excesivamente de implementaciones de software particulares, lo que puede limitar la flexibilidad y dificultar la adaptación del sistema ante nuevos desafíos operativos.

En particular, el trabajo de (Risso, Nesmachnow, y cols., 2024) propone una formulación de *Programación Entera Mixta (MIP)* orientada a capturar de forma precisa la operación coordinada del sistema en cascada del Río Negro. Este modelo considera rezagos hidráulicos, restricciones operativas horarias, funciones de producción no lineales aproximadas mediante tramos lineales, y penalizaciones por demanda no satisfecha. Su estructura permite evaluar con alto nivel de detalle el impacto operativo de distintas decisiones de despacho bajo condiciones realistas.

Entre las principales motivaciones para desarrollar un modelo MIP se encuentran tres aspectos centrales. En primer lugar, trabajar dentro del marco estándar de la programación matemática aporta simplicidad, transparencia y facilidad para extender o adaptar la formulación. En segundo lugar, el uso de variables continuas permite representar de manera directa magnitudes físicas sin recurrir a discretizaciones, evitando así los errores y limitaciones asociados a ellas. Finalmente, la separación entre modelo e implementación posibilita emplear distintas herramientas de resolución, incluidas aquellas basadas en métodos iterativos que mejoran progresivamente la solución y permiten fijar un *gap de optimalidad*, como se hace en este trabajo.

Para el trabajo de referencia en el que este proyecto se enmarca, se ha fijado el *horizonte de planificación del despacho* en 15 días. Este requerimiento surge ante la necesidad de gestionar contratos internacionales y unidades térmicas con compromisos de arranque (commitments) complejos dentro del período de planificación. Ante la falta de pronósticos confiables de generación eólica y solar más allá de las 72 horas, el modelo recurre a una aproximación determinística, utilizando promedios estimados para los próximos quince días.

Respecto a los modelos de programación matemática utilizados en el país con anterioridad, este MIP destaca porque captura la operación hidroeléctrica de manera ajustada, incluso a lo largo de horizontes de planificación de 15 días; período superior al necesario para agotar el stock en los embalses de tres de las cuatro represas que componen el parque hidroeléctrico uruguayo.

La formulación fue concebida con fines investigativos y ha sido utilizada como base para posteriores desarrollos y experimentaciones computacionales, incluyendo el trabajo de (Olivera, 2024), que aporta una caracterización detallada de los cambios históricos en la gestión energética del país y el rol creciente del modelo de despacho como herramienta de apoyo a la toma de decisiones. En paralelo al desarrollo de esta tesis, el modelo de referencia ha incorporado unidades nuevas (incluyendo Salto Grande) y refinado el ajuste de otras.

1.3. Objetivos del trabajo

El presente proyecto se inscribe en la línea de investigación iniciada por (Risso, Nesmachnow, y cols., 2024), centrada en el modelado y optimización del despacho hidroeléctrico del sistema del Río Negro. Esta línea ha permitido desarrollar formulaciones matemáticas robustas para representar el problema de Coordinación Hidrotérmica de Corto Plazo, pero su implementación práctica aún requiere mayor consolidación. En particular, es necesario traducir dichas formulaciones en herramientas computacionales reproducibles, modulares y eficientes, capaces de integrarse en flujos de trabajo reales y de evolucionar junto con las necesidades del sistema eléctrico nacional.

En este contexto, el objetivo general del trabajo consiste en desarrollar una herramienta computacional robusta y extensible que resuelva el MIP propuesto por (Risso, Nesmachnow, y cols., 2024), garantizando la correcta representación de las restricciones físicas, hidráulicas y operativas del Complejo Hidroeléctrico del Río Negro. La plataforma debe posibilitar la interacción con distintos solvers de optimización, asegurar la trazabilidad de los resultados y permitir la incorporación controlada de nuevas funcionalidades sin comprometer la consistencia del modelo.

Desde un enfoque de ingeniería de software, se busca establecer una arquitectura modular orientada a objetos que separe de forma explícita las responsabilidades de cada componente —carga y validación de datos, construcción algebraica del modelo, interfaz con los solvers, resolución, *posprocesamiento* y análisis de resultados—. Este esquema promueve la mantenibilidad, la reutilización del código y la posibilidad de extender el sistema hacia futuras versiones o nuevos horizontes temporales sin necesidad de modificaciones estructurales.

De manera complementaria, el proyecto apunta a evaluar experimentalmente la fidelidad y desempeño del sistema mediante la resolución de un conjunto amplio de instancias representativas, comparando los resultados con las implementaciones previas en MATLAB y Python. El análisis busca verificar la equivalencia numérica de las soluciones, identificar cuellos de botella computacionales y cuantificar la ganancia en eficiencia y escalabilidad obtenida a partir de la nueva arquitectura.

Por último, el trabajo busca establecer una base tecnológica abierta y documentada que pueda ser reutilizada y ampliada en desarrollos futuros, tanto académicos como aplicados. Con ello, los objetivos del proyecto apuntan a integrar rigor técnico y utilidad práctica, facilitando la transición entre la formulación teórica del modelo y su implementación como herramienta operativa.

1.4. Alcance y resultados esperados

El alcance de este proyecto, según lo establecido en el bosquejo inicial, se limita a la implementación y validación computacional del modelo de Programación Entera Mixta presentado en (Risso, Nesmachnow, y cols., 2024): *A Mixed Combinatorial Optimization Model for the Río Negro Hydroelectric Complex*. Dicho modelo constituye la formulación base sobre la cual se estructura el desarrollo de la herramienta y representa el contenido central reportado en la Subsección 5.4.1 (*Modelo base*).

Este modelo base incorpora la operación coordinada de las tres represas del Complejo Hidroeléctrico del Río Negro (Bonete, Baygorria y Palmar), considerando rezagos hidráulicos, funciones de producción aproximadas mediante tramos lineales, balance hídrico intertemporal y penalizaciones por demanda no satisfecha. Su resolución se realiza mediante solvers de programación entera mixta, empleando un criterio de optimalidad basado en un GAP relativo del 0,1 %.

No obstante, durante el desarrollo del proyecto se incorporaron elementos adicionales provenientes del trabajo (Risso y cols., 2025): *An Enriched Mixed Combinatorial Optimization Model to Manage the Hydrothermal Dispatch for the Río Negro Hydroelectric Complex*, los cuales constituyen extensiones del alcance original. Estas extensiones incluyen la incorporación de una unidad térmica de arranque rápido y la utilización de funciones no lineales para el costo del agua en Bonete, junto con penalizaciones del costo de falla proporcionales a la demanda total. Dichas ampliaciones se documentan en las secciones Subsección 4.3.4 y Subsección 4.3.5, y su implementación tiene como propósito demostrar la extensibilidad y modularidad de la arquitectura desarrollada.

De este modo, el presente documento se estructura en los siguientes capítulos, cada uno abordando aspectos clave del desarrollo y validación del modelo propuesto:

En el Capítulo 2 (*Revisión de antecedentes*) se presenta una descripción histórica y técnica de los modelos y herramientas utilizadas en la operación del sistema eléctrico uruguayo para el despacho hidroeléctrico, junto con sus formulaciones matemáticas asociadas. El capítulo combina la revisión de los sistemas utilizados, como EDF, OPERGEN, SimSEE y MOP, el análisis de las metodologías de optimización en que se basan (LP, SDP, MIP), y una discusión de las limitaciones prácticas observadas en implementaciones previas. Este recorrido permite identificar los elementos que motivan el rediseño propuesto y establece el marco conceptual y técnico sobre el cual se apoya el desarrollo posterior del proyecto.

Por otro lado, en el Capítulo 3 (*Marco teórico*) se presentan los fundamentos

conceptuales y metodológicos que sustentan este trabajo. En dicho capítulo se introducen los elementos esenciales de la optimización matemática y las distintas clases de problemas relevantes para este estudio (LP, IP y MIP), destacando tanto sus aspectos teóricos como computacionales. Asimismo, se describe el rol de los modelos MIP en la representación de sistemas reales que combinan decisiones continuas y discretas, y se analizan las técnicas de resolución más empleadas en la práctica. El capítulo también aborda el despacho eléctrico como un problema de optimización intertemporal en sistemas hidroeléctricos, detallando sus variables, restricciones y criterios de operación. Finalmente, se desarrolla un marco de programación orientada a objetos que incluye principios fundamentales, estructuras de modelado y patrones de diseño —particularmente *Facade*, *Singleton* y *Strategy*— utilizados para la construcción del software implementado en este trabajo.

En el Capítulo 4 (*Desarrollo e implementación del modelo*) se integran los aspectos matemáticos, computacionales y experimentales del desarrollo de la herramienta. En primer lugar, se formaliza y analiza la formulación MIP utilizada para modelar la operación coordinada de Bonete, Baygorria y Palmar, explicando la función objetivo, las restricciones hidráulicas, los rezagos de tránsito entre embalses y las aproximaciones mediante tramos lineales de las curvas de producción. Posteriormente, dicha formulación es validada mediante un prototipo en `Python`, el cual reproduce 204 instancias de referencia, permitiendo verificar la consistencia del modelo e identificar aspectos prácticos de su resolución. Finalmente, se presenta la arquitectura modular implementada en `C++`, la cuál favorece la mantenibilidad y facilita la extensión del sistema, lo cual se demuestra mediante casos prácticos —como la incorporación de una unidad térmica de arranque rápido y el uso de una función no lineal del costo del agua en Bonete— que pueden añadirse sin modificar el núcleo del modelo.

Posteriormente, en el Capítulo 5 (*Experimentación*) se presenta el análisis empírico del desempeño del sistema en dos etapas diferenciadas. En primer lugar, se evalúa el comportamiento del modelo MIP original, resolviendo 204 instancias independientes bajo un criterio de optimalidad basado en un GAP relativo máximo del 0,1 %. A continuación, se estudia el modelo extendido —que incorpora un costo no lineal del agua en Bonete y un costo de falla proporcional a la demanda total— mediante la ejecución de cinco instancias adicionales, lo que permite una comparación preliminar con el escenario base.

En ambos casos, se adopta una metodología experimental unificada que registra cuatro componentes temporales por ejecución: construcción del modelo interno, traducción al formato del solver, tiempo de resolución y posprocesamiento. Esta estructura permite analizar la contribución de cada fase al tiempo total y facilita la comparación entre escenarios sin modificar el flujo de trabajo del sistema.

Por último, en el Capítulo 6 (*Conclusiones y trabajo futuro*) se presentan

las principales conclusiones del trabajo y las líneas de desarrollo futuro asociadas. A partir de la reimplementación del modelo de Coordinación Hidrotérmica de Corto Plazo mediante una arquitectura modular en `C++`, se sintetizan los avances obtenidos en términos de validación, extensibilidad y aplicabilidad del sistema tanto en contextos operativos como de investigación. Asimismo, se discuten posibles direcciones de evolución del sistema, incluyendo la incorporación de nuevos agentes y tecnologías, la ampliación del módulo de despacho y la integración con metodologías de aprendizaje automático orientadas a generar y analizar grandes volúmenes de instancias.

Capítulo 2

Revisión de antecedentes

2.1. Antecedentes y Evolución de Modelos de Despacho en Uruguay

Este proyecto se enmarca en la continuidad de una línea aplicada de investigación impulsada conjuntamente por la Universidad de la República (Udelar) y UTE, orientada al estudio y modelado de estrategias de operación del sistema eléctrico nacional. El eje principal de esta línea reside en la utilización eficiente de la hidroelectricidad como recurso de respaldo frente a la creciente penetración de fuentes renovables no controlables.

Para comprender el contexto histórico y técnico de los modelos y herramientas empleadas en Uruguay, se toma como referencia el trabajo de (Olivera, 2024), donde se analizan en detalle las características y evolución de las principales plataformas desarrolladas y utilizadas en el país —entre ellas, EDF, OPERGEN y SimSEE— (ver secciones 1.2.3–1.2.5 de dicha referencia).

Génesis y adopción del sistema EDF (1988–2018)
Los primeros modelos empleados en Uruguay incluyeron el software desarrollado a medida por *Électricité de France* (EDF), contratado por UTE a fines de la década de 1980 para resolver problemas de despacho de largo plazo (Ferreira, 2008; Poder Ejecutivo, 2002; Olivera, 2024). En la práctica, EDF se utilizó en producción aproximadamente entre 1988 y 2018 para optimización estocástica en horizontes anuales, con paso semanal, integrando demanda, aportes, disponibilidad de equipos e intercambios con países vecinos.

El sistema EDF se estructuraba en dos módulos principales: *MURVAGUA*, encargado de valorar el agua en Rincón del Bonete mediante SDP, discretizando el embalse en diez niveles y considerando cinco estados hidrológicos; y *MURDOC*, encargado de la simulación de trayectorias operativas mediante Monte Carlo, a partir de las valorizaciones estimadas (Ferreira, 2008; Olivera, 2024). En esta

configuración, la valorización del agua se realizaba exclusivamente para Rincón del Bonete, considerando valores nulos en Baygorria, Palmar y Salto Grande, que eran tratadas como plantas de paso (Olivera, 2024).

La metodología aplicada en EDF permitió capturar de forma explícita el costo de oportunidad del agua —concepto central en la planificación hidrotérmica— y fue pionera en la representación del sistema como un proceso estocástico con retroalimentación dinámica. Este enfoque, aunque limitado por el poder de cómputo de la época, sentó las bases conceptuales de los modelos posteriores, en particular en lo relativo al tratamiento de los estados hidrológicos, la discretización temporal por *postes* horarios y el cálculo de valores del agua sobre horizontes de varios años (Ferreira, 2008; Olivera, 2024).

A partir de la implantación del sistema EDF, se promovió además un entorno de cooperación técnica y científica entre UTE y la Universidad, que dio lugar a una serie de proyectos de investigación aplicada orientados a mejorar los algoritmos de optimización y simulación del sistema, así como al desarrollo de capacidades en modelado estocástico y cómputo de alto desempeño. Estas iniciativas consolidaron un lenguaje técnico común y la formación de recursos humanos especializados, generando una base de conocimiento nacional en torno a la planificación energética estocástica (Olivera, 2024).

OPERGEN: articulación de mediano y corto plazo (2002–2015)
Con el objetivo de complementar la visión de largo plazo de EDF, UTE incorporó el modelo *OPERGEN*, desarrollado por la consultora IBERDROLA–PSRI, que fue utilizado entre aproximadamente 2002 y 2015. Este modelo abordó la planificación de mediano y corto plazo (MP, CPC, CPS) mediante formulaciones determinísticas y mixtas, integrando los resultados de EDF —en particular, los valores del agua— como insumos iniciales para cada horizonte operativo (Ferreira, 2008; Poder Ejecutivo, 2002; Olivera, 2024).

El modelo se basaba en técnicas de Programación Lineal y Programación Lineal Entera Mixta (MIP), lo que permitía representar la dinámica térmica e hidráulica de forma integrada. El vínculo entre EDF y OPERGEN se establecía de manera jerárquica: el modelo de largo plazo valoraba el recurso hídrico y generaba políticas de almacenamiento, mientras que OPERGEN traducía esas políticas en decisiones operativas concretas, como la secuencia de arranques y paradas de unidades térmicas, el manejo de los tiempos de tránsito hidráulico y la evaluación de intercambios energéticos internacionales (Ferreira, 2008; Olivera, 2024).

OPERGEN introdujo innovaciones relevantes en la representación técnica del despacho, tales como:

- Modelos con variables binarias para representar arranques y paradas térmicas.

- ➢ Inclusión de *postes horarios* para capturar picos y valles de demanda intra-semanales.
- ➢ Modelos de tránsito de caudales y restricciones hidrológicas explícitas.
- ➢ Incorporación de contabilidades energéticas (como créditos de Salto Grande) y ajustes por sobrecostos asociados a la frecuencia de operación.

Todas estas características se encuentran documentadas en (Ferreira, 2008; Olivera, 2024).

Estas mejoras permitieron un modelado más realista y operativo, aunque con un aumento considerable en el costo computacional. Su estructura modular facilitó la calibración y el uso rutinario en la planificación operativa, pero carecía de flexibilidad para adaptarse a la irrupción masiva de energías renovables no controlables que caracterizó la década de 2010 (Olivera, 2024).

SimSEE (desde 2012)
Desde 2012, la Administración del Mercado Eléctrico (ADME) y UTE emplean la plataforma SimSEE, desarrollada en la Facultad de Ingeniería (FING, Udelar) bajo el proyecto PDT 47/12, con apoyo del MIEM, URSEA y el BID. A diferencia de sus predecesores —EDF y OPERGEN—, diseñados como aplicaciones cerradas, SimSEE fue concebido como una plataforma genérica y extensible, estructurada en módulos de agentes que representan unidades de generación, demanda, almacenamiento e interconexiones (Casaravilla, Chaer, y Alfaro, 2009; Chaer, 2008; Chaer y cols., 2013; Olivera, 2024).

En la etapa de optimización, el sistema resuelve políticas de operación y valores del agua de manera conjunta para los embalses Bonete–Baygorria–Palmar, aplicando SDP (Casaravilla y cols., 2009; Olivera, 2024). En la etapa de simulación, evalúa la política bajo distintos escenarios (históricos o generados mediante Monte Carlo), resolviendo en cada paso un problema lineal donde los agentes actúan de forma colaborativa, aportando restricciones y objetivos individuales (Chaer, 2008; Olivera, 2024). A diferencia de EDF y OPERGEN, la valorización del agua se realiza de forma simultánea para los tres embalses del Río Negro (Bonete, Baygorria y Palmar) (Olivera, 2024).

El diseño de SimSEE permite incorporar fácilmente nuevas tecnologías —como parques eólicos, fotovoltaicos o distintas representaciones de demanda—, debido a que estas fuentes no añaden estados adicionales al problema de optimización estocástica. En esos casos, la estructura del modelo de SDP permanece inalterada y el algoritmo continúa siendo escalable.

Sin embargo, la integración de agentes que sí poseen estados propios —por ejemplo, unidades de almacenamiento energético (baterías), embalses adicionales o dispositivos con dinámica interna relevante— modifica la dimensionalidad del espacio de estados. Esto genera un crecimiento exponencial del número de nodos en el árbol de decisiones y, por ende, del costo computacional del algoritmo de

resolución hacia atrás. Si bien la arquitectura de SimSEE está preparada para extenderse a este tipo de agentes, la escalabilidad del método por SDP se ve comprometida, por lo que su implementación requiere un rediseño específico o el uso de heurísticas y aproximaciones adicionales (Chaer y cols., 2013; Olivera, 2024).

MOP (actualidad)
Como parte de esta evolución continua, UTE ha desarrollado e implementado internamente *MOP* (Modelo de Operación), una herramienta para la planificación de mediano a largo plazo basada en Stochastic Dual Dynamic Programming (SDDP) (Mauriz, Porteiro, y Ibarburu, 2024). De código abierto y disponible públicamente (UTE, 2024), MOP representa el desarrollo más reciente en la línea de herramientas computacionales específicas para el sistema eléctrico uruguayo.

Síntesis y continuidad
El recorrido histórico descrito pone de manifiesto la existencia de una línea de desarrollo coherente, que combina continuidad conceptual con modernización tecnológica. En particular, pueden distinguirse tres capas metodológicas que coexisten en la práctica:

1. *Valorización del agua* en el largo plazo (EDF/SDP), con discretización semanal y representación estocástica de aportes hidrológicos.
2. *Planificación de mediano y corto plazo* (OPERGEN), que incorpora detalle operativo y la dinámica térmica-hidráulica.
3. *Plataformas colaborativas y genéricas* (SimSEE), capaces de extender esas abstracciones a un sistema con alta penetración renovable y estructura multiagente.

