Universidad
Rey Juan Carlos

Escuela de Ingeniería de Fuenlabrada

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

El método BLOCC para optimización binivel con restricciones: definición, caracterización teórica y análisis de prestaciones.

Autor: Fernando Real Rojas
Tutor: Prof. Dr. Antonio G. Marqués
Co-tutor: Víctor Manuel Tenorio Gómez

Curso académico 2023/2024

©2024 Fernando Real Rojas
Algunos derechos reservados
Este documento se distribuye bajo la licencia "Atribución 4.0 Internacional"de Creative Commons, disponible en:
`https://creativecommons.org/licenses/by/4.0/deed.es`

Resumen

En las últimas décadas, la optimización matemática ha ganado importancia en el campo de la ingeniería, gracias en parte a los avances en la capacidad de cómputo, al acceso a ingentes cantidades de datos y a sustanciales progresos en el diseño de algoritmos. El crecimiento exponencial de la potencia de cálculo ha permitido resolver problemas de optimización complejos que antes eran inviables, mientras que la proliferación de datos de diversas fuentes ha proporcionado la materia prima necesaria para que los modelos de optimización sean más precisos y reflejen mejor el mundo real. Además, las innovaciones en los algoritmos y los conocimientos adicionales obtenidos de los avances científicos han mejorado drásticamente la capacidad de encontrar soluciones óptimas de forma rápida y eficaz, aumentando así la capacidad para abordar problemas complejos y optimizar el rendimiento en aplicaciones reales.

Canónicamente, los problemas de optimización se caracterizan por un conjunto de variables y parámetros, una función objetivo y un conjunto de restricciones. La convergencia de los algoritmos de optimización en este tipo de entornos se entiende bien cuando las variables son continuas y las funciones son convexas o suaves. Sin embargo, las aplicaciones del mundo real pueden involucrar a múltiples agentes, cada uno con sus propias funciones objetivo, lo que requiere un planteamiento de solución más complejo. En estos casos, un método sencillo y común consiste en combinar los dos objetivos en una única función objetivo mediante una combinación convexa. Sin embargo, este enfoque no representa con precisión la dinámica cuando los agentes tienen distintos niveles de influencia. Un buen ejemplo de ello es el diseño de redes, en el que un operador diseña primero la red y luego los usuarios deciden cómo utilizarla. El marco adecuado para abordar este tipo de problemas es la optimización binivel (BLO, por sus siglas en inglés), que modela eficazmente los procesos jerárquicos de toma de decisiones en los que intervienen dos agentes con grados de poder desiguales. Este Trabajo Fin de Grado (TFG) se centra en explorar y abordar estos problemas de BLO y, en particular, aquellos sujetos a restricciones.

Los problemas de BLO son sustancialmente más difíciles que sus homólogos de un solo nivel. Requieren definir un objetivo de nivel superior, un objetivo de nivel inferior y sus respectivas variables y restricciones de optimización. Los dos problemas están intrínsecamente acoplados, lo que hace que la optimización global no sea convexa, aunque cada nivel sea convexo por separado. El interés por la BLO ha aumentado en los últimos años, en parte debido a sus aplicaciones para abordar problemas complejos de aprendizaje automático. Varios estudios recientes se han centrado en el desarrollo de algoritmos eficientes basados en gradientes con garantías teóricas. Sin embargo, la literatura existente aborda principalmente problemas BLO sin restricciones o con restricciones simples que no acoplan variables entre los niveles superior e inferior, excluyendo así una amplia gama de aplicaciones complejas.

Este TFG explora el escenario mencionado, introduciendo un algoritmo de primer orden, denominado BLOCC (*Bilevel Optimization with Coupled Constraints*), para abordar este tipo de problemas mediante una estrategia de algoritmos penalizados. Nuestra contribución es doble. Desde una perspectiva científica, diseñamos el algoritmo BLOCC y, bajo una serie de hipótesis, caracterizamos su convergencia. Desde el punto de vista de la aplicación, demostramos su eficacia en dos ejemplos: la selección de hiperparámetros en el entrenamiento de modelos basados en máquinas de vectores soporte (SVM, por sus siglas en inglés) y la planificación de infraestructuras en redes de transporte.

En suma, se estima que el trabajo presentado en este TFG es un paso adelante significativo hacia la comprensión y el diseño de algoritmos para problemas de BLO y, en particular, cuando estos problemas están sujetos a restricciones que acoplan las variables de los niveles superior e inferior.

Abstract

Mathematical optimization has become increasingly relevant in engineering over the last few decades due to significant advancements in computational resources, expanded access to vast amounts of data, and substantial progress in algorithmic design. The exponential growth in computational power has enabled engineers to solve complex optimization problems that were previously infeasible, while the proliferation of data from various sources has provided the raw material necessary for optimization models to be more accurate and reflective of real-world scenarios. Furthermore, innovations in algorithms and the additional insights gotten from scientific advancements, have dramatically improved the ability to find optimal solutions quickly and effectively, thus enhancing the overall capability to tackle intricate engineering challenges and optimize performance across numerous applications.

Canonically, optimization problems are characterized by a set of optimization variables, parameters, an objective function, and a set of constraints. The convergence of optimization algorithms in these settings is well-understood when the variables are continuous and the functions involved are convex or smooth. However, real-world applications often involve multiple agents or players, each with their own objective functions, requiring a more complex solution approach. In such scenarios, a common yet simplistic method is to combine the two objectives into a single objective function using a convex combination. This approach, however, fails to accurately represent the dynamics when agents have different levels of influence. A prime example of this is seen in network design, where an operator first designs the network, and then users make decisions on how to utilize it. The appropriate framework to tackle these types of problems is bilevel optimization (BLO), which effectively models the hierarchical decision-making processes involving two agents with unveven degrees of power. This dissertation focuses on exploring and addressing these BLO problems and, in particular, those subject to constraints.

BLO problems are substantially more challenging than their single-level counterparts. They require defining an upper-level objective, a lower-level objective, and their respective optimization variables and constraints. The two problems are inherently coupled, making the overall optimization non-convex, even when each level is individually convex. Interest in BLO has surged in recent years, partly due to its applications in addressing complex machine learning challenges. Several recent studies have focused on developing efficient gradient-based algorithms with provable guarantees for solving BLO problems. However, the existing literature mainly addresses bilevel problems either without constraints or with simple constraints that do not couple variables across the upper and lower levels, thereby excluding a range of complex applications.

This Bachelor´s Thesis (BT) explores this challenging but less studied scenario and introduces a fully first-order algorithm, termed BLOCC (Bi-Level Optimization with Coupled Constraints), to tackle such problems. Our contribution is twofold. From a scientific perspective, we design the BLOCC algorithm and, under a set of assumptions, characterize its convergence. From an application perspective, we demonstrate its effectiveness on two well-known real-world applications: hyperparameter selection in support vector machine (SVM) model training and infrastructure planning in transportation networks.

Overall, we believe that the work presented in this BT is a significant step forward in understanding and designing algorithms for BLO problems, particularly in the presence of constraints.

Agradecimientos

Quiero empezar expresando mi gratitud hacia quienes han sido fundamentales en la realización de este TFG: Antonio G. Marqués y Víctor Manuel Tenorio. Su constante ayuda no solo ha contribuido significativamente en el desarrollo de este TFG, sino que también han hecho que el proceso sea mucho más sencillo y que el camino recorrido haya sido sumamente agradable. Asimismo, quiero agradecer a Luis Cadarso, cuya aportación de conocimientos ha resultado esencial para alcanzar los objetivos de este proyecto. También merecen un especial agradecimiento mis coautores del artículo de investigación en el que se basa este TFG: Liuyuan Jiang, Quan Xiao y el profesor Tianyi Chen del Rensselaer Polytechnic Institute (Nueva York, EE.UU.). De igual forma estoy agradecido al profesor Geert Leus (TU Delft, Países Bajos) por brindarme la oportunidad de participar en grupos de investigación internacional.

Desde un punto de vista más personal, quiero dedicar este TFG de manera especial a mi abuela Cayetana, que se marchó hace dos meses. A pesar de su ausencia, sé que estaría muy orgullosa y no dejaría de decir que su nieto es “dos veces ingeniero”.

Índice general

Resumen II

Abstract III

Agradecimientos IV

Glosario de acrónimos y abreviaturas IX

1. Introducción 1

1.1. Motivación y estado del arte . . . 1

1.1.1. Los problemas binivel: orígenes y definición . . . 1

1.1.2. Aplicaciones de la BLO en ciencia e ingeniería . . . 2

1.2. Contribuciones y objetivos . . . 3

1.3. Estructura del documento . . . 5

1.4. Grupos de tareas y tiempo dedicado . . . 6

1.5. Colaboraciones, publicaciones y repositorio de código . . . 6

2. Optimización binivel 8

2.1. Fundamentos de la optimización . . . 8

2.1.1. Introducción a la optimización . . . 8

2.1.2. Propiedades de la función objetivo y de las restricciones . . . 10

2.1.3. Métodos de resolución de problemas de optimización sin restricciones . . . 12

2.1.4. Métodos para problemas de optimización con restricciones . . . 15

2.2. Fundamentos de la BLO . . . 20

2.2.1. Definición y modelo básico . . . 21

2.3. Métodos para la solución de problemas de BLO . . . 23

2.4. Revisión histórica del estado del arte . . . 26

3. Diseño y caracterización del algoritmo BLOCC 28

3.1. Introducción . . . 28

3.2. Reformulación penalizada del problema BLO con restricciones . . . 30

3.2.1. Validez de la reformulación . . . 32

3.2.2. Caracterización de las propiedades de la reformulación . . . 36

3.3. El algoritmo BLOCC . . . 40

3.3.1. Descripción del algoritmo . . . 40

3.3.2. Demostración de convergencia 41
3.3.3. Resolución del nivel inferior y complejidad computacional 44
3.4. Ejemplo didáctico y resumen ejecutivo 46
3.4.1. Ejemplo didáctico 46
3.4.2. Resumen ejecutivo 48

4. Experimentos **49**
4.1. Optimización de hiperparámetros 49
4.1.1. Formulación del problema 50
4.1.2. Resultados experimentales 51
4.2. Aplicaciones a la planificación de redes de transporte 55
4.2.1. Introducción al problema 55
4.2.2. Resultados numéricos para la red de 3 nodos 61
4.2.3. Resultados numéricos para la red de 9 nodos 62
4.2.4. Resultados numéricos para la red de Sevilla 63

5. Conclusiones y líneas futuras **65**
5.1. Contribuciones, resultados e impacto 65
5.2. Líneas de trabajo futuro 66

Índice de figuras

2.1. Representación gráfica de una función $\alpha-$fuertemente convexa. 10
2.2. Representación gráfica de la propiedad de continuidad de Lipschitz. 11
2.3. Representación gráfica de la propiedad de suavidad de Lipschitz. 12
2.4. Problema de optimización binivel. Ilustración del problema del nivel inferior en función de las variables del nivel superior. 21
2.5. Evolución del nivel inferior en función de las variables del nivel superior y función objetivo en el nivel superior. 23

3.1. Reformulación con métodos de penalización de problema de optimización binivel. Cálculo de $\nabla v(x)$ en presencia de restricciones acopladas. 32
3.2. Resultados obtenidos tras aplicar el algoritmo BLOCC al ejemplo didáctico definido en (3.26). 47

4.1. Resultados del experimento de optimización de hiperparámetros con un modelo SVM aplicado al dataset diabetes. 54
4.2. Resultados del experimento de optimización de hiperparámetros con un modelo SVM aplicado al dataset fourclass. 55
4.3. Ejemplo de diseño de una red de transporte. 57
4.4. Evolución del objetivo de nivel superior negativo con el tiempo para un problema de diseño de red de 3 nodos. 60
4.5. Pérdida de optimalidad en el problema de optimización de nivel inferior con el tiempo para un problema de diseño de red de 3 nodos. 60
4.6. Evolución del objetivo de nivel superior negativo con el tiempo para un problema de diseño de red de 9 nodos. 62
4.7. Evolución del objetivo de nivel superior negativo con el tiempo para un problema de diseño de red de metro en la ciudad de Sevilla, España. 62
4.8. Topología de la red de Sevilla. 63

Índice de tablas

3.1. Comparación del algoritmo BLOCC con algoritmos existentes en el estado del arte. . . 30

4.1. Experimentos para problema de diseño de red de transporte. Valor de los parámetros para una red de 3 nodos. 61

Glosario de acrónimos y abreviaturas

AiPOD *Alternating implicit projected stochastic gradient descent.*

B&B *Branch&Bound.*

BLO *Bilevel Optimization.*

BLOCC *Bilevel optimization with coupled constraints.*

BT *Bachelor's Thesis.*

BVFSM *Bilevel value function based sequential minimization.*

CCs *Coupled Constraints.*

cf. *confer.*

DSSP *Data science and signal processing for networks and society group of URJC.*

e.g. *exempli gratia.*

EE.UU. Estados Unidos.

GAM *Gradient approximation method.*

GD *Gradient descent.*

i.e. *id est.*

IGD *Implicit gradient descent.*

KKT Karush-Kuhn-Tucker.