Este acervo constituye el *estado del arte local* sobre el que se apoya el presente trabajo (Olivera, 2024). A partir de esta base consolidada, se busca continuar la línea de desarrollo aplicada que ha caracterizado la cooperación UTE–Udelar, aportando nuevas herramientas metodológicas y computacionales que contribuyan a la evolución de los modelos de optimización y simulación del sistema eléctrico nacional.

En particular, SimSEE se distingue de otras herramientas desarrolladas específicamente para el sistema uruguayo por su arquitectura basada en agentes. Este paradigma permite modelar de forma flexible distintos tipos de participantes —como generadores, demandas o elementos de la red— organizados en una topología explícita de nodos y arcos. Sobre esta estructura es posible ejecutar ciclos de optimización y simulación que representan trayectorias de operación bajo incertidumbre (Casaravilla y cols., 2009; Chaer y cols., 2013). Esta capacidad de abstracción ha consolidado a SimSEE como referencia para el desarrollo de software de simulación energética en Uruguay, tanto para propósitos académicos como operativos (Olivera, 2024).

2.2. Formulaciones de Modelos y Herramientas de Software

Los modelos de despacho desarrollados y utilizados en Uruguay se sustentan tradicionalmente en formulaciones analíticas basadas en Programación Matemática y Programación Dinámica, aplicadas tanto a horizontes de largo como de corto plazo (Olivera, 2024). Estos enfoques, originados en los modelos EDF y OPERGEN, se consolidaron en la plataforma SimSEE, que integra técnicas de optimización determinística y estocástica dentro de una arquitectura modular orientada a agentes.

Entre las formulaciones más relevantes se destacan:

- **Programación Lineal (LP):** empleada en los subproblemas semanales del módulo MURVAGUA del sistema EDF y en las simulaciones de MURDOC, donde se linealizan funciones de costo y restricciones hidráulicas para estimar costos marginales (*shadow prices*) de los embalses (Olivera, 2024).
- **Programación Dinámica (DP) y SDP:** aplicadas en la valorización del agua en Bonete y extendidas en SimSEE para representar simultáneamente múltiples embalses y estados hidrológicos. Estas metodologías se basan en el Principio de Optimalidad de Bellman y permiten modelar decisiones secuenciales bajo incertidumbre hidrológica (Olivera, 2024).
- **Programación Lineal Entera Mixta (MIP):** formulaciones derivadas de (Risso, Nesmachnow, y cols., 2024), que incorporan restricciones hidráulicas, discretización de curvas de producción y tiempos de tránsito de caudales. Este enfoque posibilita representar decisiones binarias —como arranques térmicos o vertidos discretos— y restricciones no lineales mediante aproximaciones lineales por tramos.

En este contexto, el modelo desarrollado por (Risso, Nesmachnow, y cols., 2024) constituye una alternativa que evita la explosión dimensional típica de la SDP. A diferencia del enfoque utilizado en SimSEE —donde cada nueva unidad con estado propio incrementa la dimensión del espacio de estados y, con ello, el costo del proceso— la propuesta de (Risso, Nesmachnow, y cols., 2024) formula el problema de operación del Complejo del Río Negro dentro de un esquema completamente determinista de Programación Entera Mixta. De esta forma, la dinámica de los embalses, los retardos hidráulicos, las curvas de producción no lineales y las decisiones horarias se expresan directamente como restricciones lineales o linealizadas, en lugar de expandir el espacio de estados. En otras palabras, el "estado del sistema" deja de generar nuevas ramas en un árbol de decisiones y pasa a representarse mediante variables internas dentro de un único problema de optimización.

El modelo incorpora los estados hidráulicos de Bonete, Baygorria y Palmar como variables continuas y se resuelve mediante solvers de propósito general como

`CPLEX` o `Gurobi`. Estos solvers aplican métodos iterativos altamente optimizados —como variantes avanzadas de *Branch-and-Bound*, cortes fraccionarios y heurísticas de incumbencia— que permiten enfrentar con éxito instancias de gran escala, como se analizará en el Capítulo 5 (*Experimentación*). A su vez, el hecho de separar explícitamente el modelo matemático del algoritmo de resolución habilita la posibilidad de explorar otras familias de técnicas, como métodos de descomposición (por ejemplo, Benders o Lagrangiana), metaheurísticas especializadas o enfoques basados en inteligencia artificial, ampliando el conjunto de herramientas disponibles para abordar problemas de despacho de características similares.

Limitaciones y desafíos de las implementaciones actuales
Si bien las implementaciones prototipo desarrolladas en entornos como MATLAB y Python —entre ellas el modelo MIP de (Risso, Nesmachnow, y cols., 2024)— demostraron la validez funcional del enfoque, también evidencian las limitaciones típicas de los desarrollos iniciales: acoplamiento elevado entre componentes, dificultades de mantenimiento y una escalabilidad reducida para simulaciones de gran porte. En particular, la falta de separación clara entre la capa de modelado y la de resolución restringe la extensión de estos modelos hacia horizontes de planificación más amplios o hacia esquemas con acoplamiento hidráulico completo (Olivera, 2024). Estas limitaciones constituyen uno de los principales puntos de partida del presente trabajo, que busca diseñar un entorno de modelado flexible, mantenible y escalable, capaz de integrar técnicas de optimización híbridas y enfoques basados en agentes.

Síntesis
El examen histórico–técnico realizado permite identificar una trayectoria evolutiva bien definida en los modelos de despacho utilizados en Uruguay. Los enfoques basados en SDP han proporcionado, durante décadas, una solución metodológicamente sólida para la valorización del agua y la toma de decisiones bajo incertidumbre hidrológica. No obstante, estos métodos se encuentran inevitablemente limitados por la *maldición de la dimensionalidad* inherente al enfoque de Bellman, lo que restringe su capacidad para incorporar de forma escalable agentes con estados adicionales o dinámicas complejas.

En contraste, los solvers del estado del arte de Programación Matemática contemporáneos exhiben un rendimiento robusto al resolver modelos de tamaño operativo realista —particularmente en horizontes quincenales con paso horario— permitiendo representar simultáneamente la dinámica hidráulica, térmica y renovable con un nivel de detalle elevado. La evidencia empírica reciente muestra que su desempeño mantiene escalabilidad incluso ante la incorporación de nuevos agentes y restricciones, lo que constituye una ventaja comparativa decisiva respecto de las soluciones basadas exclusivamente en SDP.

Por otra parte, SimSEE ha demostrado ser una herramienta madura y altamente efectiva desde la perspectiva de ingeniería de software. Su arquitectura modular,

su modelo de agentes y su interfaz orientada al usuario final proporcionan un grado de usabilidad y parametrización que ha facilitado su adopción operativa y académica. Este nivel de integración contrasta con las implementaciones prototipo desarrolladas en entornos como `MATLAB` o `Python`, las cuales, si bien han permitido validar modelos recientes —incluido (Risso, Nesmachnow, y cols., 2024) y los nuevos desarrollos actualmente en preparación— presentan limitaciones en términos de mantenibilidad, extensibilidad y experiencia de usuario.

En conjunto, estos elementos ponen de manifiesto una brecha tecnológica clara: la ausencia de una arquitectura unificada que combine la expresividad y escalabilidad de los modelos modernos basados en Programación Matemática, con la modularidad, parametrización y facilidad de uso que caracterizan a SimSEE. Abordar esta brecha constituye el objetivo central de este proyecto.

En consecuencia, el presente trabajo se orienta al diseño de una abstracción arquitectónica y metodológica basada en la formulación MIP propuesta por (Risso, Nesmachnow, y cols., 2024), pero implementada en un entorno de software mantenible, extensible y accesible para el usuario final. El objetivo es contar con una base técnica que permita hacer evolucionar el sistema sin requerir reestructuraciones profundas ante futuras extensiones. En este sentido, en la Subsección 4.3.4 y en la Subsección 4.3.5 se presentan dos extensiones aplicadas a la formulación original: la incorporación de una unidad térmica rápida y la introducción de un costo del agua no lineal en Bonete, junto con una penalización del costo de falla proporcional a la demanda total.

Además, la arquitectura planteada habilita la incorporación de metodologías híbridas —como combinaciones entre SDP, MIP y técnicas de Aprendizaje por Refuerzo— así como el uso de enfoques de descomposición matemática y otras estrategias de optimización. Estas líneas potenciales se detallan en la Capítulo 6 (*Conclusiones y Trabajo Futuro*), donde se presentan posibles extensiones orientadas a ampliar el alcance funcional del sistema, incluyendo nuevas clases de agentes, variantes en el esquema de despacho y técnicas basadas en inteligencia artificial para responder a las necesidades operativas del sistema eléctrico uruguayo.

Capítulo 3

Marco Teórico

El presente capítulo expone los fundamentos teóricos necesarios para contextualizar el trabajo. Se presentan los conceptos básicos de optimización matemática y programación lineal entera-mixta (MIP), utilizados para modelar el despacho hidroeléctrico. Luego, se describen los principales elementos que caracterizan el problema del despacho económico de generación en sistemas con embalses. Finalmente, se introducen los conceptos de programación orientada a objetos y los patrones de diseño aplicados en la implementación del sistema.

3.1. Optimización matemática

La optimización matemática constituye una rama fundamental de la *investigación operativa* que se enfoca en la identificación de soluciones óptimas dentro de un conjunto de alternativas posibles, bajo un conjunto de restricciones previamente definidas. En términos formales, un problema general de optimización puede expresarse como:

$$\begin{aligned} \text{Minimizar (o maximizar)} \quad & f(x) \\ \text{sujeto a} \quad & x \in \mathcal{X} \end{aligned} \tag{3.1}$$

donde $f(x)$ representa la función objetivo que se desea optimizar, y $\mathcal{X}$ denota el conjunto de soluciones factibles, determinado por las restricciones estructurales del problema.

Los problemas de optimización pueden clasificarse según la naturaleza de la función objetivo, las restricciones y el dominio de las variables. Entre las categorías más relevantes se encuentran:

- **Programación lineal (LP):** tanto la función objetivo como las restricciones son lineales, y las variables son continuas.
- **Programación entera (IP):** todas las variables está restringida a tomar valores enteros.

- ➢ **Programación lineal entera-mixta (MIP):** algunas variables son enteras y otras continuas, con una formulación lineal de la función objetivo y las restricciones.

Desde hace varias décadas se conocen algoritmos iterativos finitos para resolver problemas de Programación Lineal (LP). El Método Símplex (Dantzig, 1949, 1951, 1963) constituye el enfoque más extendido en aplicaciones reales debido a su desempeño sobresaliente en la práctica, aun cuando su complejidad en el peor caso es de orden exponencial. Un avance decisivo se produjo en la década de 1980, cuando Karmarkar introdujo un algoritmo de Punto Interior (Karmarkar, 1984), demostrando que las instancias de LP pueden resolverse en tiempo polinomial con respecto al tamaño de la entrada. Este resultado estableció formalmente que la programación lineal pertenece a la clase de complejidad `P`. Actualmente, los solvers comerciales combinan variantes avanzadas de Símplex y de métodos de Punto Interior (Nocedal y Wright, 2006), logrando resolver problemas con cientos de millones de variables y restricciones con notable eficiencia.

La situación es marcadamente distinta en el caso de la Programación Lineal Entera-Mixta (MIP). Numerosos problemas pertenecientes a la clase `NP` admiten formulaciones equivalentes como modelos MIP, lo cual implica que la existencia de un algoritmo general eficiente para resolver cualquier MIP supondría resolver la conjetura `P` vs. `NP` (S. A. Cook, 1971; Fortnow, 2009), reconocida como uno de los Problemas del Milenio por el Clay Mathematics Institute (S. Cook, 2000). En términos teóricos, los MIP son problemas `NP-Hard`; sin embargo, la práctica moderna muestra un panorama más matizado. Los solvers industriales implementan técnicas de alto nivel —incluyendo branch-and-bound, generación dinámica de cortes válidos, presolving avanzado, heurísticas de incumbencia y estrategias de branching sofisticadas— que permiten resolver instancias con millones de variables y restricciones en tiempos competitivos, siempre que el problema no contenga subestructuras intrínsecamente intratables.

En consecuencia, si bien no existe una garantía teórica de eficiencia para los modelos MIP en términos generales, la evidencia empírica acumulada en las últimas décadas muestra que los solvers del estado del arte, como CPLEX y Gurobi, son capaces de resolver problemas de gran escala con un desempeño consistente, incluso cuando se incorporan nuevos agentes, estados o restricciones. En nuestro caso, este comportamiento se ha verificado experimentalmente, tal como se detalla en el Capítulo 5 (*Experimentación*).

3.2. Programación lineal entera-mixta (MIP)

La programación lineal entera-mixta (MIP, por sus siglas en inglés) es una técnica ampliamente utilizada en la modelación de sistemas que involucran decisiones discretas y continuas. Su formulación general se expresa de la siguiente manera:

$$\begin{aligned} \text{Minimizar} \quad & c^T x \\ \text{sujeto a} \quad & Ax \leq b \\ & x_j \in \mathbb{Z}, \quad \forall j \in \mathcal{I} \\ & x_j \in \mathbb{R}, \quad \forall j \notin \mathcal{I} \end{aligned} \tag{3.2}$$

donde $\mathcal{I}$ denota el conjunto de índices correspondientes a las variables que deben asumir valores enteros.

Este enfoque resulta particularmente útil en la formulación de problemas reales que requieren decisiones binarias o de conteo, tales como la activación de unidades generadoras, la asignación de recursos, o la planificación operativa de sistemas complejos. Sin embargo, su resolución plantea importantes desafíos computacionales, especialmente cuando el número de variables enteras y las restricciones del modelo aumentan considerablemente.

3.3. Despacho eléctrico como problema de optimización

El *despacho económico de generación* es un problema clásico de optimización en el sector energético, cuyo objetivo consiste en determinar, para cada período, la combinación de generación que permita satisfacer la demanda eléctrica al menor costo posible, respetando las restricciones técnicas y operativas del sistema.

En sistemas con alta participación hidroeléctrica, la presencia de embalses introduce un componente intertemporal que aumenta la complejidad del problema, dado que las decisiones de operación afectan no solo el período actual sino también los siguientes. En términos generales, este tipo de modelos se caracteriza por los siguientes elementos:

- **Variables de decisión:** potencia generada por unidad, caudal turbinado, volúmenes almacenados en embalses, entre otras.
- **Restricciones:** balance hídrico, balance de energía, límites de capacidad de generación y almacenamiento, restricciones ambientales, entre otras.
- **Función objetivo:** minimización de costos operativos, maximización del valor del recurso hídrico, o penalizaciones asociadas a déficits o excedentes de generación, entre otros criterios.

Un ejemplo representativo es el modelo MIP planteado por (Risso, Nesmachnow, y cols., 2024), en el que se basa este trabajo y cuya formulación se detalla en la Sección 4.1.

3.4. Programación orientada a objetos

La *programación orientada a objetos (POO)* constituye uno de los paradigmas fundamentales en el desarrollo moderno de software. Su objetivo principal es modelar los sistemas informáticos a partir de la estructura y el comportamiento de entidades del mundo real, representadas mediante *objetos* que encapsulan tanto datos como operaciones (Booch, 1991). Este enfoque promueve la modularidad, la reutilización del código y la escalabilidad de los sistemas, factores decisivos en proyectos de ingeniería de software de gran envergadura (Pressman, 2010).

3.4.1. Orígenes y fundamentos conceptuales

El paradigma orientado a objetos surge como una evolución de la *programación estructurada*, buscando superar las limitaciones de ésta en cuanto a la gestión de la complejidad (Larman, 2004). Su origen se remonta a los años 1960 y 1970 con lenguajes como `Simula` y `Smalltalk`, precursores de conceptos que más tarde se consolidarían en lenguajes ampliamente utilizados como `C++`, `Java`, `Python` y `C#` (Kay, 1993).

Desde el punto de vista teórico, la POO se fundamenta en la idea de que un sistema puede describirse mediante una colección de objetos que interactúan entre sí mediante el envío de mensajes. Cada objeto se considera una unidad autónoma que combina estado (atributos o variables internas) y comportamiento (métodos o funciones) (Meyer, 1997). Este modelo resulta especialmente compatible con la abstracción jerárquica y el pensamiento modular, pilares del diseño orientado a software mantenible y extensible (Sommerville, 2011).

3.4.2. Principios esenciales

Los principios esenciales de la POO son cuatro: abstracción, encapsulamiento, herencia y polimorfismo (Gamma, Helm, Johnson, y Vlissides, 1995).

- **Abstracción:** consiste en identificar las características y comportamientos esenciales de una entidad, ignorando los detalles irrelevantes. Permite construir modelos conceptuales que reflejan los aspectos más relevantes del problema que se busca resolver (Booch, 1991).
- **Encapsulamiento:** implica ocultar los detalles internos de un objeto, exponiendo únicamente una interfaz pública. Esto promueve la independencia entre módulos y mejora la seguridad y mantenibilidad del código (Meyer, 1997).
- **Herencia:** permite crear nuevas clases a partir de otras existentes, reutilizando y extendiendo su comportamiento. Este mecanismo fomenta la generalización jerárquica y la reutilización de código (Gamma y cols., 1995).

- **Polimorfismo:** posibilita que diferentes clases implementen métodos con el mismo nombre, pero con comportamientos distintos. De este modo, los objetos pueden ser manipulados de manera uniforme, incrementando la flexibilidad y extensibilidad del sistema (Larman, 2004).

3.4.3. Estructura y modelado en POO

En términos prácticos, la programación orientada a objetos se basa en tres unidades básicas: clases, objetos e interfaces (Sommerville, 2011).

- Una **clase** define la estructura y el comportamiento de un conjunto de objetos, actuando como una plantilla o molde.
- Un **objeto** es una instancia concreta de una clase, que mantiene su propio estado y responde a mensajes según las reglas definidas por la clase.
- Una **interfaz** establece un contrato que las clases pueden implementar, asegurando la compatibilidad entre componentes sin necesidad de conocer sus detalles internos.

El modelado orientado a objetos suele representarse mediante *diagramas UML (Unified Modeling Language)*, herramienta estándar que facilita la comunicación entre diseñadores, programadores y analistas (Fowler, 2004). En este contexto, los diagramas de clases, de secuencia y de casos de uso resultan particularmente relevantes para documentar las interacciones entre objetos y los flujos de ejecución del sistema.

3.4.4. Patrones de diseño

Los **patrones de diseño** constituyen un conjunto de soluciones probadas y reutilizables a problemas recurrentes que aparecen durante el proceso de diseño de software orientado a objetos (Gamma y cols., 1995). Su propósito no es proporcionar código reutilizable de forma directa, sino ofrecer estructuras conceptuales que orienten al diseñador en la toma de decisiones de arquitectura y en la organización de las clases y sus interacciones.

Desde un punto de vista histórico, el concepto de patrón fue introducido por (Alexander, Ishikawa, y Silverstein, 1977) en el ámbito de la arquitectura, y posteriormente adoptado en la ingeniería de software por el grupo conocido como los *Gang of Four* (GoF): Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides (Gamma y cols., 1995). Su obra seminal, *Design Patterns: Elements of Reusable Object-Oriented Software*, formalizó un catálogo de 23 patrones clasificados según su propósito y alcance, estableciendo una terminología común que transformó el modo en que se diseña software orientado a objetos.

Clasificación general de los patrones

Los patrones de diseño se dividen en tres categorías principales:

- ➢ **Patrones creacionales:** proporcionan mecanismos para crear objetos de manera controlada, promoviendo la flexibilidad y la independencia respecto a las clases concretas. Ejemplos destacados incluyen *Factory Method*, *Abstract Factory*, *Builder* y *Singleton*.
- ➢ **Patrones estructurales:** describen formas de componer clases y objetos para formar estructuras más complejas, favoreciendo la reutilización y el acoplamiento débil. Entre ellos se encuentran *Adapter*, *Decorator*, *Composite* y *Facade*.
- ➢ **Patrones de comportamiento:** se enfocan en la interacción y comunicación entre objetos, estableciendo modelos de colaboración estables y escalables. Ejemplos típicos son *Observer*, *Strategy*, *Command* y *State*.

Esta clasificación permite seleccionar el patrón adecuado según la naturaleza del problema, fomentando la coherencia arquitectónica y la mantenibilidad del sistema.

3.4.5. Aplicaciones en el contexto del presente trabajo

En el diseño del sistema desarrollado en este trabajo se identifican tres patrones de diseño fundamentales: **Facade**, **Singleton** y **Strategy**. Estos se detallarán a continuación.

Patrón *Facade*

El patrón *Facade* tiene como propósito proporcionar una interfaz unificada que agrupe la funcionalidad de uno o varios subsistemas complejos, de modo que el usuario pueda interactuar con el sistema a través de un único punto de acceso. Este patrón busca ocultar la complejidad interna y reducir el acoplamiento entre los componentes, promoviendo una estructura de software más clara y modular (Gamma y cols., 1995).

A través de una fachada, se encapsula la interacción con las clases subyacentes y se simplifica el uso de un sistema amplio o jerárquico. De esta forma, el cliente no necesita conocer la estructura interna de los subsistemas ni sus interdependencias, lo cual facilita la reutilización del código y la sustitución de componentes internos sin afectar al exterior.

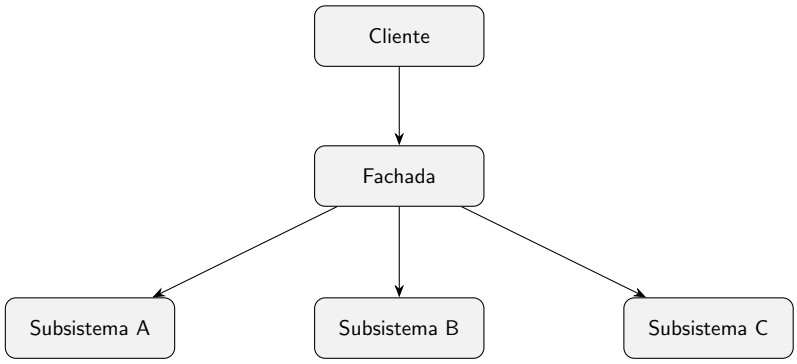

Figura 3.1: Diagrama UML del patrón *Facade*.

El patrón *Facade* se aplica típicamente cuando un sistema contiene múltiples subsistemas con interfaces complejas, y se requiere un único punto de entrada que oculte esa complejidad al usuario externo.

Patrón *Singleton*

El patrón *Singleton* tiene como finalidad garantizar que una clase posea una única instancia durante toda la ejecución del programa, proporcionando además un punto de acceso global a dicha instancia (Gamma y cols., 1995). Este enfoque resulta adecuado en situaciones donde es necesario coordinar acciones desde un único objeto compartido, como en la gestión de recursos globales o configuraciones centrales.

La implementación clásica de este patrón consiste en declarar el constructor de la clase como privado y ofrecer un método estático que controle la creación y el acceso a la instancia única, evitando así su duplicación.

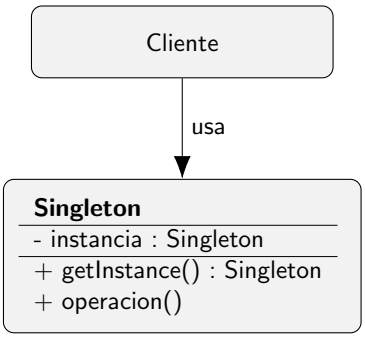

Figura 3.2: Diagrama UML del patrón *Singleton*.

Patrón *Strategy*

El patrón *Strategy* define una familia de algoritmos o comportamientos intercambiables, encapsulando cada uno en una clase independiente que implementa una interfaz común (Gamma y cols., 1995; Larman, 2004). Este patrón permite variar el comportamiento de un objeto de manera dinámica, sin modificar su estructura interna, y favorece la extensión del sistema mediante el principio de apertura/cierre.

Su aplicación se fundamenta en el uso del polimorfismo: el objeto que utiliza la estrategia mantiene una referencia a una interfaz abstracta, mientras que las implementaciones concretas representan las diferentes variantes del algoritmo.

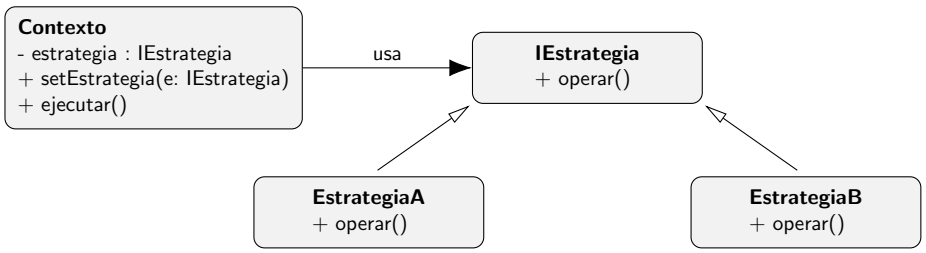

Figura 3.3: Diagrama UML del patrón *Strategy*.

El patrón *Strategy* promueve la flexibilidad y la reutilización del código, permitiendo sustituir algoritmos sin modificar la estructura de las clases que los utilizan.

Capítulo 4

Desarrollo e implementación del modelo

Este capítulo describe el desarrollo técnico del proyecto de manera unificada. En primer lugar, se presenta la formulación de Programación Entera Mixta propuesta en (Risso, Nesmachnow, y cols., 2024), detallando sus elementos fundamentales y su interpretación operativa. A partir de esta formulación, se implementó un prototipo inicial en `Python` que permitió validar el modelo, analizar su comportamiento y asegurar la consistencia de los resultados. Con esta base, se diseñó una arquitectura modular en `C++` que integra la formulación y posibilita su uso en un entorno mantenible, extensible y adecuado para una futura evolución del sistema.

4.1. Formulación matemática del modelo

La formulación del modelo MIP presentado en (Risso, Nesmachnow, y cols., 2024) describe de manera integrada la operación de las tres represas principales del sistema —Bonete, Baygorria y Palmar— incorporando tanto las variables hidráulicas y energéticas más relevantes como las restricciones operativas asociadas. El modelo combina variables continuas, tales como volúmenes almacenados, caudales turbinados y vertidos, con funciones de producción originalmente no lineales, las cuales se aproximan mediante tramos lineales para su tratamiento computacional. Además, incluye términos que penalizan el incumplimiento de la demanda eléctrica y representan el costo de oportunidad derivado del uso del recurso hídrico embalsado.

Formalmente el MIP se define de la siguiente manera:

$$
\begin{cases}
\min\limits_{x_{ih},y_{ih},v_{ih}} \; ca_{1h}(v_{1h,0}-v_{1h,T})+ca_{3h}(v_{3h,0}-v_{3h,T})+CF\sum\limits_{t=1}^{T}(d_t-g_{h,t}) & \\
d_t \geq g_{h,t}, & \text{(i)}\\
g_{h,t}=g_{1h,t}+g_{2h,t}+g_{3h,t}, & \text{(ii)}\\
g_{1h,t}=\hat{PtBon}(x_{1h,t},y_{1h,t},v_{1h,t}), & \text{(iii)}\\
g_{2h,t}=\hat{PtBay}(x_{2h,t},y_{2h,t},v_{3h,t}), & \text{(iv)}\\
g_{3h,t}=\hat{PtPal}(x_{3h,t},y_{3h,t},v_{3h,t}), & \text{(v)}\\
v_{1h,t}=v_{1h,t-1}+3600(a_{1h,t}-x_{1h,t}-y_{1h,t}), & \text{(vi)}\\
x_{2h,t}+y_{2h,t}=a_{2h,t}+x_{1h,t-8}+y_{1h,t-8}, & \text{(vii)}\\
v_{3h,t}=v_{3h,t-1}+3600(a_{3h,t}-x_{3h,t}-y_{3h,t}+x_{2h,t-16}+y_{2h,t-16}), & \text{(viii)}\\
(x_{ih},y_{ih},v_{ih})\in(X,Y,V), & \text{(ix)}
\end{cases}
$$

Para facilitar la interpretación de la notación empleada, se resumen a continuación los principales parámetros y variables del modelo:

Tabla 4.1: Parámetros y variables del modelo de optimización

Parámetros		Variables	
ca_{ih}	Costo del agua en la central i [USD/m^3].	$x_{ih,t}$	Caudal turbinado en la central i en la hora t [m^3/s].
CF	Penalización por demanda no satisfecha [USD/MWh].	$y_{ih,t}$	Caudal vertido en la central i en la hora t [m^3/s].
d_t	Demanda de energía en la hora t [MW].	$v_{ih,t}$	Volumen almacenado en el embalse de la central i en la hora t [m^3].
T	Horizonte temporal del modelo [horas].	$g_{ih,t}$	Energía generada por la central i en la hora t [MW].
$a_{ih,t}$	Aportes hídricos a la central i en la hora t [m^3/s].	$g_{h,t}$	Energía total generada por el complejo en la hora t [MW].
$\widehat{PtBon}(\cdot)$	Producción aproximada de Bonete [MW].	$v_{ih,0}$	Volumen inicial del embalse de la central i [m^3].
$\widehat{PtBay}(\cdot)$	Producción aproximada de Baygorria [MW].	$v_{ih,T}$	Volumen final del embalse de la central i [m^3].
$\widehat{PtPal}(\cdot)$	Producción aproximada de Palmar [MW].		

Finalmente, se indica a continuación la correspondencia entre los subíndices ih del modelo y las centrales hidroeléctricas del sistema:

Subindice ih	**Central hidroeléctrica**
$1h$	Rincón del Bonete
$2h$	Baygorria
$3h$	Palmar

A partir de la formulación presentada, se procede al análisis detallado de sus componentes, comenzando por la función objetivo y continuando con las restricciones que describen el comportamiento operativo del sistema.

Función objetivo

La función objetivo busca minimizar el costo total de operación del sistema hidroeléctrico a lo largo del horizonte temporal considerado. Este costo se descompone en tres componentes principales:

$$ca_{1h}(v_{1h,0} - v_{1h,T}) + ca_{3h}(v_{3h,0} - v_{3h,T}) + CF\sum_{t=1}^{T}(d_t - g_{h,t})$$

El primer término representa el valor económico del agua utilizada en la central de Bonete, al considerar la diferencia entre el volumen inicial y final del embalse ponderada por su costo de oportunidad ca_{1h} (USD/m^3). El segundo término es equivalente, pero aplicado a la central de Palmar. Ambos reflejan la intención de preservar agua embalsada cuando su uso no sea necesario, dado su valor potencial futuro.

El tercer término penaliza la demanda no satisfecha en cada período t, calculando la diferencia entre la demanda requerida d_t (MW) y la generación total $g_{h,t}$ (MW). Esta penalización se acumula a lo largo del horizonte y se multiplica por un parámetro CF (USD/MWh), que representa el costo asociado al incumplimiento del suministro eléctrico.

Restricciones

El modelo está sujeto a un conjunto de restricciones que reflejan las características físicas, técnicas y operativas del sistema hidroeléctrico en cascada.

Restricción (i)
Se establece que la generación no debe superar la demanda en cada período, es decir, $g_{h,t} \leq d_t; \forall t$. Esto garantiza un despacho factible desde el punto de vista del sistema eléctrico.

Restricción (ii)
En este caso, se define que la producción total $g_{h,t}$ (MW) resulta de la suma de

las energías generadas por cada una de las tres centrales: Bonete, Baygorria y Palmar.

Restricciones (iii), (iv) y (v)
En este conjunto de restricciones se modela la producción hidráulica de cada una de las tres centrales del Complejo del Río Negro—Bonete, Baygorria y Palmar—, estableciendo la relación entre la energía generada $g_{ih,t}$ (MW) y las variables físicas que la determinan: el caudal turbinado $x_{ih,t}$ (m^3/s), el vertido $y_{ih,t}$ (m^3/s) y el volumen almacenado $v_{ih,t}$ (m^3) en cada instante t.

Dado que la conversión de energía potencial en potencia eléctrica es intrínsecamente no lineal, especialmente por su dependencia simultánea de la altura del embalse y del nivel del río aguas abajo, es que adoptaron una formulación funcional cuadrática calibrada con parámetros técnicos específicos de cada central, la cuál se detalla en (Risso y cols., 2025) y se describe a continuación.

En condiciones hidrológicas normales o secas, la producción de potencia hidráulica puede aproximarse mediante las expresiones:

$$\begin{cases} \widehat{P}_t^{\text{Bon}} = (\hat{p}_{1h}^{(1)} x_{1h,t} - \hat{p}_{1h}^{(4)} x_{1h,t}^2) + (\hat{p}_{1h}^{(2)} v_{1h,t} - \hat{p}_{1h}^{(3)} v_{1h,t}^2) x_{1h,t}, \\ \widehat{P}_t^{\text{Bay}} = (\hat{p}_{2h}^{(1)} x_{2h,t} - \hat{p}_{2h}^{(2)} x_{2h,t}^2) - (\hat{p}_{2h}^{(3)} v_{3h,t} - \hat{p}_{2h}^{(4)} v_{3h,t}^2) x_{2h,t}, \\ \widehat{P}_t^{\text{Pal}} = (\hat{p}_{3h}^{(1)} x_{3h,t} - \hat{p}_{3h}^{(4)} x_{3h,t}^2) + (\hat{p}_{3h}^{(2)} v_{3h,t} - \hat{p}_{3h}^{(3)} v_{3h,t}^2) x_{3h,t}, \end{cases} \tag{4.1}$$

donde los parámetros $\hat{p}_{ih}^{(j)} > 0$ representan rendimientos promedio o coeficientes técnicos obtenidos a partir de la operación real de las plantas.

Los primeros términos de cada paréntesis en (4.1) constituyen funciones cóncavas en el caudal turbinado. Dado que su inclusión directa rompería la linealidad del modelo, se las aproxima mediante un conjunto de tangentes precomputadas, garantizando un error despreciable y preservando la convexidad del sistema. Sea $z_{jh,t}$ una variable auxiliar que representa la envolvente de estas tangentes; su definición surge del siguiente conjunto de desigualdades:

$$\begin{cases} z_{jh,t} \leq \hat{r}_{jh}^{(1)}\, x_{jh,t}, \\ z_{jh,t} \leq \hat{r}_{jh}^{(2)}\, x_{jh,t} + \hat{s}_{jh}^{(2)}, \\ z_{jh,t} \leq \hat{r}_{jh}^{(3)}\, x_{jh,t} + \hat{s}_{jh}^{(3)}, \end{cases} \tag{4.2}$$

donde $\hat{r}_{jh}^{(k)}$ y $\hat{s}_{jh}^{(k)}$ corresponden a las pendientes e interceptos de las tangentes que aproximan la relación entre potencia y caudal para cada planta j. Este procedimiento produce una envolvente poligonal cóncava que sustituye a la relación cuadrática original, permitiendo su incorporación dentro de un modelo lineal entero mixto. La Figura 4.1 ilustra esta aproximación: la curva roja representa la función cóncava original de producción, las rectas corresponden a las tangentes calculadas en puntos característicos, y la línea azul muestra la envolvente resultante empleada en el modelo.

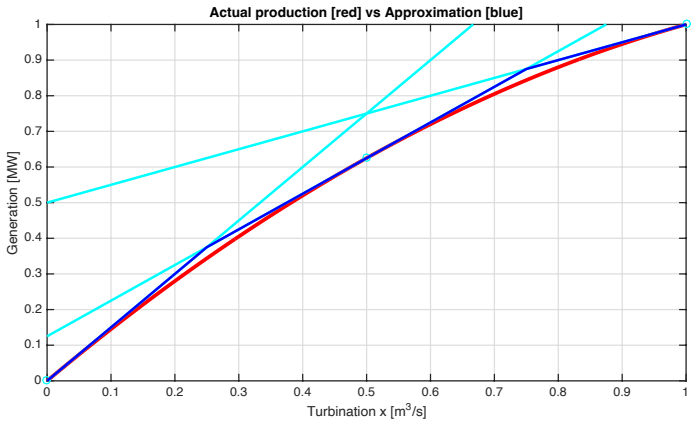

Figura 4.1: Producción hidráulica normalizada: curva real (rojo), tangentes en puntos característicos (cian) y aproximación lineal capturada (azul). Figura obtenida de (Risso y cols., 2025).

Los segundos términos del lado derecho de (4.1), en cambio, capturan el rendimiento incremental por altura del embalse, es decir, la variación adicional en la generación asociada al aumento del nivel del lago. Este efecto se considera significativo en Bonete y Palmar, pero despreciable en Baygorria. Para reflejarlo, se define la potencia total generada en dichas centrales como:

$$g_{jh,t} = z_{jh,t} + \sum_{i=1}^{3} \theta_{jh,t}^{(i)}, \qquad j \in \{1, 3\}, \tag{4.3}$$

donde las variables $\theta_{jh,t}^{(i)} \geq 0$ representan incrementos discretos de producción asociados al aumento de la altura del embalse sobre umbrales de discretización.