LICQ *Linear independence constraint qualification.*

LV-HBA *Lagrangian value function-based hessian-free bilevel algorithm.*

MILP *Mixed integer linear programming.*

NeurIPS *Neural information processing systems.*

NP *Nondeterministic Polynomial time.*

PGD *Projected gradient descent.*

SVM *Support vector machine.*

TFG Trabajo de Fin de Grado.

URJC Universidad Rey Juan Carlos.

Capítulo 1

Introducción

El Trabajo Fin de Grado (TFG) que se presenta en esta memoria aborda la resolución de problemas de optimización binivel (BLO, por sus siglas en inglés) con restricciones, proponiendo, caracterizando y aplicando un nuevo algoritmo basado en métodos de penalización. Este capítulo introductorio está dividido en cinco secciones. La sección 1.1 proporciona una descripción general de la BLO y destaca su importancia en diversas áreas de aplicación. La sección 1.2 detalla las contribuciones científicas y tecnológicas de este TFG. La sección 1.3 explica la estructura de los capítulos restantes. En la sección 1.4 se describen detalladamente las principales tareas realizadas y se estima el tiempo total dedicado a este TFG. Finalmente, la sección 1.5 brinda información sobre las publicaciones científicas derivadas de las tareas realizadas y el código utilizado.

1.1. Motivación y estado del arte

Los algoritmos de optimización se han convertido en una herramienta fundamental en la ciencia y la ingeniería, destacándose por su papel crucial en el modelado y resolución de problemas complejos. Esta amplia adopción se debe no solo a la versatilidad de los enfoques basados en optimización para captar relaciones no triviales entre variables de interés, sino también a los recientes avances en capacidad de cómputo y en el diseño de algoritmos con garantías teóricas. Estos progresos han mejorado significativamente nuestra capacidad para abordar con eficacia problemas a gran escala y de alta dimensión, haciendo de la optimización una herramienta indispensable en diversos campos, desde el aprendizaje automático y los sistemas de transporte hasta la gestión de la energía o el diseño adaptativo de sistemas de comunicaciones.

En su forma canónica, los problemas de optimización implican una única función objetivo con múltiples restricciones. Este tipo de formulación puede abarcar una amplia gama de situaciones, desde la minimización de costes y la maximización de la eficiencia hasta el equilibrio en la asignación de recursos. Sin embargo, algunos problemas del mundo real son más complejos y requieren formulaciones de optimización más sofisticadas. La BLO es uno de estos campos avanzados, cuyo interés emana de su capacidad para gestionar procesos jerárquicos de toma de decisiones en los que un problema de decisión está anidado dentro de otro. Este trabajo se centra en la BLO y, en particular, en el desarrollo de esquemas eficientes y con garantías de convergencia, capaces de resolver problemas de BLO con restricciones en el nivel inferior que acoplan variables de nivel superior e inferior.

1.1.1. Los problemas binivel: orígenes y definición

Los orígenes de la BLO se remontan a mediados del siglo XX, vinculados a los trabajos sobre teoría de juegos de Von Stackelberg [1], surgiendo la primera formulación de un modelo de BLO, como un problema independiente, dos décadas más tarde [2]. Desde entonces, el interés en la BLO ha crecido de manera constante. No obstante, los avances (tanto analíticos como algorítmicos) han sido más lentos

que en otras áreas de la optimización. Esto se debe, en parte, a que los problemas de BLO presentan una estructura anidada que, tal y como se explica en distintos apartados de esta memoria, hace que su resolución sea más compleja.

Formalmente, la BLO es una clase de problemas de optimización que involucra a dos niveles jerárquicos de toma de decisiones, denominados comúnmente nivel superior y nivel inferior. En un problema de BLO, la solución del problema del nivel superior depende del resultado de un problema de optimización anidado en el nivel inferior. Específicamente, el problema del nivel superior busca optimizar una función objetivo teniendo en cuenta la respuesta óptima del problema del nivel inferior. A su vez, el problema del nivel inferior es un problema de optimización en sí mismo, normalmente restringido por las decisiones tomadas en el nivel superior. Las diferencias entre los problemas de nivel superior e inferior se ilustran en mayor detalle en los dos párrafos siguientes.

- El problema del nivel superior suele ser la principal tarea de optimización que debe resolver un responsable de la toma de decisiones. Esto puede implicar decisiones estratégicas en áreas como la economía, el diseño técnico o la gestión de recursos. La función objetivo del nivel superior está influida tanto por las decisiones tomadas en este nivel como por las soluciones óptimas derivadas del problema del nivel inferior.

- El problema del nivel inferior representa una tarea de optimización secundaria que responde a las decisiones tomadas en el nivel superior. Implica un conjunto diferente de variables de decisión y restricciones. El objetivo en este nivel es encontrar la respuesta óptima a las decisiones del nivel superior. Las soluciones del problema del nivel inferior se retroalimentan al problema del nivel superior, creando un bucle de dependencia.

Todo lo anterior pone de manifiesto que, debido a su estructura anidada y al acoplamiento entre los dos niveles, los problemas de BLO sean intrínsecamente difíciles de resolver. Esta complejidad se agrava aún más cuando existen restricciones que acoplan las variables de ambos niveles, haciendo que el problema sea aún más complejo de resolver. La presencia de restricciones añade una capa adicional de dificultad, ya que requieren una cuidadosa coordinación entre los dos niveles, complicando tanto la formulación como el proceso de resolución. No obstante, la mayoría de problemas reales en ciencia e ingeniería presentan restricciones, lo que hace necesario desarrollar algoritmos de optimización bilevel eficientes que sean capaces de lidiar con restricciones y que lo hagan con una complejidad computacional asociada moderada. El diseño de algoritmos de BLO con restricciones es un área que ha recibido atención en los últimos años y, tal y como ya se ha explicado, constituye el problema principal que se aborda en este TFG.

1.1.2. Aplicaciones de la BLO en ciencia e ingeniería

La BLO se presenta de forma natural problemas del mundo real que implican una relación jerárquica entre dos niveles de decisión. Estos se encuentran en campos tan diversos como la gestión (ubicación de instalaciones [3, 4, 5], regulación ambiental [6, 7], política energética [8]), la ingeniería civil (diseño óptimo de estructuras [9, 10, 11, 12]), la planificación económica (políticas sociales y agrícolas [7], precios de la energía eléctrica [13]) o los sistemas de control [14]. La BLO es también importante en el contexto de la ciencia de datos y los sistemas de transporte. Estas dos áreas son de interés para el grupo de investigación donde se ha gestado este proyecto y, además, los escenarios de estudio que se utilizan en el parte final de este trabajo para evaluar y demostrar la utilidad de nuestros algoritmos están relacionados con la ciencia de datos y el transporte. Por ello, se proporcionan a continuación más detalles sobre la relevancia de la BLO en estas dos áreas.