En el caso de Bonete, la dinámica de activación de estos incrementos se modela mediante el siguiente conjunto de restricciones:

$$\begin{cases} 0 \leq \theta_{1h,t}^{(1)} \leq (\hat{p}_{1h}^{(2)} \overline{V1h1} - \hat{p}_{1h}^{(3)} \overline{V1h1}^{2})\, x_{1h,t}, \\ 0 \leq \theta_{1h,t}^{(2)} \leq (\hat{p}_{1h}^{(2)} \Delta\overline{V1h2} - \hat{p}_{1h}^{(3)} \Delta\overline{V1h2}^{2})\, x_{1h,t}, \\ 0 \leq \theta_{1h,t}^{(2)} \leq 680\, (\hat{p}_{1h}^{(2)} \Delta\overline{V1h2} - \hat{p}_{1h}^{(3)} \Delta\overline{V1h2}^{2})\, \varphi_{1h,t}^{(1)}, \\ 0 \leq \theta_{1h,t}^{(3)} \leq (\hat{p}_{1h}^{(2)} \Delta\overline{V1h3} - \hat{p}_{1h}^{(3)} \Delta\overline{V1h3}^{2})\, x_{1h,t}, \\ 0 \leq \theta_{1h,t}^{(3)} \leq 680\, (\hat{p}_{1h}^{(2)} \Delta\overline{V1h3} - \hat{p}_{1h}^{(3)} \Delta\overline{V1h3}^{2})\, \varphi_{1h,t}^{(2)}, \end{cases} \tag{4.4}$$

donde la constante 680 [m^3/s] representa la máxima turbinación admisible en Bonete. Las variables $\varphi_{1h,t}^{(k)}$ determinan la activación secuencial de los tramos de altura, según el volumen actual $v_{1h,t}$ (m^3) del embalse:

$$\begin{cases} \varphi^{(1)}_{1h,t} \leq 1 - \dfrac{\overline{V1h2} - v_{1h,t}}{M_1}, \\ \varphi^{(2)}_{1h,t} \leq 1 - \dfrac{\overline{V1h3} - v_{1h,t}}{M_1}, \end{cases} \tag{4.5}$$

siendo M_1 un parámetro suficientemente grande que evita activaciones espurias y garantiza la correcta linealización lógica del sistema. Para establecer el orden de activación de estas variables, se incluye la siguiente restricción:

$$\varphi^{(2)}_{1h,t} \leq \varphi^{(1)}_{1h,t} \tag{4.6}$$

Por otro lado, en Baygorria, al tratarse de una planta de pasada sin capacidad de almacenamiento, el segundo término de (4.1) no aplica, simplificándose a:

$$g_{2h,t} = z_{2h,t} \tag{4.7}$$

Finalmente, en el caso de Palmar, la formulación es análoga, aunque con un nivel de discretización más fino debido a la mayor capacidad de regulación del embalse. Así para el segundo término en (4.1) se plantean las siguientes restricciones:

$$\begin{cases} 0 \leq \theta^{(1)}_{3h,t} \leq (\hat{p}^2_{3,h}\overline{V3h1} - \hat{p}^3_{3,h}\overline{V3h1}^2)\, x_{3h,t}, \\ 0 \leq \theta^{(2)}_{3h,t} \leq (\hat{p}^2_{3,h}\overline{V3h2} - \hat{p}^3_{3,h}\overline{V3h22})\, x_{3h,t}, \\ 0 \leq \theta^{(2)}_{3h,t} \leq 1380\,(\hat{p}^2_{3,h}\overline{V3h2} - \hat{p}^3_{3,h}\overline{V3h22})\, \varphi^{(1)}_{3h,t}, \\ 0 \leq \theta^{(3)}_{3h,t} \leq (\hat{p}^2_{3,h}\overline{V3h3} - \hat{p}^3_{3,h}\overline{V3h32})\, x_{3h,t}, \\ 0 \leq \theta^{(3)}_{3h,t} \leq 1380\,(\hat{p}^2_{3,h}\overline{V3h3} - \hat{p}^3_{3,h}\overline{V3h32})\, \varphi^{(2)}_{3h,t}, \\ 0 \leq \theta^{(4)}_{3h,t} \leq \hat{(}\hat{p}^2_{3,h}\overline{V3h4} - \hat{p}^3_{3,h}\overline{V3h42}), x_{3h,t}, \\ 0 \leq \theta^{(4)}_{3h,t} \leq 1380\,(\hat{p}^2_{3,h}\overline{V3h4} - \hat{p}^3_{3,h}\overline{V3h42})\, \varphi^{(3)}_{3h,t}, \\ 0 \leq \theta^{(5)}_{3h,t} \leq (\hat{p}^2_{3,h}\overline{V3h5} - \hat{p}^3_{3,h}\overline{V3h52})\, x_{3h,t}, \\ 0 \leq \theta^{(5)}_{3h,t} \leq 1380\,(\hat{p}^2_{3,h}\overline{V3h5} - \hat{p}^3_{3,h}\overline{V3h52})\, \varphi^{(4)}_{3h,t}, \end{cases} \tag{4.8}$$

donde la constante 1380 [m^3/s] representa la máxima turbinación admisible en Palmar.

Las variables binarias $\varphi^{(i)}_{3h,t}$ determinan la activación secuencial de los distintos niveles del embalse, en función del volumen almacenado $v_{3h,t}$ (m^3):

$$\begin{cases} \varphi^{(1)}_{3h,t} \leq 1 - \frac{\overline{V3h2} - v_{3h,t}}{M3}, \\ \varphi^{(2)}_{3h,t} \leq 1 - \frac{\overline{V3h3} - v_{3h,t}}{M3}, \\ \varphi^{(3)}_{3h,t} \leq 1 - \frac{\overline{V3h4} - v_{3h,t}}{M3}, \\ \varphi^{(4)}_{3h,t} \leq 1 - \frac{\overline{V3h5} - v_{3h,t}}{M3}, \end{cases} \tag{4.9}$$

siendo M_3 un parámetro suficientemente grande que garantiza la correcta linealización de las relaciones lógicas. Para asegurar la activación ordenada de los niveles discretos, se imponen además las restricciones:

$$\varphi^{(4)}_{3h,t} \leq \varphi^{(3)}_{3h,t} \leq \varphi^{(2)}_{3h,t} \leq \varphi^{(1)}_{3h,t}. \tag{4.10}$$

Estas restricciones garantizan que los tramos superiores del embalse solo puedan activarse cuando los niveles inferiores ya se encuentran operativos.

El conjunto de ecuaciones (4.2)–(4.10) completa la descripción de las restricciones (iii)–(v).

En síntesis, la formulación traduce la física no lineal de la producción hidráulica del complejo en un sistema lineal equivalente, combinando:

(I) una aproximación por tangentes para la parte cóncava en el caudal turbinado, y

(II) un esquema de activación por niveles discretos para capturar el efecto de altura del embalse.

Restricciones (vi), (vii) y (viii)
La dinámica del volumen en cada embalse se modela mediante las restricciones (vi), (vii) y (viii). En Bonete (vi), el volumen al instante t resulta del volumen en $t-1$, más los aportes hídricos registrados, menos el caudal turbinado y el vertido. En Baygorria (vii), además de sus propios aportes, se incorpora el agua proveniente de Bonete con un rezago de 8 horas, reflejando su ubicación aguas abajo. Finalmente, Palmar (viii) recibe también aportes diferidos desde Baygorria con un rezago de 16 horas, además de su aporte natural y de las salidas por turbinado y vertido.

Restricciones (ix)
Finalmente, esta restricción impone que las variables de decisión —caudal turbinado $x_{ih,t}$ (m^3/s), vertido $y_{ih,t}$ (m^3/s) y volumen almacenado $v_{ih,t}$ (m^3)— respeten los límites técnicos y físicos de operación de cada central. En términos prácticos, estos conjuntos (X, Y, V) representan restricciones simples de cota:

$$x^{\text{mín}}_{ih,t} \leq x_{ih,t} \leq x^{\text{máx}}_{ih,t}, \qquad y^{\text{mín}}_{ih,t} \leq y_{ih,t} \leq y^{\text{máx}}_{ih,t}, \qquad v^{\text{mín}}_{ih,t} \leq v_{ih,t} \leq v^{\text{máx}}_{ih,t}, \quad \forall i, t$$

donde los límites inferior y superior corresponden a las capacidades hidráulicas máximas de turbinado y vertido para cada planta, así como a los volúmenes mínimo y máximo admisibles de cada embalse. Estas restricciones garantizan que la operación resultante sea técnica y físicamente factible.

Detalles adicionales

El modelo completo incorpora funciones de producción no lineales para las centrales de Bonete, Baygorria y Palmar, dependientes del volumen almacenado, del caudal turbinado y del vertido. En cambio, la versión MIP reemplaza dichas relaciones por aproximaciones lineales por tramos, lo que permite conservar la estructura lineal del modelo y asegurar su resolubilidad mediante técnicas de Programación Entera Mixta.

Como resultado de esta simplificación, la solución obtenida no necesariamente coincide con la que se derivaría del modelo no lineal original. Por esta razón, una vez resuelto el MIP se lleva a cabo un ajuste posterior de los niveles de turbinación y vertido, de modo que la generación estimada sea consistente con las funciones de producción reales de cada central hidroeléctrica. Este procedimiento permite verificar la factibilidad hidráulica del despacho propuesto y obtener una estimación más precisa del valor de la función objetivo, bajo las condiciones operativas efectivas del sistema.

Debido a ello, una vez obtenida la solución, esta debe *posprocesarse* para realizar el ajuste correspondiente.

4.2. Validación del modelo mediante prototipo en Python

Con el objetivo de validar la formulación del MIP detallado anteriormente y comprender su dinámica operativa, se desarrolló un prototipo funcional en `Python`, empleando la biblioteca `Pyomo` por su flexibilidad, compatibilidad con múltiples solvers y amplia adopción en el ámbito académico. El desarrollo se apoyó en una base de código fuente proporcionada por el supervisor Ignacio Ramírez, alojada en un repositorio privado de GitLab, correspondiente al proyecto del Convenio UTE–IIE–INCO 2023. Esta base, que utilizaba la biblioteca *python-mip* y el solver abierto CBC, sirvió como punto de partida para la implementación y adaptación de las funcionalidades requeridas en el prototipo.

Esta primera etapa permitió verificar la consistencia del modelo y su correcta implementación mediante la comparación con un conjunto de 204 instancias previamente resueltas por el equipo de investigación original. Cada una de estas instancias incluía tanto la formulación como la solución óptima obtenida, lo que permitió realizar una validación cruzada precisa. La reproducibilidad fue un aspecto central del desarrollo, garantizando que el sistema fuera capaz de cargar escenarios, ejecutar distintos solvers y generar resultados consistentes con los casos de referencia.

El prototipo se estructuró de forma modular, con un diseño orientado a funciones independientes pero interconectadas. El código principal, contenido en el

archivo `main.py`, coordina la ejecución general del sistema e integra los distintos módulos. Este componente actúa como punto de entrada e interfaz de usuario, permitiendo seleccionar instancias de prueba, configurar parámetros y visualizar resultados a través de un menú interactivo por terminal.

La lectura y organización de los datos de entrada se implementó en el módulo `cargar_instancia.py`, encargado de procesar archivos de texto o comprimidos (`.gz`) y construir una estructura unificada de información denominada `instancia`. Este diccionario reúne los aportes hídricos a los embalses, la demanda horaria, los volúmenes iniciales y finales, y los costos asociados a la operación del complejo. A continuación, el módulo `pre_procesamiento.py` transforma dicha información en un conjunto de parámetros listos para ser utilizados por el modelo de optimización, incorporando límites técnicos, coeficientes hidráulicos y constantes de operación.

La representación matemática del problema fue implementada en el módulo `construir_modelo.py`, donde se definen los conjuntos, variables, restricciones y la función objetivo siguiendo la formulación del MIP. Este módulo traduce la estructura física y operativa del sistema hidroeléctrico en un modelo algebraico compatible con `Pyomo`, que puede resolverse con diferentes solvers. La gestión de estos solvers se realiza a través del módulo `ejecutar_solver.py`, que permite seleccionar entre opciones abiertas como CBC o comerciales como Gurobi y CPLEX, controlando el proceso completo de resolución y el manejo de los resultados.

Una vez obtenida la solución, el módulo `verificar_solucion.py` se encarga de validar los valores óptimos calculados, comparándolos con los resultados esperados de las instancias originales. Posteriormente, `pos_procesamiento.py` ajusta y refina los resultados del solver, corrigiendo posibles discrepancias respecto a las demandas horarias y los límites hidráulicos. En esta etapa se calculan las potencias efectivas de cada central y se reconstruye la trayectoria de operación del complejo hidroeléctrico. Finalmente, el módulo `guardar_resultados.py` almacena los resultados en formato de texto o CSV, con una estructura tabular estandarizada que facilita su análisis posterior.

El prototipo desarrollado cumplió su propósito de validar la formulación matemática y sentó las bases conceptuales y estructurales para el posterior desarrollo en `C++`. La experiencia adquirida en esta etapa permitió afinar la comprensión del modelo, consolidar criterios de diseño modular y definir una metodología de trabajo que guiaría la implementación final del sistema. Además, este entorno de experimentación temprana ofreció la oportunidad de analizar el desempeño de distintas estrategias de resolución, permitiendo evaluar comparativamente el comportamiento de diversos solvers en instancias reales del problema.

Durante esta etapa se evaluaron tres solvers de optimización: CBC, CPLEX y Gurobi. El análisis se centró exclusivamente en los tiempos de ejecución al

resolver 204 instancias del modelo, manteniendo fija su estructura y variando únicamente los datos de entrada. Todas las pruebas se realizaron en el entorno descrito en la Sección 5.3.

Los resultados evidenciaron diferencias claras en la velocidad de resolución entre los solvers. Gurobi y CPLEX obtuvieron los mejores tiempos, resolviendo las instancias de manera eficiente. CBC presentó el peor desempeño entre los tres, aunque se mantiene como una alternativa de código abierto.

A continuación se presenta un resumen de los tiempos mínimos, promedio y máximos observados para cada solver.

Tabla 4.2: Resumen de tiempos de ejecución por solver

Solver	**Tiempo mínimo**(s)	**Tiempo promedio**(s)	**Tiempo máximo**(s)
CBC	4,2	184.462625	3167.52
Gurobi	0,58	9,540784	174,97
CPLEX	0,51	6,994411	181,41

Con el objetivo de garantizar un criterio homogéneo de comparación, todos los solvers fueron configurados con un conjunto común de parámetros: una tolerancia relativa de optimalidad (MIP Gap) del 0,1 % y el uso de todos los núcleos disponibles del procesador. El entorno de cómputo empleado se describe en el Capítulo 5.

Cabe señalar que, se pueden establecer límites de tiempo para las ejecuciones ya que el solver encontraba soluciones factibles de buena calidad sin alcanzar la verificación formal de optimalidad. Este comportamiento es habitual en entornos de optimización aplicada, donde, además de las restricciones temporales, la naturaleza no convexa del problema puede impedir la certificación exacta del óptimo global. En estos contextos, se prioriza obtener soluciones suficientemente buenas en tiempos razonables por sobre la certificación matemática exacta de optimalidad, especialmente en escenarios operativos donde las decisiones deben tomarse bajo estrictas limitaciones de tiempo.

Una vez construido el modelo matemático, el sistema permite al usuario seleccionar el solver deseado, el cual ejecuta el proceso de resolución respetando los parámetros definidos. Los resultados obtenidos son luego almacenados, procesados y comparados con soluciones previamente validadas, lo que permite evaluar tanto la calidad de la solución como la consistencia de la implementación.

Durante la experimentación con las 204 instancias, se observó que el margen de error entre las soluciones obtenidas por el sistema y las soluciones originales proporcionadas por el equipo de investigación fue en todos los casos inferior al

1 %.

4.3. Arquitectura computacional e implementación en C++

Finalizada la fase de validación del modelo, esta etapa del proyecto se orienta al diseño e implementación de una herramienta computacional flexible y eficiente, capaz de formular y resolver el MIP definido previamente. El objetivo principal es disponer de una plataforma que permita obtener soluciones de manera consistente y que, a su vez, facilite la incorporación de nuevas funcionalidades, la elección del solver y la adaptación de la estructura del modelo según las necesidades del usuario.

Para cumplir con estos objetivos, se optó por una implementación en C++ con un enfoque orientado a objetos. Si bien la mayor carga computacional recae en los solvers externos cómo se verá en Capítulo 5 (*Experimentación*), el uso de C++ permite una estructura modular y un control preciso de los componentes del sistema (gestión de datos, formulación del modelo, interfaz con el solver y manejo de resultados), favoreciendo una implementación clara, mantenible y fácilmente extensible. Además, este entorno de desarrollo resulta familiar para el equipo, lo cual contribuyó a un avance más eficiente en la construcción de la herramienta.

El diseño general de este software se estructura en torno a una arquitectura sencilla pero escalable, organizada en módulos bien definidos, lo que permite abstraer la lógica del modelo de optimización del resto del sistema, facilitando tareas como la incorporación de nuevas unidades generadoras, la implementación de restricciones adicionales o el cambio del solver sin alterar el núcleo del programa.

A continuación, se detallan las funcionalidades específicas de cada clase y sus responsabilidades dentro del flujo de ejecución. Posteriormente, se expone el proceso de uso estándar del sistema, desde la carga de datos hasta la visualización de resultados, incluyendo los comandos disponibles para el usuario.

En las secciones finales se presentan dos extensiones desarrolladas para mostrar la flexibilidad del software. La primera consiste en incorporar una unidad térmica adicional que no formaba parte de la formulación original del modelo. La segunda introduce un costo del agua de Bonete con comportamiento no lineal, junto con un costo de falla proporcional a la demanda. Ambas modificaciones se presentan como ejemplos de cómo el diseño modular adoptado permite realizar ajustes estructurales sin reescribir componentes centrales del sistema.

4.3.1. Diagrama general de clases

La Figura 4.2 ofrece una visión general de la arquitectura del sistema implementado en `C++`, desarrollada para abordar el problema de MIP planteado.

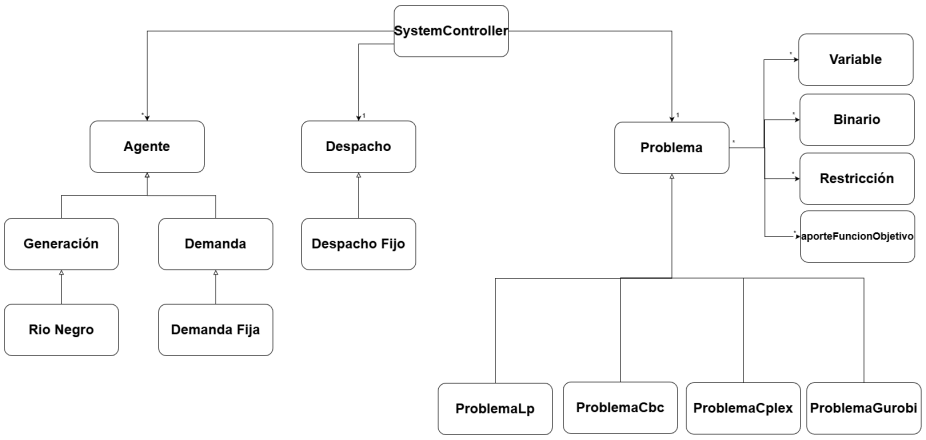

Figura 4.2: Esquema conceptual basado en una simplificación del modelo UML

El diseño se basó en la idea de mantener separados distintos componentes lógicos del sistema: agentes del modelo (generación y demanda), estrategias de despacho, representación del problema de optimización y elementos auxiliares como variables y funciones objetivo.

En la sección siguiente se describe en detalle el rol de cada clase, sus responsabilidades y la forma en que interactúan dentro del flujo de ejecución del sistema.

4.3.2. Descripción de clases y responsabilidades

SystemController

Constituye el núcleo funcional de la arquitectura implementada. Su diseño como *singleton* garantiza una única instancia global que centraliza el control del sistema durante toda la ejecución, coordinando los módulos de agentes, despacho y resolución. A través de esta clase se canaliza toda la interacción con el sistema, permitiendo al usuario inicializar el entorno de ejecución, incorporar agentes de generación y demanda, seleccionar la estrategia de despacho y definir el solver con el cual se formulará y resolverá el modelo de optimización.

A través de esta clase se organiza el flujo de trabajo completo, desde la configuración del horizonte de planificación hasta la ejecución del modelo o la generación del archivo correspondiente, manteniendo separadas las responsabilidades de los distintos componentes internos.

Agentes

Agrupa a las entidades activas que intervienen en el sistema eléctrico modelado, abarcando tanto a las unidades generadoras como a las demandas. Esta clase se define como abstracta y establece una interfaz común que deben implementar sus subclases, lo que permite una representación homogénea, extensible y coherente de los distintos actores del sistema. En la implementación actual del proyecto, se han desarrollado dos especializaciones concretas: `Generación`, que modela agentes generadores, y `Demanda`, que representa agentes de demanda.

Cada agente dispone de una función que recibe como argumento una instancia de la clase `Problema`. Mediante esta función, el agente actualiza el objeto, incorporando sus contribuciones operativas: variables, restricciones, términos binarios y aportes a la función objetivo. Esta interacción ocurre de manera dinámica cuando el `SystemController` lo solicita, especialmente durante las etapas de grabación —en la que el modelo construido se exporta a un formato estándar de programación matemática, como `.lp` o `.mps`, ampliamente utilizados por solvers comerciales y de código abierto— y resolución —donde el solver procesa el archivo y ejecuta el algoritmo de optimización—, cuya descripción detallada se presenta más adelante.

Adicionalmente, cada agente posee un identificador global y un identificador específico según su tipo (generación o demanda), lo cual permite distinguir su rol funcional dentro del sistema y habilita la construcción agregada de restricciones comunes. Esta distinción será especialmente relevante al momento de definir el despacho conjunto de generación y demanda.

El sistema base desarrollado incluye una instancia de `RioNegro` como agente de generación hidráulica, y una instancia de `DemandaFija` que representa la demanda residual conocida en cada instante temporal.

Despacho

La responsabilidad principal de esta clase es la construcción de las restricciones que articulan la interacción entre los agentes de generación y demanda. A través de este componente se definen las condiciones que regulan el equilibrio entre la oferta y la demanda de energía, las cuales se integran posteriormente al modelo de optimización global.

Cuando se indica a `SystemController` que configure un despacho específico, este le comunica a la clase `Despacho` la cantidad de unidades generadoras y los agentes de demanda activos en el sistema. Esta distinción resulta fundamental para la formulación del modelo:

- **Generación:** cada agente generador incorpora una variable de decisión $g_i(t)$ asociada a su producción en cada período. Por tanto, el número de unidades generadoras determina directamente la cantidad de variables

de este tipo en el modelo.

- ➢ **Demanda:** cada agente de demanda aporta un vector temporal predefinido. Estos valores actúan únicamente como parámetros: el despacho los consulta y suma en cada período, sin introducir nuevas variables en la formulación.

De esta manera, la estructura del modelo permanece fija: aumentan las variables únicamente cuando se incorporan nuevos generadores, mientras que el número de agentes de demanda afecta únicamente los valores paramétricos de entrada.

Formalmente:

$$g(t) = \sum_{i=1}^{n} g_i(t), \qquad d(t) = \sum_{j=1}^{m} d_j(t)$$

donde cada $g_i(t)$ es una variable de decisión de generación, mientras que cada $d_j(t)$ corresponde a la entrada del vector de demanda del agente j en el período t.

De esta manera, el módulo `Despacho` permite preservar de forma estructurada restricciones fundamentales del modelo MIP, como la condición (i), que establece que la generación total debe ser menor o igual que la demanda total en cada período.

De forma análoga al comportamiento de los agentes, el módulo `Despacho` contribuye al modelo de optimización actualizando la instancia de la clase `Problema` con las restricciones que le son propias. El `SystemController` solicita estas actualizaciones de manera dinámica durante la construcción del modelo global, en las etapas de grabación o resolución, y `Despacho` incorpora las contribuciones correspondientes al modelo siguiendo la misma dinámica que los agentes.

Actualmente, el sistema cuenta con una implementación inicial denominada `DespachoFijo`, pensada para escenarios en los que la penalización por demanda no satisfecha se modela a través de un costo fijo por unidad de energía no entregada. Esta implementación respeta la lógica del modelo MIP original y establece un criterio simple pero efectivo para evaluar la asignación de generación.

Cabe destacar que la arquitectura modular adoptada permite extender este módulo con nuevas estrategias de despacho como se mostrará en la Subsección 4.3.5.

Problema

Esta clase constituye el núcleo de la formulación computacional del modelo de optimización. Su función principal es consolidar las aportaciones de los distintos módulos del sistema —agentes y despacho— en una estructura unificada que representa el problema completo a resolver. Para ello, proporciona una interfaz que permite añadir variables, restricciones, términos binarios y contribuciones

a la función objetivo, los cuales, como se describió anteriormente, se incorporan de manera dinámica durante las etapas de construcción o resolución del modelo.

Esta clase cuenta con dos operaciones centrales: `grabar` y `resolver`. El método `grabar` permite generar un archivo especificando la ruta de destino y el tipo de formato deseado, de modo que el modelo pueda ser interpretado por diversos solvers externos. Por su parte, el método `resolver` ejecuta el proceso de optimización utilizando el solver seleccionado por el usuario, imprime los resultados en pantalla y almacena la solución obtenida. De esta manera, `Problema` no solo construye el modelo, sino que también conserva los valores óptimos de todas las variables, los cuales son utilizados posteriormente en el proceso de posprocesamiento, donde se genera la solución final del sistema.

Para mantener la independencia respecto del solver utilizado, la clase `Problema` se especializa en distintas subclases concretas, cada una asociada a un solver específico: `ProblemaLp`, `ProblemaCbc`, `ProblemaCplex` y `ProblemaGurobi`. Esta organización permite conservar una interfaz común y un flujo de trabajo uniforme, mientras se delega a cada subclase la traducción e interacción concreta con el motor de optimización correspondiente.

La clase `ProblemaLp` cumple un rol exclusivamente descriptivo: permite generar archivos en formato `.lp`, sin ofrecer funcionalidades de resolución. Su utilidad principal radica en facilitar la inspección del modelo formulado o su utilización en entornos externos.

Las restantes subclases —`ProblemaCbc`, `ProblemaCplex` y `ProblemaGurobi`— encapsulan la lógica necesaria para interactuar directamente con las interfaces nativas de cada solver. Para ello, traducen internamente la representación abstracta del modelo a un formato compatible con el solver correspondiente, ya sea para almacenarlo en disco o para ejecutarlo directamente. La responsabilidad de generar los archivos específicos o resolver el modelo recae en cada una de estas subclases, de acuerdo con las capacidades del solver subyacente.

Los resultados permiten evaluar el rendimiento relativo de los solvers bajo condiciones homogéneas de ejecución. Como se muestra en la Tabla 4.2, los solvers comerciales `Cplex` y `Gurobi` obtienen los mejores tiempos de resolución. En términos de desempeño promedio, `Cplex` presenta una ligera ventaja, mientras que `Gurobi` alcanza tiempos mínimos marginalmente menores, con diferencias reducidas en los tiempos máximos observados. Estas variaciones sugieren que ambos solvers son altamente competitivos para este tipo de problemas.

El solver `Cbc`, de código abierto, exhibe tiempos superiores en relación con las alternativas comerciales, aunque continúa siendo una opción sólida en contextos donde la accesibilidad y la ausencia de costos de licenciamiento son factores relevantes.

Cabe señalar que tanto `Cplex` como `Gurobi` ofrecen licencias académicas gratuitas, lo que facilita su utilización en entornos universitarios y de investigación, sin perjuicio de las ventajas que implica disponer también de alternativas abiertas como `Cbc`.

Finalmente, las clases auxiliares —*Variable*, *Binario*, *Restricción* y *AporteFuncionObjetivo*— complementan el funcionamiento de `Problema`, proporcionando los mecanismos necesarios para almacenar y manipular los elementos básicos del problema.

4.3.3. Flujo de uso del sistema

El sistema desarrollado sigue un flujo de ejecución interactivo mediante una interfaz de comandos, diseñado para guiar al usuario a lo largo de las distintas etapas del proceso: inicialización del entorno, configuración de agentes, creación de despachos, selección de solver y grabación o resolución de problemas. La Figura 4.3 ilustra gráficamente estas etapas del flujo operativo, mientras que la Tabla 4.3 presenta un resumen de los comandos disponibles, indicando su sintaxis y la función específica que cumplen dentro del ciclo de operación.

Estructura general de la interacción
La interacción con el sistema se realiza exclusivamente a través de una interfaz de línea de comandos (CLI, por sus siglas en inglés), lo que permite una operación directa, sin intermediarios gráficos. Cada comando ingresado es gestionado por la función central `procesarComando`, que desempeña el rol de coordinador principal: esta función analiza la entrada textual, verifica su validez sintáctica, y canaliza la ejecución hacia el método correspondiente del controlador general del sistema (`SystemController`).

Este enfoque modular, orientado a comandos, asegura un control estructurado del ciclo de vida del sistema, previniendo ejecuciones fuera de contexto y garantizando una transición segura y lógica entre las distintas fases: desde la inicialización hasta la resolución del modelo.

Estados operativos
El sistema funciona bajo un esquema de doble estado claramente definido:

- **Estado no inicializado:** fase previa a la activación, en la que únicamente se permite definir el horizonte temporal del modelo.
- **Estado inicializado:** una vez definido el horizonte, se habilita el acceso a todas las funcionalidades, incluyendo la incorporación de agentes, la configuración de parámetros del despacho y la ejecución del solver.

Representación esquemática del flujo
El flujo operativo del sistema está representado gráficamente en la Figura 4.3, que ilustra las distintas etapas y su relación secuencial. Como se aprecia, el flujo inicia con la definición del horizonte temporal, seguido de la incorporación de

agentes. El sistema exige que al menos un agente de generación y al menos un agente de demanda hayan sido registrado para poder avanzar hacia la creación de un despacho. Posteriormente, se configura el solver de optimización y, finalmente, el usuario puede optar por resolver el modelo y/o guardar el modelo creado en un formato como `.lp` o `.mps` antes de finalizar la sesión.

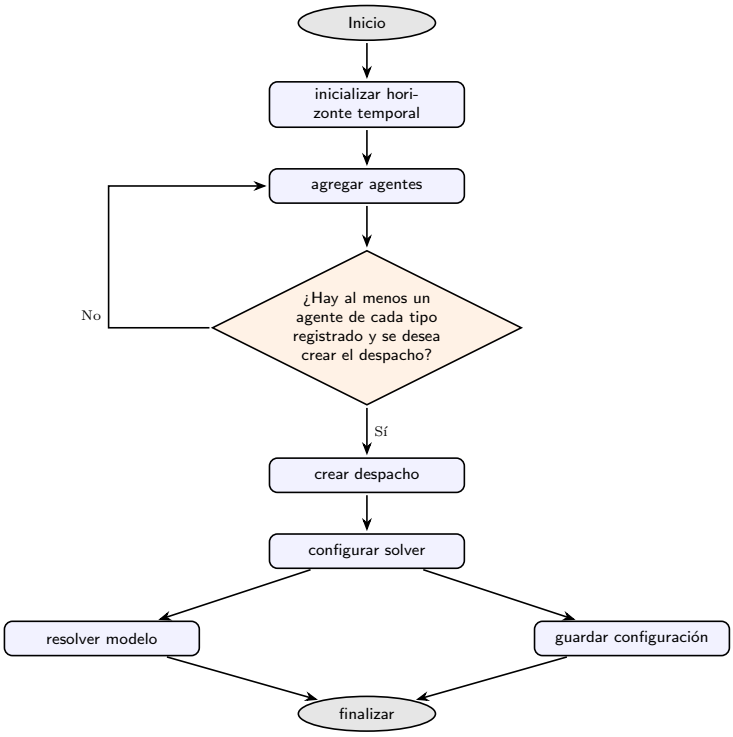

Figura 4.3: Flujo lógico normal de uso del sistema.

Comandos disponibles y organización funcional

El sistema expone un conjunto compacto y jerarquizado de comandos, que constituyen la interfaz de control de alto nivel. Cada instrucción corresponde a una operación autónoma y específica, diseñada para ser ejecutada en un momento determinado del flujo operativo. Esta organización funcional contribuye a preservar la coherencia del sistema, evitando la ejecución de acciones fuera de orden o en contextos inadecuados.

La Tabla 4.3 presenta un resumen detallado de los comandos actualmente disponibles, junto con una descripción clara de su propósito y uso:

Tabla 4.3: Resumen de comandos del sistema

Comando	Descripción
`inicializar --t <n>`	Define el horizonte temporal y activa el sistema.
`agregarAgente --RioNegro`	Crea el agente hidráulico compuesto por los embalses Bonete, Baygorria y Palmar.
`agregarAgente --DemandaFija`	Crea el agente de demanda interna con una secuencia de demandas fijas.
`crearDespacho --fija`	Define un despacho con un costo de falla asociado.
`configurarSolver --Gurobi|--Cbc|--Cplex`	Configura el tipo de solver de optimización a utilizar.
`grabar <ruta>`	Guarda la configuración actual del modelo en disco.
`resolver [--noConstante] <ruta.csv>`	Ejecuta el modelo y exporta los resultados a un archivo externo.
`finalizar`	Libera recursos internos y permite reconfigurar el sistema desde cero.
`salir`	Termina la ejecución del programa y libera todos los recursos utilizados.

El sistema ha sido diseñado para interpretar y ejecutar comandos de manera secuencial. A continuación, se describe el comportamiento interno del sistema ante la ejecución de cada uno de los comandos definidos.

Cuando se ejecuta el comando `inicializar --t <n>`, el valor del horizonte temporal especificado es almacenado dentro del objeto central `SystemController`, el cual registra este parámetro como condición previa indispensable para habilitar las siguientes etapas del flujo. Este paso marca el inicio del estado *inicializado* del sistema.

En los comandos `agregarAgente --RioNegro` y `agregarAgente --DemandaFija`, el sistema interpreta la instrucción como una solicitud para crear una nueva instancia de agente. En ambos casos, los datos necesarios para su construcción —como identificadores, parámetros y tipo— se envían al `SystemController`, que se encarga de instanciar el agente correspondiente, almacenarlo internamente y asignarle dos identificadores únicos: uno que identifica al agente en general y otro que lo distingue dentro de su tipo (por ejemplo, entre los distintos generadores o demandas)

La ejecución del comando `crearDespachoFijo` indica al sistema la necesidad de crear una instancia del tipo `DespachoFijo`. Esta operación también es gestionada por el `SystemController`, que almacena la instancia creada y la asocia a la colección de elementos activos en la planificación.

Cuando el usuario invoca `configurarSolver`, el sistema identifica la opción seleccionada a través del parámetro `--Lp`, `--Gurobi`, `--Cbc` o `--Cplex`, y procede a instanciar el objeto correspondiente de tipo `ProblemaLp`, `ProblemaGurobi`, `ProblemaCbc` o `ProblemaCplex`. La instancia generada es registrada en el `SystemController` como el solver actualmente activo para el sistema.

Respecto al comando `grabar <ruta>`, su ejecución activa el proceso de serialización del modelo. Esta acción es delegada por el `SystemController` al solver previamente configurado, el cual implementa los métodos necesarios para guardar la configuración actual del modelo en el archivo de salida indicado.

El comando `resolver` constituye una de las operaciones más críticas del sis-

tema. Al ejecutarse, inicia el proceso de resolución del MIP mediante el solver seleccionado. La solución obtenida no se utiliza de manera inmediata, sino que se almacena temporalmente dentro de la clase correspondiente al problema y, adicionalmente, se exporta a un archivo en formato `.csv`.

A continuación, el `SystemController` transmite esta solución a los agentes y al módulo de despacho, quienes aplican un proceso de posprocesamiento específico. Durante esta etapa, se actualizan las variables internas de cada componente del sistema de acuerdo con los valores óptimos obtenidos. Posteriormente, el sistema genera un segundo archivo `.csv` con los valores resultantes del posprocesamiento, asegurando trazabilidad entre la solución obtenida por el solver y su interpretación operativa.

Finalmente, `SystemController` calcula el valor de la función objetivo posterior al posprocesamiento.

En cuanto al comando `finalizar`, su propósito es liberar los recursos asociados a la configuración actual del modelo, permitiendo que el usuario pueda reiniciar el flujo con un nuevo horizonte temporal u otros parámetros, sin necesidad de cerrar el programa. Esta función restablece internamente el estado del `SystemController`, descartando los agentes, despachos y solver previamente definidos, pero manteniendo activa la sesión de trabajo.

Por último, el comando `salir` ejecuta la terminación completa del entorno de ejecución. A diferencia de `finalizar`, esta instrucción no sólo elimina los objetos definidos durante la sesión, sino que también cierra de forma definitiva el programa, liberando todos los recursos del sistema operativo y concluyendo la ejecución del software.

4.3.4. Extensión del sistema: incorporación de una unidad generadora térmica

En esta sección se describe el proceso de extensión del modelo mediante la incorporación de una unidad térmica de arranque rápido, con el objetivo de evaluar en la práctica el nivel de modularidad y extensibilidad alcanzado por la arquitectura implementada.

Las unidades térmicas rápidas constituyen una categoría de generación de respaldo que puede activarse en plazos muy breves, lo que las hace especialmente útiles para compensar fluctuaciones en la generación renovable o atender picos de demanda. Estas unidades fueron formalizadas en (Risso y cols., 2025) como submodelos de Programación Entera Mixta dentro de la formulación general del despacho energético nacional, y su comportamiento fue adoptado como referencia para la extensión aquí presentada.

Se detalla a continuación el modelo y posteriormente las modificaciones en el código para contemplar la incorporación de una unidad térmica rápida.

Modelo matemático
El comportamiento operativo de una unidad térmica rápida se representa mediante el siguiente modelo de Programación Entera Mixta:

$$\begin{cases} \min\limits_{x_t,\, y_t,\, w_t} \left(a \sum\limits_{t \in T} x_t + b \sum\limits_{t \in T} w_t + \alpha \sum\limits_{t \in T} y_t \right) & \\ m_{GT} \cdot x_t \leq w_t, & \text{(i)} \\ w_t \leq M_{GT} \cdot x_t, & \text{(ii)} \\ y_t \geq x_t - x_{t-1}, & \text{(iii)} \\ 2x_t - 2x_{t+1} + x_{t+2} + x_{t+3} \geq 0, & \text{(iv)} \\ 2x_t - 2x_{t+1} + x_{t+2} + x_{t+3} \leq 2, & \text{(v)} \\ x_t,\, y_t \in \{0, 1\}, \quad w_t \geq 0 & \text{(vi)} \end{cases}$$

donde:

Tabla 4.4: Parámetros y variables del modelo térmico

Parámetros		Variables	
a	Costo fijo de operación por encendido de la unidad [US$/encendido].	x_t	Variable binaria: 1 si la unidad está encendida en t, 0 en caso contrario .
b	Costo variable por unidad de energía generada [US$/MWh].	w_t	Energía generada por la unidad en el período t [MWh].
α	Costo de arranque por cambio de estado (apagado → encendido) [US$/arranque].	y_t	Variable binaria que indica encendido de la unidad en el período t [1hr].
m_{GT}	Potencia mínima técnica cuando la unidad está encendida [MW].		
M_{GT}	Potencia máxima técnica de la unidad [MW].		
T	Conjunto de períodos del horizonte temporal [1hr].		

Las restricciones (i) y (ii) aseguran que la potencia generada se mantenga dentro de los límites técnicos únicamente cuando la unidad está encendida. La restricción (iii) identifica los instantes de arranque, mientras que (iv) garantiza que, una vez encendida, la unidad térmica permanezca operativa durante al menos tres horas. Por último, la restricción (v) impone que, si la unidad se apaga, no pueda volver a encenderse antes de transcurrido ese mismo período mínimo de tres horas.

Modificaciones del código
Para incorporar este agente, se extendió la clase `Generación` mediante la crea-

ción de una subclase denominada `UnidadTermicaRapida`. Esta subclase implementa una función de contribución al modelo que recibe como parámetro una instancia de la clase `Problema`, sobre la cual se definen las variables x_t, y_t y w_t, las restricciones (i)–(v), y los términos correspondientes a la función objetivo.

Finalmente, se añadió en la clase `SystemController` un método específico para la creación e incorporación de este nuevo agente, de modo que la unidad térmica pueda añadirse al conjunto de agentes activos y participar en el proceso de despacho conjunto con las unidades hidroeléctricas. Esta extensión fue incorporada sin necesidad de modificar el núcleo del sistema, lo cual valida empíricamente la flexibilidad y capacidad de expansión de la arquitectura implementada.

4.3.5. Extensión del sistema: nuevo escenario

Se extiende el modelo MIP original con el objetivo de incorporar dos modificaciones propuestas en un *paper* actualmente en proceso de revisión (Risso y cols., 2025).

En dicho trabajo se introducen dos extensiones principales:

1. Incorporación de un costo no lineal asociado al agua del embalse de Bonete.
2. Incorporación de un costo de falla lineal proporcional a la demanda total del sistema.

A continuación, se describe la primera de estas extensiones, correspondiente al modelado del costo no lineal del agua en Bonete. La segunda extensión —relativa a la incorporación de un costo de falla proporcional a la demanda total— se presenta posteriormente en este documento.