- En los **sistemas de transporte**, la BLO se utiliza para modelar la interacción entre el diseño de redes de alto nivel y la gestión del flujo de tráfico de nivel inferior. Por ejemplo, los planificadores urbanos (nivel superior) pueden optimizar el trazado de una red de transporte para minimizar la

congestión general, mientras que los semáforos individuales (nivel inferior) se optimizan para gestionar el flujo de tráfico basándose en las decisiones de diseño de la red. Otro ejemplo relacionado es la interacción entre el operador de una red de transporte y los pasajeros que utilizan dicha red para sus viajes. En el nivel superior, el operador optimiza la planificación del transporte, como la asignación de rutas y horarios, para maximizar la eficiencia o minimizar los costos. En el nivel inferior, los pasajeros toman decisiones sobre sus rutas y modos de transporte basándose en la planificación del operador. Este enfoque jerárquico permite al operador diseñar sistemas de transporte que consideren las respuestas de los pasajeros, logrando una optimización más efectiva y adaptada a las necesidades reales [15, 16, 17, 18]. Otro ejemplo relevante es el problema de fijación de peajes [19, 20], donde una autoridad optimiza los peajes en una red de carreteras actuando como líder, mientras que los usuarios toman decisiones en función del diseño de los peajes.

Más allá de la planificación y el diseño de la infraestructura de la red, la BLO es también importante dentro de otros aspectos de los sistemas de transporte. Un problema destacado consiste en estimar la demanda entre nodos de una red de transporte [21, 22], que generalmente se obtiene a partir de encuestas estadísticas y sirve como entrada para el proceso de asignación de equilibrio. Si este último produce flujos que no coinciden con los flujos observados (fiables), entonces se puede establecer un problema binivel donde se busca una matriz de demanda (nivel superior) que coincida lo mejor posible con los flujos de equilibrio inducidos (problema de asignación de flujo de nivel inferior).

- En **ciencia de datos** y aprendizaje automático, la BLO desempeña un papel fundamental en el ajuste de hiperparámetros [23, 24]. El problema del nivel superior consiste en seleccionar los mejores hiperparámetros para un modelo de aprendizaje automático, mientras que el problema del nivel inferior se enfoca en entrenar el modelo con esos hiperparámetros para minimizar una función de pérdida. Esta configuración jerárquica permite optimizar simultáneamente los hiperparámetros y los parámetros del modelo, conduciendo a algoritmos de aprendizaje más eficientes y eficaces [25, 26, 27]. Es importante destacar que, especialmente en espacios de hiperparámetros de alta dimensión, los métodos clásicos que seleccionan los hiperparámetros a través de una búsqueda exhaustiva tienen un coste computacional tan alto que, habitualmente, no son capaces ni siquiera de encontrar un óptimo local.

  Además del ajuste de hiperparámetros, la BLO también juega un papel importante en otros ámbitos de la ciencia de datos, como el diseño de arquitecturas de aprendizaje automático definidas sobre grafos (donde un problema consiste en estimar el grafo y otro en utilizar el grafo en una arquitectura que ataca un problema de inferencia asociado), los esquemas de metalearning, o el problema de aprendizaje por refuerzo de agente único involucrando dos niveles temporales [28, 29].

En resumen, la BLO ofrece un marco rico y potente para abordar problemas donde los procesos de toma de decisiones están anidados y dependen uno de otro, proporcionando perspectivas que la optimización a un solo nivel no puede ofrecer. Esto la convierte en una valiosa herramienta para desarrollar soluciones en escenarios complejos del mundo real.

1.2. Contribuciones y objetivos

El TFG aborda el diseño de un algoritmo para resolver problemas de BLO con restricciones acopladas (CCs, por sus siglas en inglés) en el nivel inferior. Para ello se propone una formulación basada en métodos de penalización que da lugar a un algoritmo de descenso de primer orden con garantías en términos de complejidad computacional y convergencia. Las contribuciones metodológicas fundamentales son:

- El diseño de un nuevo algoritmo para abordar problemas de BLO con CCs en el nivel inferior. Este algoritmo transforma el problema BLO en un problema de un único nivel haciendo uso de funciones de penalización. Para lidiar con las CCs del nivel inferior, se utilizan métodos primales-duales. De forma iterativa, se actualizan de forma separada las variables de los niveles superior e inferior, manteniendo el acoplamiento a través de los multiplicadores de Lagrange asociados a las restricciones. Este algoritmo asegura soluciones que convergen a mínimos locales del problema BLO a resolver en tiempo finito, como se demuestra en el capítulo 3.

- La caracterización teórica de las propiedades de dicho algoritmo. Asumiendo ciertas propiedades de convexidad y suavidad de Lipschitz de las funciones que componen el problema BLO, el teorema 1 demuestra que el problema que se resuelve iterativamente es convexo y proporciona una solución ϵ-aproximada al problema original, donde ϵ es un parámetro asociado a la precisión. Además, se proporcionan garantías de convergencia en tiempo finito para el algoritmo en los teoremas 2 y 3.

- La aplicación del algoritmo propuesto para resolver problemas de optimización BLO que caracterizan dos problemas reales: uno en el ámbito de la ciencia de datos (selección de hiperparámetros en el diseño de una SVM) y otro en el ámbito del transporte (diseño de una red capacitada). La aplicación a este último es especialmente relevante debido a dos factores. i) el problema involucra un gran número de variables, por lo que resolverlo sin la necesidad de realizar proyecciones sofisticadas resulta en grandes ahorros computacionales. ii) para modelar correctamente el comportamiento de los usuarios, es crítico satisfacer la condición de optimalidad en el nivel inferior. El algoritmo propuesto proporciona soluciones que garantizan la optimalidad del problema del nivel inferior en cada iteración, mientras que otros algoritmos existentes solo la garantizan a la salida del algoritmo.

Cada una de las contribuciones listadas anteriormente puede identificarse como un objetivo científico orientado a avanzar el estado del arte. La realización de este TFG perseguía además unos objetivos individuales, orientados a la formación del alumno. Entre ellos destacan:

- Profundización en conocimientos de optimización. Durante el grado, el alumno ha recibido nociones básicas de optimización, conociendo los elementos más importantes y los algoritmos clásicos de resolución. Gracias a este TFG, ha desarrollado conocimientos de optimización más profundos, aprendiendo técnicas y algoritmos más sofisticados y aplicándolos a problemas que suponen un reto para la comunidad científica.

- Familiarización con el lenguaje matemático, la presentación de resultados teóricos y demostración de teoremas. A través del desarrollo de este trabajo, el alumno ha adquirido habilidades para leer, escribir y comprender el lenguaje matemático formal, además de aprender a estructurar y demostrar teoremas de manera rigurosa, utilizando la metodología aprendida en asignaturas de formación matemática básica del grado, como Cálculo y Álgebra.

- Puesta en común de distintos conocimientos y habilidades adquiridos a lo largo de su formación. En este TFG, el alumno ha mezclado conocimientos de diversas áreas aprendidas durante la formación de doble grado. Entre ellas, ha destacado la optimización, el procesado de datos, el desarrollo de algoritmos eficientes y el manejo de distintas herramientas informáticas, cuyo conocimiento ha adquirido a lo largo de varias asignaturas vistas en la carrera.

- Habilidades transversales, interacción con grupos extranjeros, escritura de documentos científicos, asistencia a conferencias. El alumno ha desarrollado habilidades comunicativas y de colaboración, participando en proyectos con investigadores internacionales, redactando artículos científicos y presentando sus resultados en conferencias especializadas, lo cual ha fomentado un desarrollo integral y profesional.

Tal y como se concluye al final de esta memoria, se estima que, a la finalización del proyecto, tanto los objetivos científicos como formativos listados anteriormente se han alcanzado de una forma exitosa.

1.3. Estructura del documento

Este documento se articula en torno a cinco capítulos. A continuación, se proporciona una breve descripción de los cuatro capítulos restantes.

- En el capítulo 2 se presentan los fundamentos de BLO, así como los métodos principales para resolver problemas de BLO. El capítulo comienza con una revisión de los fundamentos de la optimización, incluyendo la presentación de los elementos básicos de un problema de optimización y la descripción de algunas propiedades matemáticas deseables, como la convexidad o la suavidad de Lipschitz. Luego se presentan distintos métodos de resolución de problemas de optimización sin restricciones, basados principalmente en técnicas de descenso por gradiente. Posteriormente, se explican los métodos más habituales para resolver problemas de optimización con restricciones, haciendo especial énfasis en los métodos basados en la teoría dual, como los algoritmos duales o primales-duales. Finalmente, se introduce BLO, definiendo un modelo básico y describiendo sus elementos. A continuación, se revisa la evolución histórica de BLO y se describen los métodos de resolución de problemas propuestos hasta la fecha, con un enfoque particular en los métodos actuales para resolver problemas de BLO con CCs.

- El capítulo 3 detalla los resultados principales de este TFG, con un enfoque en el algoritmo desarrollado para BLO con CCs. Inicialmente, se describen los problemas de BLO con restricciones y se revisan los algoritmos del estado del arte, destacando sus ventajas y principales limitaciones. Posteriormente, se presenta una reformulación penalizada de los problemas de BLO con CCs, validando su eficacia bajo ciertas hipótesis de convexidad y suavidad de Lipschitz en las funciones involucradas en el problema. A continuación, se introduce el algoritmo BLOCC, un método innovador para la resolución de problemas de BLO con CCs, que supera las limitaciones predominantes en los algoritmos existentes. Se proporciona un análisis riguroso de la convergencia del algoritmo y se examina su complejidad para problemas de BLO con una estructura específica. Finalmente, se demuestra el funcionamiento del algoritmo mediante un caso de estudio ilustrativo.

- El capítulo 4 aplica el algoritmo desarrollado a dos problemas distintos: i) la selección de hiperparámetros de una SVM y ii) el diseño de una red de transporte capacitada. En el primer problema, se aborda la tarea de encontrar la configuración óptima de hiperparámetros para maximizar el rendimiento de un modelo SVM. Este problema se descompone en dos niveles: en el nivel superior, se optimizan los valores de los hiperparámetros del modelo, minimizando el error en un conjunto de datos de validación. En el nivel inferior, se ajusta el modelo de aprendizaje automático para una configuración específica de hiperparámetros, utilizando un conjunto de datos de entrenamiento. Este enfoque de optimización permite mejorar significativamente la capacidad predictiva del modelo SVM al encontrar la mejor combinación de hiperparámetros. El segundo problema se centra en el diseño de una red de transporte en un entorno con varias redes competidoras. Aquí también se considera una estructura de dos niveles: en el nivel superior, se optimizan los beneficios del operador de la red de transporte, considerando factores como los costes operativos y los ingresos procedentes de los usuarios. En el nivel inferior, los usuarios eligen la red de transporte a utilizar en base a su función de utilidad, que puede incluir parámetros como el tiempo de viaje o la tarifa. Este modelo permite diseñar redes de transporte más eficientes y atractivas para los usuarios, maximizando al mismo tiempo los beneficios para el operador. En ambos casos los resultados demuestran el valor de nuestro diseño y ponen de manifiesto las ventajas frente a otras soluciones existentes en el estado del arte.

- En el capítulo 5 se proporcionan unas breves conclusiones con los resultados más significativos y se identifican varias líneas de trabajo futuro, lo que cierra este TFG.

1.4. Grupos de tareas y tiempo dedicado

Para el desarrollo de este TFG, el estudiante ha realizado diversas tareas que abarcan desde una revisión exhaustiva del estado del arte en el ámbito de estudio hasta la implementación de un nuevo algoritmo para resolver problemas de BLO con CCs. A continuación se detallan estas tareas junto con una breve descripción de cada una:

- Revisión de la literatura. Esta revisión se realizó en dos fases. En una fase inicial se revisaron artículos de BLO (tanto tutoriales como de algoritmos penalizados), así como de aplicaciones de BLO a transporte. En una segunda fase, más avanzado el proyecto, se llevó a cabo una revisión más exhaustiva de la literatura de BLO, con especial énfasis en métodos binivel capaces de manejar restricciones.
- Gestación de la idea original, formulando un problema BLO con restricciones y uso de multiplicadores de Lagrange para estimar el efecto de las CCs en la función de valor.
- Revisión del diseño de algoritmos y de los resultados téoricos, incluyendo los enunciados y demostraciones de los teoremas.
- Implementación del algoritmo desarrollado para optimización binivel aplicada a SVMs y problemas de transporte. En ambos casos las tareas incluyeron: particularización de la formulación, diseño de los algoritmos y programación del código. Los lenguajes de programación utilizados fueron Python y Matlab. Los problemas de optimización se resolvieron desarrollando algoritmos propios.
- Escritura del artículo, escritura de la presente memoria y preparación de la presentación. Si bien la tarea de escritura en el artículo [30] se ciñó a la sección experimental, el estudiante ha redactado la totalidad de la memoria asociada a este TFG.