Incorporación de un costo no lineal del agua en Bonete

El modelo base desarrollado para el despacho incluye un costo de oportunidad lineal asociado al volumen de agua almacenado en el embalse de Bonete, expresado como $ca_{1h}(v_{1h,0} - v_{1h,T})$. Tal como se detalla en la Tabla 4.1, se tiene que:

- $ca_{1,h}$ es el costo del agua en la central i medido en USD/m^2.
- $v_{1h,0}$ es el volumen inicial del embalse en Bonete medido en m^3.
- $v_{1h,T}$ es el volumen final del embalse en Bonete medido en m^3.

Si bien esta aproximación es aceptable para horizontes de planificación cortos, su validez se ve limitada cuando el período de planificación se extiende y las fluctuaciones del nivel del embalse se tornan más relevantes. En tales escenarios, una representación lineal tiende a subestimar o sobreestimar el verdadero valor marginal del recurso hídrico, afectando la fidelidad del modelo.

Dado que el embalse de Bonete constituye la principal reserva energética del sistema interconectado uruguayo, su tratamiento adecuado dentro del modelo resulta fundamental. En particular, incorporar una función de costo del agua no lineal permite capturar de forma más precisa las externalidades relacionadas con la utilización del recurso, reflejando mejor su impacto en la generación y en la seguridad del suministro eléctrico.

Se describirá los cambios en el modelo matemático a continuación y posteriormente los cambios en el código que realizamos en el código para contemplarlo.

Modelo matemático
En el modelo se sustituye el término lineal $ca_{1h}(v_{1h,0} - v_{1h,T})$ por una variable δ_{1h} que representa la diferencia entre los valores de Bellman asociados al volumen inicial y final, es decir, por lo que:

$$\delta_{1h} = Blm_{1h}(v_{1h,0}) - Blm_{1h}(v_{1h,T})$$

donde $\mathrm{Blm}_{1h}(v)$ representa el valor acumulado del agua hasta el volumen v según la función no lineal interpolada.

Como $\mathrm{Blm}_{1h}(v)$ es una función no lineal y cóncava, su diferencia respecto del volumen inicial,

$$\Delta\mathrm{Blm}_{1h}(v) \;=\; \mathrm{Blm}_{1h}(v_{1h,0}) - \mathrm{Blm}_{1h}(v),$$

es convexa. Para mantener la linealidad global del modelo, dicha función convexa se aproxima mediante el máximo de tres tangentes lineales, construidas alrededor del volumen inicial y de variaciones del $\pm 15\,\%$ respecto del mismo.

Esto se implementa mediante las siguientes restricciones:

$$\begin{cases} \delta_{1h} \;\geq\; p_{1h}^{(1)} - q_{1h}^{(1)} \left(v_{1h,T} - 0.85\, v_{1h,0}\right), \\ \delta_{1h} \;\geq\; -\; q_{1h}^{(2)} \left(v_{1h,T} - v_{1h,0}\right), \\ \delta_{1h} \;\geq\; p_{1h}^{(3)} - q_{1h}^{(3)} \left(v_{1h,T} - 1.15\, v_{1h,0}\right). \end{cases} \tag{4.11}$$

donde los parámetros $p_{1h}^{(k)}$ y $q_{1h}^{(k)}$ provienen de la interpolación de la función de Bellman (derivadas y valores en puntos específicos). Estas restricciones aseguran que δ_{1h} represente correctamente la envolvente convexa local de $\Delta\mathrm{Blm}_{1h}(v)$.

Modificaciones del código
Para incorporar el nuevo esquema de valoración no lineal del agua, se extendió la implementación mediante la creación de una nueva clase denominada `RioNegroNoLineal`. Esta clase hereda de `RioNegro` y redefine los dos aspectos fundamentales del comportamiento del agente: la generación de restricciones y el aporte a la función objetivo.

En primer lugar, se modifco la función `generarRestriccionesBonete`. Además de llamar a las restricciones definidas en la clase base, esta versión ampliada incorpora explícitamente las nuevas restricciones que modelan la no linealidad del embalse, presentadas en (4.11). Este enfoque permite mantener la estructura original del modelo y, al mismo tiempo, añadir la lógica adicional necesaria para capturar el comportamiento más realista del recurso hídrico.

En segundo lugar, se modificó la función `generarAporteFuncionObjetivo`, encargada de definir los términos de costo de oportunidad asociados a los embalses de Bonete y Palmar en la función objetivo del modelo. En esta actualización se mantuvo sin cambios el tratamiento correspondiente a Palmar, mientras que el término lineal aplicado a Bonete fue reemplazado por la variable δ_{1h}, tal como se explicó previamente. De este modo, la función objetivo refleja adecuadamente la nueva formulación basada en la diferencia entre los valores de Bellman interpolados.

Finalmente, se incorporó un nuevo comando en el controlador principal del sistema (`SystemController`), permitiendo la creación explícita de agentes del tipo `RioNegroNoLineal` mediante un comando. Esta incorporación fue diseñada para mantener la coherencia con el flujo de comandos existente, sin introducir cambios estructurales en el resto del sistema.

En resumen, la extensión implementada no implicó cambios profundos en la estructura del sistema, pero permitió incorporar un esquema de valoración no lineal del agua que mejora la representación del comportamiento del embalse. Con ello, el modelo refleja de manera más adecuada las decisiones operativas y de planificación asociadas a distintos niveles de almacenamiento en Bonete.

Incorporación de un costo de falla lineal proporcional a la demanda total del sistema.

En el modelo base detallado en la Sección 4.1 se tiene un costo de falla dado por la siguiente expresión:

$$CF \sum_{i=1}^{T} (d_t - g_{h,t})$$

donde según se detalla en la Tabla 4.1, se tiene que:

- CF es la penalización por demanda no satisfecha.
- d_t es la demanda de energía en la hora t medido en MW.
- $g_{h,t}$ es la energía generada por la central i en la hora t medido en MW.

Sin embargo, según una normativa del 2023 se define un costo de falla por rangos lo que provoca que el costo de falla ya no es una recta, sino una función *piecewise*, que "salta" de un valor a otro. Por ejemplo, esto podría describirse a partir de la siguiente tabla:

Severity	min [MWh]	max [MWh]	Cost [USD/MWh]	Failure [MWh] per Range	Subtotal [USD] por Range
Range 1	0	20	370	20	7400
Range 2	20	70	600	50	30000
Range 3	70	145	2400	30	72000
Range 4	145	1000	4000	0	0
Totals				100	109400

Figura 4.4: Ejemplo de cómo aplicar la regulación de costos de falla en una hora, con una demanda total de 1000 MWh y una falla de 100 MWh. Esta tabla fue extraída de (Risso y cols., 2025).

La incorporación de esta consideración en el modelo se detalla a continuación y a posterior los cambios en el código.

Modelo matemático
En esta nueva versión el costo de falla es reemplazado por una nueva variable CFT definida por:

$$CFT = \sum_{t=1}^{T} f_t \tag{4.12}$$

Además se deben incorporar las siguientes restricciones:

$$\begin{cases} f_t \geq \text{máx}\{0,\ \hat{q}_{1f}^{(1)}(d_t - g_{h,t})\} & \text{(i)} \\ f_t \geq \hat{p}_{1f}^{(2)} + \hat{q}_{1f}^{(2)}(d_t - g_{h,t}) & \text{(ii)} \\ f_t \geq \hat{p}_{1f}^{(3)} + \hat{q}_{1f}^{(3)}(d_t - g_{h,t}) & \text{(iii)} \\ f_t \geq \hat{p}_{1f}^{(4)} + \hat{q}_{1f}^{(4)}(d_t - g_{h,t}) & \text{(iv)} \end{cases} \tag{4.13}$$

Estos parámetros $\hat{q}_{1f}^{(i)}$ adoptan los valores definidos en la progresión de penalidades establecida por la normativa. Por ejemplo, si dicha normativa fuera la mostrada en la Figura 4.4, se tendría $\hat{q}_{1f}^{(1)} = 370$ USD/MWh, $\hat{q}_{1f}^{(2)} = 600$ USD/MWh, etc.

Por otro lado, los valores de $\hat{p}_{1f}^{(i)}$ se determinan de manera de garantizar la continuidad entre los distintos tramos de severidad.

Modificaciones del código
Se implementaron dos enfoques para incorporar lo expuesto anteriormente. El primero consiste en integrar directamente en el código la formulación matemática desarrollada. El segundo se apoya en ciertas observaciones que permiten apartarse parcialmente de dicha formulación y adoptar una implementación alternativa.

En relación con el primer enfoque, se amplió el modelo mediante la creación de una nueva clase denominada `DespachoProporcional`. Esta clase hereda de `Despacho` y redefine dos aspectos fundamentales del comportamiento del agente: la generación de restricciones y su aporte a la función objetivo.

En primer lugar, se modificó la función `generarRestricciones`. Además de invocar las restricciones definidas en la clase base, esta versión incorpora explícitamente las nuevas restricciones que modelan el costo de falla por tramos, presentadas en (4.13). Para cada período t del horizonte temporal, se añaden las cuatro restricciones que definen la envolvente convexa de dicha función de costo.

En segundo lugar, se ajustó la función `generarAporteFuncionObjetivo`, encargada de determinar los términos de penalización por demanda no satisfecha en la función objetivo del modelo. En este caso, no se consideran los aportes definidos en la clase base, sino que se especifica explícitamente el término correspondiente descrito en la ecuación (4.12).

Finalmente, se incorporó un nuevo comando en el controlador principal del sistema (`SystemController`), que habilita la creación explícita de agentes del tipo `DespachoProporcional` mediante un comando dedicado.

Por otro lado, el segundo enfoque se basa en la observación de que, para el costo de falla, es posible trabajar únicamente con dos valores y prescindir completamente de la construcción de los tramos. En la práctica, las fallas que superan el segundo tramo son prácticamente inexistentes, por lo que esta aproximación resulta plenamente justificable.

La idea consiste en trabajar con la demanda total promedio del escenario. Para los valores inferiores a dicho promedio, el aporte al costo de falla es lo suficientemente pequeño como para suponer que, de ocurrir, la falla ingresaría directamente en el segundo tramo; por lo tanto, se utiliza el coeficiente asociado a ese tramo. Para los valores superiores al promedio, en cambio, se emplea el coeficiente del primer tramo.

Más formalmente, en la expresión asociada al costo de falla en (4.12), en lugar de introducir la sumatoria completa de los términos f_t, el aporte correspondiente al período t se calcula de la siguiente manera:

$$\begin{cases} \hat{q}_{1f}^{(1)}(d_t - g_{h,t}) & \text{si } d_t \geq \overline{d} \\ \hat{q}_{1f}^{(2)}(d_t - g_{h,t}) & \text{si } d_t < \overline{d}. \end{cases} \tag{4.14}$$

Obsérvese que, si se empleara un único coeficiente CF —es decir, si $\hat{q}_{1f}^{(1)} = \hat{q}_{1f}^{(2)} = CF$—, el modelo colapsaría nuevamente en la estructura del modelo base, en la cual el costo de falla se representaba mediante un único parámetro.

La implementación de este enfoque se llevó a cabo mediante la creación de una nueva clase denominada `DespachoProporcionalAlternativo`, que extiende a `Despacho`, de forma análoga a `DespachoProporcional`, pero redefiniendo únicamente la función `generarAporteFuncionObjetivo`. Nótese que la función `generarRestricciones` no se redefine, dado que en este enfoque no se incorporan las restricciones de la formulación por tramos presentadas en (4.13), ya que —como se explicó previamente— se prescinde de la construcción explícita de dichos tramos.

Esta clase recibe como parámetro la demanda total en cada período d_t. De este modo, en `generarAporteFuncionObjetivo` se calcula $\overline{d}$ y se añaden los términos detallados en (4.14) a la función objetivo.

4.3.6. Síntesis del diseño e implementación

El sistema desarrollado para el despacho hidroeléctrico se organizó siguiendo una arquitectura modular, orientada a objetos y programada en C++. Esta estructura permite abstraer las distintas funcionalidades en componentes independientes e interoperables, facilitando su mantenimiento y ampliación futura.

Funcionalmente, el sistema opera como una herramienta basada en comandos que guía al usuario a través de todas las etapas del flujo de ejecución: inicialización, configuración de agentes, definición de esquemas de despacho, selección del solver y resolución del modelo. La operación se soporta en una lógica de estados diferenciada —no inicializado e inicializado— que asegura consistencia y evita errores de contexto.

En la implementación, la clase `SystemController` centraliza la gestión de agentes, la configuración del entorno y la coordinación de la ejecución. La función `procesarComando` interpreta las órdenes del usuario y delega la acción correspondiente a los métodos del controlador.

Cada agente, ya sea generador o demandante, se modeló como una subclase especializada de una clase abstracta común, lo que permite encapsular comportamientos específicos manteniendo una interfaz unificada. El módulo `Despacho` define las restricciones de balance y las condiciones que relacionan generación con demanda, apoyándose en una formulación algebraica agregada.

La clase `Problema` constituye el núcleo de la formulación matemática, integrando variables, restricciones y función objetivo definidas por agentes y despacho. Para adaptarse a distintos motores de optimización —CBC, CPLEX y Gurobi— dispone de subclases específicas que preservan una interfaz común.

El sistema se validó resolviendo instancias conocidas, demostrando resultados consistentes. Además, su diseño modular permitió extenderlo sin modificar el núcleo: se incorporó una unidad térmica de arranque rápido y un esquema no

lineal de costo del agua para el embalse de Bonete, mostrando la flexibilidad del enfoque adoptado.

En conjunto, el sistema proporciona una base sólida y flexible para el modelado y resolución del problema de Coordinación Hidrotérmica de Corto Plazo, con capacidad de adaptación a futuras extensiones y líneas de investigación aplicada, cómo se verá con más detalle en el Capítulo 6 (*Conclusiones y trabajo futuro*).

Capítulo 5

Experimentación

Este capítulo presenta el análisis empírico del desempeño del sistema en dos etapas diferenciadas. En primer lugar, se evalúa el comportamiento del modelo MIP original. A continuación, se estudia el desempeño del sistema extendido, el cual incorpora un costo no lineal del agua en Bonete y un costo de falla proporcional a la demanda total. En ambos casos se adopta la misma metodología experimental con el fin de permitir una comparación consistente.

5.1. Metodología

Para el escenario base se ejecutaron 204 instancias independientes, mientras que para el modelo extendido se resolvieron únicamente 5 instancias. En ambos casos se adoptó un criterio de optimalidad basado en un GAP relativo máximo de 0,1 %.

La calidad de las soluciones se verificó contrastándolas con resultados de referencia, observándose un error relativo menor al 1 % en todos los casos.

En cada ejecución se registraron cuatro componentes temporales:

1. **Tiempo de construcción del modelo interno**: creación de variables, restricciones y función objetivo dentro de la infraestructura propia del sistema.

2. **Tiempo de traducción al modelo del solver**: transformación del modelo interno al formato requerido por el solver, incluyendo la generación de estructuras auxiliares y su transferencia para la resolución.

3. **Tiempo de resolución**: proceso interno del solver hasta alcanzar el criterio de optimalidad.

4. **Tiempo de posprocesamiento**: extracción de resultados, validación de condiciones, reconstrucción de objetos de salida y almacenamiento de la información.

El tiempo total se define como la suma de los cuatro tiempos anteriores.

5.2. Análisis estadístico

Se calcularon métricas descriptivas para cada componente temporal y para el tiempo total:

- ➢ mínimo, máximo, promedio, desviación estándar;
- ➢ proporción relativa de cada componente sobre el tiempo total;
- ➢ distribución de tiempos mediante gráficos (boxplots/histogramas).

5.3. Entorno computacional

Las ejecuciones se realizaron en una estación de trabajo equipada con un procesador `Intel Core i7-12700KF` (12 núcleos, 20 hilos) a 3.6 GHz, 16 GB de memoria RAM y disco sólido (SSD). El sistema operativo utilizado fue `Windows 11 Pro` (64 bits).

5.4. Resultados experimentales

5.4.1. Modelo base

La siguiente tabla presenta las estadísticas descriptivas de los tiempos de ejecución registrados en las 204 instancias correspondientes al modelo base detallado en la Sección 4.1. Como se menciono anteriormente, cada ejecución se descompuso en cuatro etapas principales —construcción del modelo interno, traducción al formato del solver, resolución y posprocesamiento—, lo que permite aislar los componentes dominantes del tiempo total y evaluar su comportamiento estadístico.

Tabla 5.1: Estadísticas descriptivas de los componentes temporales (modelo base)

Componente	Mínimo (s)	Promedio (s)	Máximo (s)	Desv. Est. (s)	Proporción (%)
Construcción del modelo interno	0.012	0.013	0.016	0.0005	0.19
Traducción al modelo del solver	0.023	0.024	0.030	0.0008	0.34
Resolución (con Cplex)	0.376	20.708	60.108	23.180	81.20
Posprocesamiento	0.035	1.418	16.585	2.800	18.26
Tiempo total	1.634	22.162	61.344	22.536	—

Los resultados numéricos, complementados con la evidencia visual, permiten identificar diferencias claras entre las distintas etapas del modelo. A partir de

esta información, se procede a analizar las distribuciones individuales de cada componente.

En particular, la fase de resolución se identifica como la más exigente y, al mismo tiempo, la más variable, tanto en términos de magnitud como de dispersión. Su tiempo promedio se sitúa en torno a los 21 segundos, mientras que los valores observados oscilan entre 0,4 y 60 segundos, con una desviación estándar cercana a 23. La Figura 5.1 revela que esta variabilidad responde a un patrón bimodal, con dos modos claramente definidos: uno concentrado en tiempos de resolución muy bajos (aproximadamente entre 0 y 3 segundos) y otro asociado a instancias cuyo tiempo se aproxima al máximo registrado.

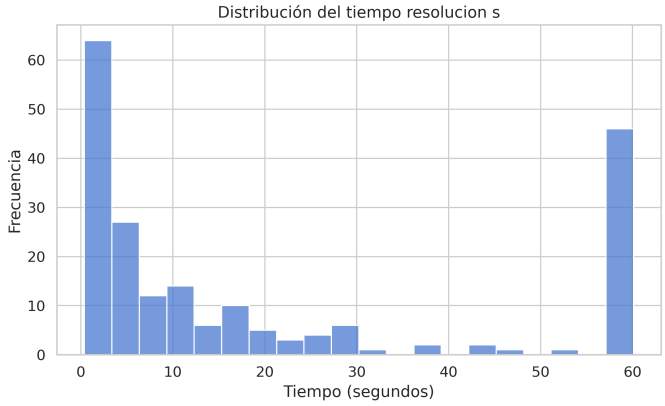

Figura 5.1: Histograma del tiempo de resolución. Se observa una distribución claramente bimodal, con dos modos bien definidos y de frecuencias comparables: uno concentrado en tiempos muy bajos (aproximadamente entre 0 y 3 segundos) y otro asociado a instancias con tiempos cercanos al máximo registrado (alrededor de 60 segundos).

Por su parte, la fase de posprocesamiento se posiciona como la segunda más demandante, tal como se evidencia en la Tabla 5.1. La Figura 5.2 muestra una distribución unimodal, caracterizada por una fuerte concentración de valores bajos y una cola derecha tenue asociada a algunos pocos casos de mayor duración. La mayoría de las observaciones se sitúa por debajo de los 2 segundos, lo que sugiere que el posprocesamiento presenta un comportamiento considerablemente más homogéneo y predecible en comparación con la fase de resolución.