De forma general, las tareas previamente listadas se han ejecutado secuencialmente. No obstante, en ocasiones, los resultados intermedios en alguna de las tareas han revelado problemas o aspectos no previstos inicialmente, lo que ha exigido revisitar las tareas precedentes. Entre los ejemplos en esta última categoría, pueden mencionarse la modificación de la formulación en función de los resultados obtenidos durante las simulaciones, o la lectura de literatura cuando, en la formulación, se ha decido incorporar una nueva característica de la red de transporte.

El trabajo aquí presentado se ha llevado a cabo durante 5 meses (primeros pasos en septiembre de 2023 y luego desde abril a julio de 2024) y ha supuesto la inversión de unas 450 horas de trabajo por parte del estudiante. De forma aproximada, las tareas de revisión del estado del arte consumieron un 15 % del tiempo. La formulación, diseño y programación de algoritmos consumió en torno al 30 % del tiempo. La revisión de resultados teóricos llevó en torno al 15 %. El procesamiento de datos, diseño y ejecución de simulaciones consumió un 15 % del tiempo y el análisis y visualización de resultados un 5 %. La redacción de los documentos consumió un 20 % del tiempo.

1.5. Colaboraciones, publicaciones y repositorio de código

El trabajo presentado en esta memoria se enmarca en las actividades del grupo de investigación de alto rendimiento de la Universidad Rey Juan Carlos (URJC) en Ciencia de Datos y Procesamiento de Señal (DSSP, por sus siglas en inglés).

En lo que a financiación se refiere, las tareas de investigación y desarrollo del TFG se asocian a paquetes de trabajo de los proyectos de investigación competitivos "Procesamiento de señal para datos definidos sobre grafos: Aprovechando la estructura en dominios irregulares" (MCI/AEI/PID2019-105032GB-I00), "Integración de las Ciencias del Transporte y de los Datos para impulsar una movilidad en red Socialmente Responsable y un Entorno Sostenible" (MCI/AEI/TED2021-130347B-I00) y "Procesamiento y aprendizaje de datos sobre grafos: Desde la inferencia de la estructura a las aplicaciones" (MCI/AEI/PID2022-136887NB-I00), dirigidos por el tutor de este TFG.

Asimismo, los miembros del grupo DSSP colaboran con distintos grupos nacionales e internacionales. Dos colaboraciones que han jugado un papel relevante en la gestación y desarrollo de este TFG se han desarrollado con el grupo de optimización y aprendizaje automático del profesor Tianyi Chen (Rensselaer Polytechnic Institute, EE.UU.), así como el grupo en sistemas aeroespaciales y transporte dirigido por el profesor Luis Cadarso (URJC, España). En este sentido, el contenido de este TFG, especialmente en lo que a los resultados de análisis teórico se refiere, está ligado a una colaboración entre la URJC y el Rensselaer Polytechnic Institute, que se ha sustanciado en la siguiente publicación:

- Liuyuan Jiang, Quan Xiao, Victor M. Tenorio, Fernando Real-Rojas, Antonio G. Marques, and Tianyi Chen. *"A Primal-Dual-Assisted Penalty Approach to Bilevel Optimization with Coupled Constraints"*, Conference on Neural Information Processing Systems (NeurIPS), Dec. 2024 (enviado).

Es de justicia, por lo tanto, agradecer la ayuda prestada por los miembros de los grupos anteriormente mencionados y por los coautores del artículo en torno al que se ha realizado este TFG, así como el apoyo financiero de la Agencia Estatal de Investigación, que ha facilitado llevar a cabo las tareas asociadas al TFG, incluidas la asistencia a conferencias y las visitas a los grupos de investigación que colaboran en los proyectos anteriormente mencionados.

Por último, hay que indicar que el código utilizado para ejecutar los experimentos numéricos presentados en los distintos capítulos se encuentra disponible en:

- `https://github.com/frealr/TFG-BLO`

Capítulo 2

Optimización binivel

Este capítulo presenta los conceptos necesarios dentro del campo de la optimización, define de forma rigurosa el problema de BLO, describe la evolución histórica de este campo, discute distintas formas para la resolución de problemas de este tipo e identifica algunas aplicaciones relevantes. El objetivo es proporcionar al lector el contexto necesario para poder entender el diseño y propiedades del algoritmo que se diseña en el capítulo 3 de esta memoria.

Para ello, la información de este capítulo se estructura en torno a cuatro secciones. La sección 2.1 revisa conceptos básicos de optimización que serán importantes en el resto del documento. Estos incluyen las definiciones de convexidad y de continuidad de Lipschitz, los algoritmos de descenso de primer orden, la optimización con restricciones, los métodos de penalización y los métodos duales. Aprovechando estos conceptos y definiciones, la sección 2.2 define de forma rigurosa el problema de BLO, introduce la terminología y discute varios de los desafíos en lo que se refiere a las condiciones de optimalidad de un problema de BLO. La sección 2.3 describe tres métodos clásicos para tratar con problemas de BLO. Se arranca con métodos de descenso, proporcionando detalles sobre la obtención del gradiente del problema binivel utilizando el teorema de la función implícita. A continuación se comentan los métodos de penalización para BLO (que son la base del algoritmo diseñado en este TFG) y, finalmente, se discute brevemente el uso de algoritmos evolutivos en el contexto de BLO. Apoyándose en las secciones anteriores, el capítulo se cierra con la sección 2.4, donde se proporciona una discusión sobre los hitos más importantes en la evolución de la BLO.

2.1. Fundamentos de la optimización

Como ya se ha mencionado, este TFG se centra en el diseño de métodos de penalización de primer orden, i.e. basados en gradientes, para la optimización binivel con restricciones. En consecuencia, la revisión de la optimización de un solo nivel en esta sección pone especial énfasis en los métodos de gradiente, así como los enfoques de penalización y duales para manejar restricciones. Referencias excelentes para el material que se cubre a continuación incluyen [31], [32], y [33].

2.1.1. Introducción a la optimización

De manera sucinta, la optimización se define como el proceso de encontrar la mejor solución posible a un problema dentro de un conjunto definido de parámetros y restricciones, evaluando las soluciones mediante el uso de una función objetivo. Este es un concepto crítico en varios campos como la ingeniería, la economía y la logística, donde el objetivo es maximizar o minimizar una función objetivo. Un problema de optimización típicamente involucra cuatro componentes principales: las variables de estado, las variables de diseño, la función objetivo y las restricciones, tanto de desigualdad como de igualdad.