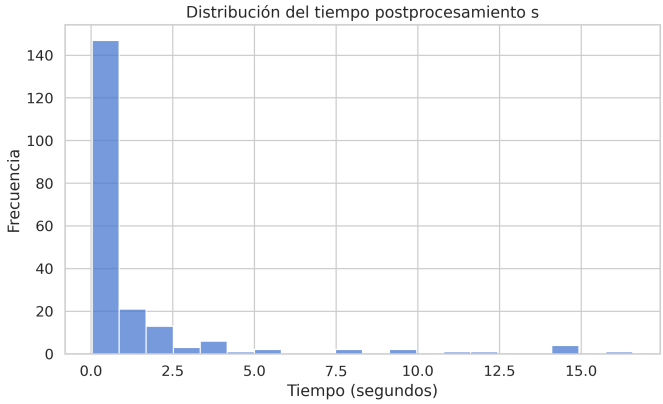

Figura 5.2: Histograma del tiempo de posprocesamiento. La distribución es unimodal, con una fuerte concentración de observaciones en valores bajos y una cola derecha tenue, asociada a algunos casos aislados de mayor duración.

Las fases iniciales —construcción del modelo interno y traducción al formato del solver— son las que menos tiempos conllevan como se puede observar en la Tabla 5.1.

En la Figura 5.3 y en la Figura 5.5 se puede observar que sus distribuciones son unimodales, con muy baja variabilidad, sin colas pronunciadas ni valores extremos significativos, lo que evidencia que estas etapas presentan un comportamiento prácticamente determinístico y no constituyen cuellos de botella relevantes en términos computacionales.

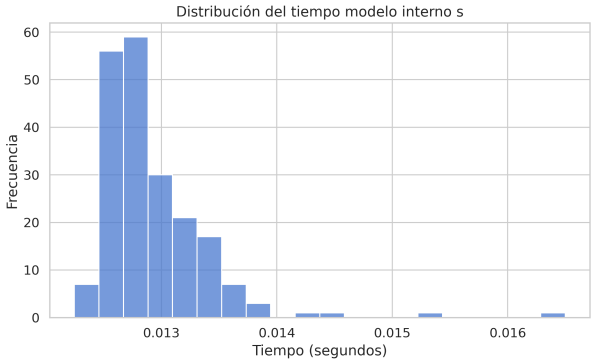

Figura 5.3: Histograma del tiempo de construcción del modelo interno. La distribución presenta un comportamiento unimodal altamente concentrado en torno a valores muy bajos, con escasa variabilidad y sin presencia de colas pronunciadas ni valores atípicos relevantes.

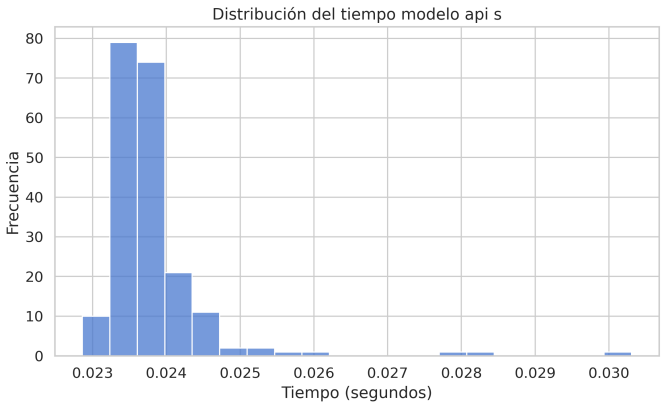

Figura 5.4: Histograma del tiempo de traducción al formato del solver. La distribución presenta un patrón unimodal altamente concentrado en valores muy bajos, con una dispersión mínima y sin presencia de colas pronunciadas ni valores extremos relevantes. Este comportamiento indica que la etapa de traducción posee un costo temporal prácticamente constante entre instancias, contribuyendo de forma marginal y estable al tiempo total de ejecución.

El análisis conjunto de los boxplots presentado en la Figura 5.5 refuerza esta interpretación: las etapas de construcción del modelo interno y traducción al solver muestran una dispersión mínima, con valores altamente concentrados. En contraste, la fase de resolución presenta una variabilidad marcadamente superior, lo que explica gran parte de la heterogeneidad observada en el tiempo total. El posprocesamiento, por su parte, exhibe un comportamiento intermedio, con mayor dispersión que las etapas iniciales pero sensiblemente menor que la de la resolución. La similitud entre la distribución del tiempo total y la del tiempo de resolución confirma que la variabilidad del desempeño global está dominada principalmente por el comportamiento del solver.

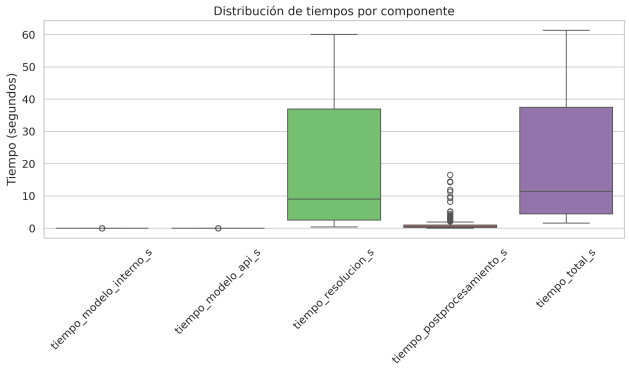

Figura 5.5: Dispersión de tiempos por componente mediante boxplots. Las etapas de construcción del modelo interno y traducción al solver presentan una variabilidad muy reducida, con valores concentrados en rangos estrechos. En contraste, la fase de resolución exhibe una dispersión considerablemente mayor, mientras que el posprocesamiento presenta un comportamiento intermedio.

Finalmente, la distribución del tiempo total refleja esta estructura mixta: un conjunto mayoritario de ejecuciones rápidas y un grupo no despreciable de instancias con tiempos cercanos al máximo, lo que resulta en una elevada dispersión total (desviación estándar de 22,5 s). Esto indica que, si bien el modelo base presenta un desempeño eficiente en promedio, su comportamiento no es homogéneo y depende fuertemente de la configuración particular de cada instancia.

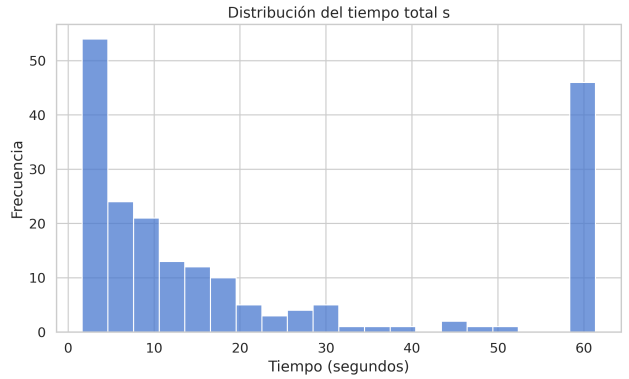

Figura 5.6: Distribución del tiempo total de ejecución. Se observa una estructura caracterizada por una alta concentración de ejecuciones rápidas y un subconjunto no despreciable de instancias con tiempos significativamente mayores, lo que se traduce en una dispersión elevada. Esta configuración refleja la coexistencia de dos regímenes diferenciados, inducidos principalmente por el comportamiento heterogéneo de la fase de resolución.

En este contexto, el análisis del modelo base permite concluir que el desempeño computacional del sistema está fuertemente condicionado por la etapa de resolución, tanto en términos de magnitud media como de variabilidad. Las restantes fases presentan un comportamiento estable y predecible, con una contribución temporal acotada y prácticamente constante entre instancias.

5.4.2. Modelo extendido

El análisis del modelo extendido se basa en la implementación alternativa detallada en la Subsección 4.3.5, considerada por razones de simplicidad. Su objetivo es validar empíricamente dos aspectos centrales del sistema: (i) que la incorporación de un costo no lineal del agua en Bonete y de un costo de falla proporcional a la demanda total no introduzca penalizaciones computacionales significativas; y (ii) que las soluciones obtenidas sean correctas y consistentes con la formulación teórica propuesta.

Dado que estas extensiones forman parte de desarrollos actualmente en proceso de revisión (Risso y cols., 2025), no se dispone de un conjunto amplio de instancias específicamente diseñado para esta versión del modelo. En consecuencia, el análisis se basa en un subconjunto reducido de cinco instancias representativas. Si bien este tamaño muestral limita el alcance estadístico de los resultados, resulta adecuado para una evaluación comparativa preliminar del impacto computacional de la extensión.

Selección de instancias y validación de rendimiento
Las cinco instancias analizadas se seleccionaron en función del comportamiento observado en el modelo base y se extendieron para el presente modelo, con el propósito de abarcar escenarios extremos junto con un caso intermedio representativo. En particular, se consideraron instancias asociadas a los tiempos mínimos y máximos de posprocesamiento, los tiempos mínimos y máximos de ejecución total, así como una instancia seleccionada de manera aleatoria.

En concreto, se incluyeron las instancias *04f* (menor tiempo de posprocesamiento), *28a* (mayor tiempo de posprocesamiento), *34e* (menor tiempo total), *44b* (mayor tiempo total) y *12b*, seleccionada de forma aleatoria.

Para cada una de estas instancias se resolvió tanto el modelo base como su versión extendida, registrándose los tiempos correspondientes a las cuatro etapas del proceso: construcción del modelo interno, traducción al formato del solver, resolución y posprocesamiento. Este enfoque permite aislar el efecto de la extensión sobre cada componente temporal y detectar posibles fuentes de sobrecarga.

Como ejemplo ilustrativo, la Tabla 5.2 presenta la comparación detallada entre el modelo base y el modelo extendido para las instancias mencionadas anteriormente:

Tabla 5.2: Comparación de tiempos de ejecución entre el modelo base y el modelo extendido

Instancia	Gen. interno (s)	Proc. API (s)	Resolución (s)	Post-proc. (s)	Total (s)
04f	0.016	0.031	26.775	0.031	26.854
04f-extended	0.016	0.027	10.029	0.035	10.107
28a	0.016	0.033	0.502	0.088	0.640
28a-extended	0.016	0.029	0.247	8.737	9.028
34e	0.016	0.028	1.334	0.032	1.410
34e-extended	0.102	0.179	21.804	0.342	22.426
44b	0.016	0.028	60.063	0.035	60.142
44b-extended	0.015	0.027	16.825	0.279	17.146
12b	0.016	0.031	60.067	0.029	60.143
12b-extended	0.016	0.028	3.072	26.711	29.827

El análisis comparativo de la Tabla 5.2 muestra que la incorporación de la extensión no introduce una sobrecarga significativa en las etapas bajo control directo de la implementación. En particular, los tiempos de construcción del modelo interno, traducción al solver y posprocesamiento permanecen acotados y comparables a los observados en el modelo base, incluso en las instancias extremas.

Las diferencias más relevantes se concentran en la fase de resolución, cuyo comportamiento depende del solver y de la estructura efectiva de cada instancia. En este sentido, no se observa una relación directa entre la complejidad computacional del modelo base y la del modelo extendido: instancias costosas de resolver bajo el modelo base pueden presentar reducciones sustanciales en el tiempo total al resolverse con el modelo extendido, y viceversa. Este resultado sugiere que la extensión modifica la estructura del problema MIP enfrentado por el solver, sin introducir penalizaciones sistemáticas en las etapas bajo control del modelo.

Validación de correctitud

La validación de la correctitud del modelo extendido se realizó mediante dos mecanismos complementarios: (i) la contrastación de las soluciones obtenidas con el tutor del modelo y (ii) la verificación de una propiedad de consistencia estructural que permite recuperar exactamente el modelo base como caso particular.

En particular, se analizó la formulación del costo de falla proporcional definida por la expresión por tramos presentada en (4.14), la cual introduce heterogeneidad en la penalización de las fallas en función de la magnitud relativa de la demanda. Como prueba de consistencia, se verificó que, cuando los coeficientes asociados a ambos tramos coinciden —esto es, cuando $\hat{q}1f^{(1)} = \hat{q}1f^{(2)} = CF$—, el modelo extendido colapsa exactamente en la estructura del modelo base. Bajo esta condición, se constató que las diferencias relativas en el valor óptimo de la función objetivo son inferiores al 1 %, lo que resulta consistente con la equivalencia teórica entre ambas formulaciones.

Este resultado confirma que, en ausencia de heterogeneidad en los coeficientes del costo de falla, el modelo extendido reproduce el comportamiento del modelo base. En consecuencia, las diferencias observadas en los escenarios generales pueden atribuirse exclusivamente a la nueva parametrización introducida, sin evidenciar inconsistencias lógicas ni numéricas en la implementación.

Capítulo 6

Conclusiones y Trabajo Futuro

El presente trabajo tuvo como objetivo principal el diseño, validación e implementación de una herramienta computacional para la resolución del problema de Coordinación Hidrotérmica de Corto Plazo, basada en un modelo de Programación Entera Mixta. A partir de una formulación desarrollada previamente en el entorno académico, se logró no solo replicar sus resultados mediante herramientas estándar, sino también ampliar significativamente su alcance mediante una arquitectura modular implementada en `C++`.

El sistema permite representar y resolver modelos del STHTC con distintas configuraciones de agentes, esquemas de despacho y solvers. Su diseño orientado a objetos facilita la interacción por parte de operadores y analistas, así como futuras modificaciones por desarrolladores. En particular, se evaluó su capacidad de extensión en la Subsección 4.3.4 y la Subsección 4.3.5 mediante dos incorporaciones concretas: la adición de una unidad generadora térmica rápida y la modificación del tratamiento del costo del agua en una unidad hidroeléctrica, introduciendo una función objetivo no lineal y considerando un costo de falla proporcional a la demanda total. Ambas modificaciones se implementaron sin alterar la integridad del sistema, lo que evidencia la consistencia de la arquitectura modular adoptada.

Este enfoque representa un aporte concreto al diseño de herramientas abiertas y eficientes para la planificación energética, capaces de adaptarse al dinamismo de la matriz eléctrica nacional y de ser utilizadas tanto en aplicaciones operativas como en contextos de investigación y desarrollo.

A partir de esta base consolidada, se abren múltiples líneas de trabajo futuro orientadas a ampliar tanto el alcance funcional como la profundidad metodológica del sistema. Una extensión natural consiste en la incorporación de

nuevos agentes de generación y/o demanda, tales como la represa hidroeléctrica de Salto Grande, así como la posibilidad de modelar transferencias de energía mediante contratos con países vecinos.

Asimismo, resulta pertinente considerar la inclusión de unidades de almacenamiento, como baterías, que por su naturaleza dual —capaces de inyectar o absorber energía— exigen una representación flexible dentro del modelo. Para ello, se proyecta el desarrollo de una subclase dentro de la clase `Agente`, posiblemente denominada `Mixta`, que integre atributos de generación y demanda. Esta ampliación implicará, a su vez, ajustes en el esquema de despacho y en el tratamiento de las restricciones agregadas, con el fin de preservar la coherencia y consistencia global del modelo.

Desde una perspectiva metodológica, una dirección particularmente prometedora es la integración del sistema con técnicas de aprendizaje automático. En este sentido, una funcionalidad clave es la capacidad de generar automáticamente grandes volúmenes de instancias del problema, lo que resulta fundamental para entrenar modelos basados en redes neuronales. La posibilidad de configurar distintas composiciones de agentes, perfiles de demanda u horizontes de planificación permite construir bases de datos representativas, útiles para aproximar soluciones del STHTC mediante métodos de inteligencia artificial.

Esta línea será profundizada por uno de los integrantes del equipo en el marco de su maestría, como parte de los avances recientes impulsados por el convenio entre UTE y la Universidad de la República. En particular, se ha comenzado a explorar el uso de Redes Neuronales Recurrentes (RNN) como herramienta para reproducir decisiones óptimas del STHTC. Una vez entrenadas, estas redes permiten generar soluciones en tiempo real, lo cual habilita simulaciones de largo plazo y evaluaciones de políticas energéticas con bajo costo computacional.

No obstante, uno de los principales desafíos de este enfoque radica en el costo de entrenamiento, ya que cualquier modificación en la estructura del sistema obliga a reconstruir el conjunto de instancias y reentrenar el modelo desde cero. Como alternativa, se propone una estrategia de descomposición modular, consistente en resolver modelos parciales por agente, lo cual es compatible con la arquitectura actual. Esta aproximación permitiría generar soluciones intermedias reutilizables y reducir significativamente el esfuerzo computacional requerido.

En conjunto, estas posibles extensiones delinean una agenda de trabajo que combina herramientas clásicas de optimización con metodologías de inteligencia artificial. El objetivo es avanzar hacia sistemas de apoyo a la decisión más ágiles, adaptativos y capaces de operar bajo incertidumbre, contribuyendo así a una gestión energética más eficiente, robusta y sostenible.

Referencias

Alexander, C., Ishikawa, S., y Silverstein, M. (1977). *A pattern language: Towns, buildings, construction*. Oxford University Press.

Bellman, R. (1957a). *Dynamic programming*. Princeton, USA: Princeton University Press.

Bellman, R. (1957b). A markovian decision process. *Journal of Mathematics and Mechanics*, *6*, 679–684.

Booch, G. (1991). *Object-oriented design with applications*. Benjamin/Cummings.

Casaravilla, G., Chaer, R., y Alfaro, P. (2009). *SIMSEE - Memoria final de ejecución proyecto PDT 47/12* (Inf. Téc.). Montevideo, Uruguay: Proyecto PDT 47/12. Descargado de https://simsee.org/db-docs/Docs_secciones/nid_10222/pdt_47_12.pdf (Accedido el 8 de noviembre de 2025)

Chaer, R. (2008). *Simulación de sistemas de energía eléctrica*. Montevideo, Uruguay. Descargado de https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/2877 (Accedido el 8 de noviembre de 2025)

Chaer, R., Coppes, E., Barreto, F., Tutté, C., Maciel, F., Forets, M., ... Cohn, D. (2013). *Memoria final proyecto ANII-FSE2009-18* (Inf. Téc.). Montevideo, Uruguay: ANII. Descargado de https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/36745/1/CBTMFCGAPCC10.pdf (Accedido el 8 de noviembre de 2025)

Cook, S. (2000). *The p vs np problem*. Clay Mathematics Institute, Millennium Problems. Descargado de https://www.claymath.org/millennium-problems/p-vs-np-problem

Cook, S. A. (1971). The complexity of theorem-proving procedures. En *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158). New York, NY: ACM. doi: 10.1145/800157.805047

Dantzig, G. B. (1949). *Programming in a linear structure* (Inf. Téc. n.º P-392). Santa Monica, CA: The RAND Corporation.

Dantzig, G. B. (1951). Maximization of a linear function of variables subject to linear inequalities. En T. C. Koopmans (Ed.), *Activity analysis of production and allocation* (pp. 339–347). Wiley.

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton, NJ: Princeton University Press.

Farhat, I., y El-Hawary, M. (2009). Optimization methods applied for solving the short-term hydrothermal coordination problem. *Electric Power Systems Research*, *79*(9), 1308–1320.

Ferreira, G. (2008). *Modelos utilizados para el despacho energético óptimo.* Descargado de https://www.bcu.gub.uy/Comunicaciones/Jornadas%20de%20Economa/iees03j3681009.pdf (Accedido el 8 de noviembre de 2025)

Fortnow, L. (2009). The status of the p versus np problem. *Communications of the ACM*, *52*(9), 78–86. doi: 10.1145/1562164.1562186

Fowler, M. (2004). *Uml distilled: A brief guide to the standard object modeling language* (3.ª ed.). Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., y Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software.* Addison-Wesley.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. En *Proceedings of the sixteenth annual acm symposium on theory of computing* (pp. 302–311). New York, NY, USA: Association for Computing Machinery. Descargado de https://doi.org/10.1145/800057.808695 doi: 10.1145/800057.808695

Kay, A. C. (1993). The early history of smalltalk. *ACM SIGPLAN Notices*, *28*(3), 69–95.

Larman, C. (2004). *Applying uml and patterns: An introduction to object-oriented analysis and design and iterative development* (3.ª ed.). Prentice Hall.