- **Variables de estado:** Las variables de estado son los parámetros o condiciones fijas del problema de optimización. Estas variables definen el entorno o el estado inicial y no están sujetas a cambios durante el proceso de optimización. Proporcionan el contexto necesario para el problema e incluyen cualquier condición externa o inicial que debe considerarse. El proceso de optimización utiliza estas variables dadas como "entrada" del problema y no están sujetas a optimización.

- **Variables de diseño:** Las variables de diseño son los parámetros controlables que se pueden ajustar para lograr la solución óptima. Estas variables son centrales para el proceso de optimización ya que influyen directamente en el resultado. El problema de optimización involucra encontrar los mejores valores para estas variables de diseño que maximicen o minimicen la función objetivo. La selección y gestión adecuadas de las variables de diseño son cruciales para una optimización efectiva, ya que determinan la flexibilidad y el alcance de la solución. En esta sección, las variables de optimización se denotan como $x \in \mathbb{R}^{d_x}$, siendo d_x el número de variables.

- **Función objetivo:** La función objetivo $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es una expresión matemática que representa el objetivo del problema de optimización. Pese a que la función puede depender tanto de las variables de estado como de las variables de diseño (optimización), en la práctica suele denotarse de forma explícita únicamente su dependencia con las variables de optimización. Desde este punto de vista, para cada valor de las variables de diseño, devuelve un número real que cuantifica la calidad de la solución. De forma alternativa, f *cuantifica* lo que necesita ser optimizado, ya sea para minimizar (si el valor generado por f representa un coste) o maximizar (si el valor generado por f representa una utilidad). La función objetivo proporciona una medida de rendimiento o efectividad, guiando el proceso de optimización hacia el mejor resultado posible. La forma de f puede variar ampliamente, desde funciones lineales simples hasta expresiones no lineales complejas, dependiendo de la naturaleza del problema.

- **Restricciones:** Las restricciones son condiciones que la solución debe satisfacer. Son esenciales para definir la región factible dentro de la cual debe estar la solución óptima. Se definen habitualmente a través de una función auxiliar que toma como entrada las variables de diseño y limitan su valor. Existen dos tipos de restricciones:

  - **Restricciones de desigualdad.** Estas restricciones limitan los valores de las variables de diseño, de forma que la solución evaluada en la función que representa la restricción debe ser menor o igual a un cierto valor. Representan limitaciones que no deben ser superadas, como el estrés máximo permitido en un componente estructural o las restricciones presupuestarias en un modelo financiero. En esta sección las denotaremos como $g^c : \mathbb{R}^{d_x} \to \mathbb{R}^{d_c}$, donde d_c es el número de restricciones de desigualdad.
  - **Restricciones de igualdad.** Estas restricciones requieren que las variables de diseño satisfagan ciertas condiciones exactas, de forma que la solución evaluada en la función que representa la restricción debe ser igual a un cierto valor. Representan requisitos precisos que deben cumplirse, como el equilibrio de fuerzas en un sistema mecánico o la conservación de la masa en un proceso químico. En esta sección las denotaremos como $h^c : \mathbb{R}^{d_x} \to \mathbb{R}^{d_h}$, donde d_h es el número de restricciones de igualdad.

Entre todos los valores de x que satisfacen las restricciones, la optimización implica escoger el valor de x que alcance un mejor valor objetivo $f(x)$. Sin pérdida de generalidad, la formulación canónica de un problema de optimización se formula como la siguiente minimización

$$\begin{aligned} \min_{x} \quad & f(x) && \text{(2.1a)} \\ \text{s.a:} \quad & g^c(x) \leq 0 && \text{(2.1b)} \\ & h^c(x) = 0. && \text{(2.1c)} \end{aligned}$$

Tanto en este como en el resto de problemas de optimización de este documento, "mín" indica "mínimo" y "s. a" indica "sujeto a". En caso de que el problema persiguiese maximizar $f(x)$, bastaría con redefinir la función objetivo de (2.1) como $-f(x)$. Nótese, además, que en (2.1) se asume que las restricciones se escriben de forma que el valor que limita a g^c y a h^c es cero. En caso de que ese valor fuera distinto de cero, bastaría con sustraer (restar) ese valor a ambos lados de la (des-)igualdad y redefinir la función que modela las restricciones. Así, por ejemplo, la restricción $x^2 \leq 4$ puede reescribirse como $x^2 - 4 \leq 0$.

2.1.2. Propiedades de la función objetivo y de las restricciones

Cuando se diseñan algoritmos para resolver un problema de optimización, las propiedades de dichos algoritmos (capacidad de encontrar un óptimo local o global, velocidad de convergencia, robustez frente a errores...) dependen de ciertas propiedades fundamentales de las funciones (objetivo y restricciones) que definen el problema de optimización. Entre estas propiedades se encuentran la convexidad, la continuidad de Lipschitz y la condición de cualificación de restricciones linealmente independientes (LICQ, por sus siglas en inglés). Estas propiedades no solo son esenciales para entender y analizar problemas de optimización, sino que también sirven como base para demostrar los teoremas que caracterizan las propiedades del algoritmo propuesto en este TFG (véase el capítulo 3 para consultar detalles adicionales).

Convexidad y convexidad fuerte

Una función $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es convexa si para todos $x_1, x_2 \in \mathbb{R}^{d_x}$ y $\theta \in [0, 1]$, se cumple la siguiente desigualdad:

$$f\big(\theta x_1 + (1-\theta)x_2\big) \leq \theta f(x_1) + (1-\theta) f(x_2). \tag{2.2}$$

Un conjunto $\mathcal{C} \subseteq \mathbb{R}^{d_x}$ es convexo si para todos $x_1, x_2 \in \mathcal{C}$ y $\theta \in [0, 1]$, el punto $\theta x_1 + (1-\theta)x_2$ también está en $\mathcal{C}$. Matemáticamente,

$$\forall x_1, x_2 \in \mathcal{C},\ \forall \theta \in [0, 1],\ \theta x_1 + (1-\theta)x_2 \in \mathcal{C}. \tag{2.3}$$

Un conjunto $\mathcal{C}$ es convexo y cerrado si es convexo (como se definió anteriormente) y si contiene todos sus puntos límite x tal que $x \in \mathcal{C}\ \forall x \in \partial\mathcal{C}$, donde $\partial\mathcal{C}$ denota la frontera de $\mathcal{C}$. Es decir, para cualquier punto x en la frontera de $\mathcal{C}$, el punto x también pertenece a $\mathcal{C}$.