Mauriz, J., Porteiro, R., y Ibarburu, M. (2024, noviembre). MOP: First Industrial Application of Stochastic Dual Dynamic Programming for Optimization of the Uruguayan Power System. En *2024 IEEE URUCON* (pp. 1–5). IEEE. doi: 10.1109/URUCON63440.2024.10850472

Meyer, B. (1997). *Object-oriented software construction* (2.ª ed.). Prentice Hall.

Nocedal, J., y Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer.

Olivera, B. (2024). *Despacho hidrotérmico optimo con técnicas de aprendizaje por refuerzos* (Tesis de maestría, Universidad de la República (Uruguay), Facultad de Ingeniería). Descargado de https://hdl.handle.net/20.500.12008/46912 (Disponible en: https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/46912)

Poder Ejecutivo. (2002). *Reglamento del mercado mayorista de energía eléctrica.* Diario Oficial, Documentos Poder Ejecutivo, Nro. 26.097.

Pressman, R. S. (2010). *Software engineering: A practitioner's approach* (7.ª ed.). McGraw-Hill.

Risso, C., Cabrera, L., Porteiro, R., y Ibarburu, M. (2024). *Modelos para optimizar la eficiencia energética en el complejo hidroeléctrico del río negro* (Inf. Téc.). Montevideo, Uruguay: Facultad de Ingeniería, Universidad de la República. (Presentado en las Jornadas Uruguayas de Computación (JUC 2024))

Risso, C., Nesmachnow, S., Porteiro, R., Vignolo, M., Sierra, E., y Ibarburu, M. (2024). A mixed combinatorial optimization model

for the río negro hydroelectric complex. En *ICSC-CITIES 2024 - VII Congreso Ibero-Americano de Ciudades Inteligentes, San Carlos, Costa Rica* (pp. 1–15). Descargado de https://hdl.handle.net/20.500.12008/47830 (12-14 nov. 2024. Disponible en: https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/47830)

Risso, C., Nesmachnow, S., Vignolo, M., Ramírez, I., Porteiro, R., Ibarburu, M., y Sierra, E. (2025). An enriched mixed combinatorial optimization model to manage the hydrothermal dispatch for the río negro hydroelectric complex. *Programming and Computer Software [to appear]*.

SimSEE. (2024). *Simulación de Sistemas de Energía Eléctrica.* https://simsee.org. (Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República)

Sommerville, I. (2011). *Software engineering* (9.ª ed.). Addison-Wesley.

UTE. (2024). *Modelo de Operación (MOP).* Descargado de https://gitlab.com/utepladmm/MOP (Accessed: 2025-09-18)

van der Wal, J. (1980). *Stochastic dynamic programming: Successive approximations and nearly optimal strategies for markov decision processes and markov games* (Ph.D. thesis). Technische Hogeschool Eindhoven.

Anexo A

Manual de Usuario

A.1. Introducción

El sistema está diseñado para modelar y resolver problemas de despacho eléctrico mediante una interfaz de línea de comandos. A través de esta herramienta, el usuario puede agregar agentes generadores o de demanda, seleccionar el tipo de despacho, configurar el solver, ejecutar la resolución del problema de optimización y guardar el modelo resultante en formato `.lp` o `.mps`.

Características principales

El sistema ofrece las siguientes funcionalidades destacadas:

- ➢ Modelado de sistemas hidro-térmicos.
- ➢ Compatibilidad con múltiples solvers, incluyendo Gurobi, Cbc y Cplex.
- ➢ Generación automática de archivos en formato CSV con los resultados obtenidos.
- ➢ Posibilidad de ejecutar pruebas mediante archivos batch.

A.2. Flujo de trabajo

El flujo de trabajo general del sistema se muestra en la Figura 4.3. Sin embargo, su diseño flexible permite flujos más complejos: por ejemplo, se puede resolver un modelo y luego guardarlo, o guardar primero y resolver en una etapa posterior. Asimismo, el usuario puede interrumpir un proceso en curso y comenzar uno nuevo desde el inicio para trabajar con un conjunto de datos distinto.

A.3. Comandos disponibles

La Tabla 4.3 presenta una descripción general de los comandos disponibles en el sistema. A continuación, se ofrece una explicación detallada de cada uno de ellos, incluyendo su sintaxis, funcionalidad y posibles parámetros de configuración.

inicializar

Descripción

Inicializa el sistema con un horizonte temporal definido por el usuario. Este comando debe ejecutarse como primer paso antes de cualquier otro.

Sintaxis

```
inicializar --t <horizonteTemporal>
```

Parámetros

- `horizonteTemporal`: Número entero positivo que representa la cantidad de períodos a modelar (por ejemplo, `24` para 24 horas).

Ejemplo

```
> inicializar --t 24
[success] El programa se ha iniciado correctamente
```

Restricciones

- Debe ser el primer comando ejecutado al iniciar el programa.
- Solo puede ejecutarse una vez por sesión.

agregarAgente

Descripción

Permite agregar diferentes tipos de agentes generadores al sistema. A continuación se detallan los tipos disponibles, su sintaxis y los parámetros asociados.

Para agregar un agente específico, se debe ejecutar el comando anterior segui-

do de " --<nombre del agente>". A continuación, se describen los distintos agentes disponibles y sus características.

Rio Negro

Descripción

Agrega el sistema hidroeléctrico del Río Negro (Bonete, Baygorria, Palmar) utilizando un modelo lineal.

Sintaxis

```
agregarAgente --RioNegro
<aportes_bonete>
<aportes_baygorria>
<aportes_palmar>
<volumen_inicial_bonete>
<volumen_inicial_palmar>
<costo_agua_bonete>
<costo_agua_palmar>
```

Parámetros

- `aportes_*`: Vectores de longitud `horizonteTemporal` (caudales en m^3/s).
- `volumen_inicial_*`: Volumen inicial del embalse (hm^3).
- `costo_agua_*`: Costo del agua almacenada ($\$/m^3$).

Ejemplo resumido

```
> agregarAgente --RioNegro
> 100 105 98...(24 valores)
> 50 52 48...(24 valores)
> 200 205 198...(24 valores)
> 5000
> 3500
> 0.01
> 0.015
[success] Se ha creado el agente 'Rio Negro' exitosamente
```

Restricciones

- Solo se puede agregar **uno** de los siguientes: `--RioNegro` o `--RioNegroNoLineal` (mutuamente excluyentes).

RioNegroNoLineal

Descripción

Similar a `--RioNegro`, pero utiliza un modelo no lineal para la central Bonete. Los parámetros son idénticos en orden y formato. Es mutuamente excluyente con `--RioNegro`.

UnidadTermicaRapida

Descripción

Agrega una unidad térmica de arranque rápido.

Sintaxis

```
agregarAgente --UnidadTermicaRapida
<potencia_minima>
<potencia_maxima>
<costo_fijo>
<costo_variable>
<tiempo_arranque>
<costo_encendido>
```

Parámetros

- `potencia_minima`, `potencia_maxima` (MW)
- `costo_fijo` ($/h), `costo_variable` ($/MWh)
- `tiempo_arranque` (horas), `costo_encendido` ($)

Ejemplo

```
> agregarAgente --UnidadTermicaRapida
> 50
> 250
> 1000
> 45
> 2
> 5000
[success] Se ha creado el agente 'Unidad T\'ermica R\'apida'
    exitosamente
```

Nota

Se pueden agregar múltiples unidades térmicas utilizando este comando varias veces.

DemandaFija

Descripción
Agrega el perfil de demanda eléctrica del sistema.

Sintaxis

```
agregarAgente --DemandaFija
<vector_demandas>
```

Parámetros

➢ `vector_demandas`: Vector de tamaño `horizonteTemporal` con los valores de demanda por período (MW).

Ejemplo

```
> agregarAgente --DemandaFija
> 300 320 310...320
[success] Se ha creado el agente 'Demanda Interna'
    exitosamente
```

Restricciones

➢ Solo puede agregarse una única demanda.
➢ Es obligatorio agregar la demanda antes de crear el despacho.

.....

crearDespacho

Descripción
Crea el problema de despacho económico en función de los agentes generadores y la demanda definidos previamente.

Para seleccionar un despacho en concreto, se debe ejecutar el comando anterior seguido de `" --<nombre del despacho>"`. A continuación, se describen los distintos agentes disponibles y sus características

DespachoFijo

Sintaxis

```
crearDespacho --fijo
<costo_falla>
```

Parámetros

➢ `costo_falla`: Costo de energía no suministrada ($/MWh).

Ejemplo

```
> crearDespacho --fijo
> 10000
[success] Se ha creado el despacho fijo exitosamente
```

Restricciones

➢ Debe ejecutarse después de haber agregado al menos un agente generador y una demanda.

DespachoProporcional

Descripción

Crea el problema de despacho económico donde la demanda se reparte en escalones definidos por porcentajes y valores de demanda.

Sintaxis

```
crearDespacho --proporcional
<p2> <p3> <p4>
<d1> <d2> ... <dN>
<q1> <q2> <q3> <q4>
```

Parámetros

- `p2, p3, p4`: representan las variables $\hat{p}_{1f}^{(2)}$, $\hat{p}_{1f}^{(3)}$ y $\hat{p}_{1f}^{(4)}$, respectivamente, definidas en la ecuación (4.13).
- `d1, d2, ..., dN`: `di` corresponde al valor total de la demanda en la hora i.
- `q1, q2, q3, q4`: representan las variables $\hat{q}_{1f}^{(1)}$, $\hat{q}_{1f}^{(2)}$, $\hat{q}_{1f}^{(3)}$ y $\hat{q}_{1f}^{(4)}$, respectivamente, definidas en la ecuación (4.13).

Ejemplo

```
> crearDespacho --proporcional
> 0.30 0.50 0.80
> 435.342 325.34 224.53 ... 3414.21
> 10 200 4500 6000
[success] Se ha creado el despacho proporcional por escalones
     exitosamente
```

Restricciones

- Debe ejecutarse después de haber agregado al menos un agente generador y una demanda.
- La cantidad de porcentajes debe ser una menos que la cantidad de escalones.
- Los porcentajes deben estar en orden creciente y estar entre 0 y 1.

DespachoProporcionalAlternativo

Descripción

Crea el problema de despacho económico donde la demanda se reparte en escalones definidos por porcentajes y valores de demanda.

Sintaxis

```
crearDespacho --proporcional
<d1> <d2>...<dN>
<q1> <q2>
```

Parámetros

- `d1, d2, ..., dN`: `di` es la demanda total en la hora i.
- `q1, q2`: corresponden a $\hat{q}_{1f}^{(1)}$ y $\hat{q}_{1f}^{(2)}$, según (4.13).

Ejemplo

```
> crearDespacho --proporcionalAlternativo
> 435.342 325.34 224.53...3414.21
> 10 200
[success] Se ha creado el despacho proporcional alternativo
    exitosamente
```

Restricciones

- Debe ejecutarse después de agregar al menos un agente generador y una demanda.
- La cantidad de porcentajes debe ser una menos que la cantidad de escalones.
- Los porcentajes deben estar en orden creciente y estar entre 0 y 1.

configurarSolver

Descripción

Configura el solver de optimización que se utilizará para resolver el problema de despacho eléctrico.

Sintaxis

```
configurarSolver --<TipoSolver>
```

Opciones disponibles

- `--DummyLp`
- `--Gurobi`
- `--Cbc`
- `--Cplex`

Ejemplo

```
> configurarSolver --Gurobi
[success] Solver 'Gurobi' configurado exitosamente
```

Restricciones

- Este comando debe ejecutarse después de haber creado el despacho.
- Solo puede haber un solver activo por sesión.

grabar

Descripción

Guarda el modelo de optimización actual en un archivo con formato LP compatible con solvers externos.

Sintaxis

```
grabar [--noConstante] <ruta.lp>
```

Parámetros

- `ruta.lp`: Ruta donde se desea guardar el archivo LP.
- `--noConstante` (opcional): Excluye constantes en la función objetivo.

Ejemplos

```
> grabar modelo_despacho.lp
[success] Se ha guardado el modelo exitosamente

> grabar --noConstante modelo_sin_constantes.lp
[success] Se ha guardado el modelo exitosamente
```

resolver

Descripción

Resuelve el problema de optimización utilizando el solver configurado y genera un archivo en formato CSV con los resultados.

Sintaxis

```
resolver [--noConstante] <ruta.csv>
```

Parámetros

- `ruta.csv`: Ruta donde se desea guardar el archivo con los resultados.
- `--noConstante` (opcional): Resuelve sin incluir constantes en la función objetivo.

Ejemplo

```
> resolver resultados_despacho.csv
[success] Problema resuelto exitosamente
```

Salida generada

El archivo CSV contiene los siguientes datos:

- Valores de todas las variables de decisión.
- Valor de la función objetivo (original y post-procesada).
- Potencias generadas por cada agente.
- Estado (encendido/apagado) de unidades térmicas.
- Volúmenes almacenados en los embalses por período.

ejecutarPruebas

Descripción

Ejecuta una serie de comandos contenidos en un archivo de texto (modo batch), permitiendo automatizar secuencias de pruebas. Este mecanismo aplica el efecto "capas", es decir, los comandos del archivo se procesan de forma acumulativa, respetando el orden.

Sintaxis

```
ejecutarPruebas <ruta_archivo>
```

Ejemplo

```
> ejecutarPruebas pruebas/caso_base.txt
[info] Procesando comandos desde archivo: pruebas/caso_base.
    txt
[success] Procesamiento del archivo completado
```

Genera

Un archivo `resultados.csv` con el resumen de todas las pruebas ejecutadas.

finalizar

Descripción

Limpia completamente la memoria del sistema, eliminando todos los agentes, demandas y datos cargados. Permite reinicializar el sistema desde cero sin necesidad de cerrar el programa.

Sintaxis

```
finalizar
```

Ejemplo

```
> finalizar
[success] Memoria limpiada exitosamente..
```

Efecto

- Libera completamente la memoria del sistema, eliminando agentes y demanda.
- Habilita la posibilidad de ejecutar nuevamente el comando `inicializar`.

salir

Descripción

Finaliza la ejecución del programa. Antes de cerrar, libera todos los recursos utilizados y guarda correctamente los archivos necesarios, como los resultados del despacho.

Sintaxis

```
salir
```

Ejemplo

```
> salir
[success] Archivo de resultados cerrado.
[success] Saliendo del programa..
```

A.4. Ejemplos de uso

Caso 1: modelo base

Un ejemplo de ejecución del modelo base descrito en el Sección 4.1 se muestra a continuación:

Flujo de comandos (hidroeléctrico simple)

```
> inicializar --t 24
[success] El programa se ha iniciado correctamente
> agregarAgente --RioNegro
> 100 105 98...(24 valores)
> 50 52 48...(24 valores)
> 200 205 198...(24 valores)
> 5000
> 3500
> 0.01
> 0.015
[success] Se ha creado el agente 'R\'io Negro' exitosamente
> agregarAgente --DemandaFija
> 300 320 310...320
[success] Se ha creado el agente 'Demanda Interna' exitosamente
> crearDespacho --fijo
> 10000
[success] Se ha creado el despacho fijo exitosamente
> configurarSolver --Gurobi
[success] Solver 'Gurobi' configurado exitosamente
> resolver resultados_hidro.csv
(solver Gurobi ejecutandose)
[success] Archivo CSV generado correctamente
[success] Post-procesamiento completado. Nuevo valor F.O.:...
> salir
[success] Archivo de resultados cerrado.\033[0m
[success] Saliendo del programa..
```

Caso 2: Sistema Hidro-Térmico y costo de agua no lineal de Bonete

En este caso, al modelo descrito en la Sección 4.1 se le añade una unidad térmica de arranque rápido, como se describe en la Subsección 4.3.4, y se incorpora el costo del agua no lineal en Bonete, según lo detallado en la Subsección 4.3.5:

Flujo de comandos (modelo hidro-térmico mixto)

```
> inicializar --t 24
[success] El programa se ha iniciado correctamente
> agregarAgente --RioNegroNoLineal
> (datos de entrada.)
[success] Se ha creado el agente 'R\'io Negro Bonete No Lineal'
    exitosamente
> agregarAgente --UnidadTermicaRapida
> 50
> 250
> 1000
> 45
> 2
> 5000
[success] Se ha creado el agente 'Unidad T\'ermica R\'apida'
    exitosamente
> agregarAgente --UnidadTermicaRapida
> 30
> 150
> 800
> 35
> 1.5
> 3000
[success] Se ha creado el agente 'Unidad T\'ermica R\'apida'
    exitosamente
> agregarAgente --DemandaFija
> 300 320 310...320
[success] Se ha creado el agente 'Demanda Interna' exitosamente
> crearDespacho --fija
> 10000
[success] Se ha creado el despacho fijo exitosamente
> configurarSolver --Cplex
[success] Solver 'Cplex' configurado exitosamente
> grabar modelo_mixto.lp
[success] Se ha guardado el modelo exitosamente
> resolver resultados_mixto.csv
(solver Cplex ejecutandose)
[success] Archivo CSV generado correctamente
[success] Post-procesamiento completado. Nuevo valor F.O.:...
> finalizar
[success] Memoria limpiada exitosamente..
```

A.5. Formato de archivos de prueba

Los archivos de prueba (`.txt`) deben contener una secuencia de comandos válidos, uno por línea. Las líneas que comienzan con el carácter `#` son tratadas como comentarios y son ignoradas durante la ejecución.

Ejemplo de archivo válido:

archivo_prueba.txt

```
inicializar 24

agregarAgente --RioNegro
100 105 98 102...
50 52 48 51...
200 205 198 202...
5000
3500
0.01
0.015

agregarAgente --DemandaFija
300 320 310...

crearDespacho --fija
10000

configurarSolver --Gurobi
resolver salida_prueba.csv
```

A.6. Excepciones y mensajes de Error

Los siguientes mensajes son generados por el sistema ante errores comunes. La tabla detalla la causa y su solución recomendada.

Mensaje	Causa	Solución
`[error] El programa ya está inicializado`	Se intentó ejecutar `inicializar` más de una vez sin reiniciar el sistema.	Ejecutar `finalizar` antes de volver a usar `inicializar`.
`[error] Comando no valido`	Se ejecutó un comando antes de inicializar el sistema.	Iniciar el sistema con `inicializar` como primer paso.
`[error] Ya existe una Demanda Fija`	Se intentó agregar una segunda demanda.	Solo se permite una demanda por sesión.
`[error] Ya existe un Río Negro`	Se intentó agregar una segunda instancia del sistema Río Negro.	Solo se permite una: `--RioNegro` o `--RioNegroNoLineal`.
`[error] Aportes inválidos o cantidad incorrecta`	El vector de aportes tiene una longitud distinta a `horizonteTemporal`.	Verificar que haya exactamente `horizonteTemporal` valores.
`[error] Tipo de solver no reconocido`	El nombre del solver especificado no es válido.	Usar uno de los siguientes: `DummyLp`, `Gurobi`, `Cbc`, `Cplex`.

Tabla A.1: Mensajes de error comunes: causas y soluciones sugeridas.

A.7. Notas adicionales

⚠ Restricciones Importantes

Requisitos obligatorios:

- ➢ Ejecutar `inicializar` antes de cualquier otra operación.
- ➢ Agregar al menos un agente generador y uno de demanda antes de crear el despacho.

Operaciones prohibidas:

- ➢ Agregar simultáneamente `--RioNegro` y `--RioNegroNoLineal`.
- ➢ Ejecutar `resolver` sin haber creado el despacho previamente.

Archivos Generados Automáticamente

Archivo	**Contenido**
`*.lp`	Modelo de optimización exportado en formato LP, compatible con diversos solvers.
`*.csv`	Resultados obtenidos tras resolver el modelo, incluyendo variables de decisión y métricas.
`resultados.csv`	Resumen consolidado de todas las pruebas ejecutadas mediante archivos batch.