Una función $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es α_g-fuertemente convexa si para todos $x_1, x_2 \in \mathbb{R}^{d_x}$ y $\theta \in [0, 1]$,

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2) + \frac{\alpha}{2}\theta(1-\theta)\|x_1 - x_2\|^2. \tag{2.4}$$

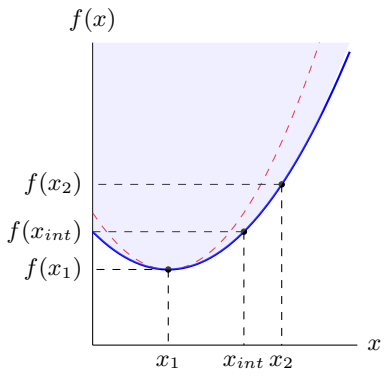


Figura 2.1: Representación gráfica de una función $\alpha-$fuertemente convexa. x_{int} se define como $x_{int} = \theta x_1 + (1-\theta)x_2$, para $\theta = \frac{1}{3}$.

Una función $f(x, y)$ es *conjuntamente* convexa en (x, y) si para cualquier $x_1, x_2 \in \mathbb{R}^{d_x}$, $y_1, y_2 \in \mathbb{R}^{d_y}$, y $\theta \in [0, 1]$,

$$f(\theta x_1 + (1-\theta)x_2, \theta y_1 + (1-\theta)y_2) \leq \theta f(x_1, y_1) + (1-\theta) f(x_2, y_2) \tag{2.5}$$

Continuidad y suavidad de Lipschitz

Una función $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es Lipschitz continua si existe una constante $l_0 \geq 0$ tal que para cualquier $x_1, x_2 \in \mathbb{R}^{d_x}$ se cumple

$$|f(x_1) - f(x_2)| \leq l_0 \|x_1 - x_2\|. \tag{2.6}$$

El menor valor de l_0 que cumple esta desigualdad se denomina la constante de Lipschitz continua de f.

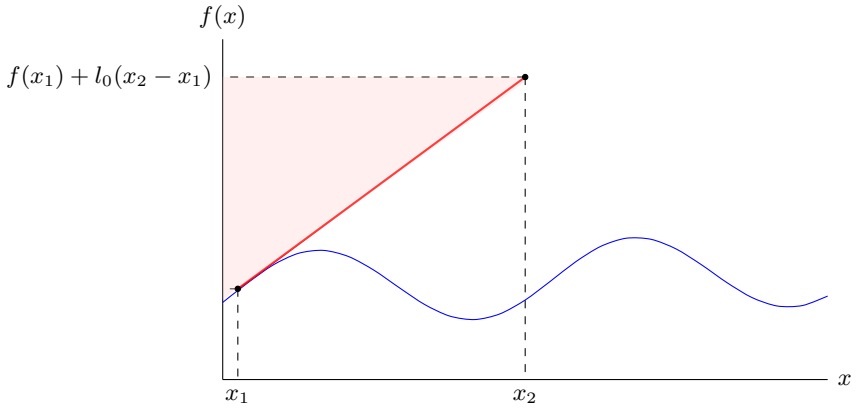


Figura 2.2: Representación gráfica de la propiedad de continuidad de Lipschitz.

Una función $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es Lipschitz suave si su gradiente $\nabla_x f$ es Lipschitz continuo. Es decir, existe una constante $l_1 > 0$ tal que para cualquier $x_1, x_2 \in \mathbb{R}^{d_x}$ se cumple

$$\|\nabla_x f(x_1) - \nabla_x f(x_2)\| \leq l_1 \|x_1 - x_2\|. \tag{2.7}$$

El menor valor de l_1 que cumple esta desigualdad se denomina la constante de Lipschitz suave de f. Por consiguiente, l_1 también cumple la desigualdad

$$|f(x_1) - f(x_2)| \leq |\langle \nabla_x f(x_2), (x_1 - x_2) \rangle + \frac{l_1}{2}(x_1 - x_2)^2|, \tag{2.8}$$

donde $\langle \cdot, \cdot \rangle$ denota el producto escalar euclídeo[1].

[1]Para simplificar la notación, se asume que la notación $\langle \cdot, \cdot \rangle$ se puede aplicar cuando ambos argumentos son vectores (definición clásica), así como cuando un argumento es un vector y el otro es una matriz. En el segundo caso $\langle x, Z \rangle$ da como resultado un vector v en el que la entrada i-ésima se obtiene como el producto escalar euclídeo del vector x con Z_i, la fila i-ésima de la matriz Z. Esto es, $v_i = \langle x, Z_i \rangle = \sum_j x_j Z_{ij}$

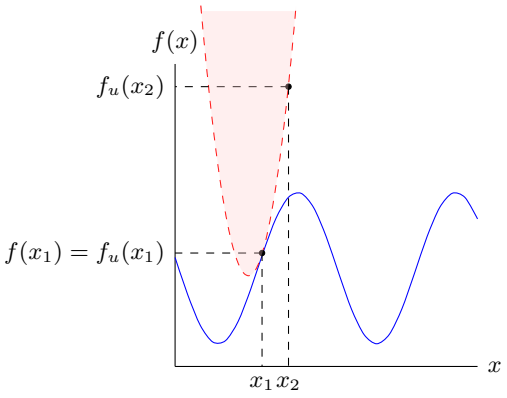


Figura 2.3: Representación gráfica de la propiedad de suavidad de Lipschitz. $f_u(x_2)$ representa una cota superior al valor de $f(x_2)$, y viene dado por $f_u(x_2) = f(x_1) + \nabla_x f(x_1)(x_2 - x_1) + l_1(x_2 - x_1)^2$.

Condición LICQ

La condición LICQ se aplica a los gradientes de las restricciones activas en un punto factible de un problema de optimización con restricciones. Específicamente, consideremos el problema de optimización (2.1). Sea $\bar{x}$ un punto factible. El conjunto de restricciones activas en $\bar{x}$ incluye todas las restricciones de igualdad y aquellas restricciones de desigualdad tales que $g_i^c(x) = 0$. La condición LICQ establece que los gradientes de estas restricciones activas son linealmente independientes en $\bar{x}$. Matemáticamente, los gradientes $\{\nabla_x g_i^c(\bar{x})\}_{i\in\{0,\dots,d_c\}} \cup \{\nabla_x h_j^c(\bar{x})\}_{j\in\{0,\dots,d_h\}}$ son linealmente independientes.

2.1.3. Métodos de resolución de problemas de optimización sin restricciones

La mayoría de los problemas de optimización en ingeniería no tienen soluciones analíticas y requieren el uso de métodos numéricos para resolverlos. Considérese, por ejemplo, el caso de una función objetivo diferenciable en un problema sin restricciones. En este caso, puede calcularse el gradiente de la función objetivo, igualarlo a cero y despejar las variables de diseño para encontrar la solución óptima. Sin embargo, el gradiente suele ser una función no lineal, por lo que encontrar las raíces del sistema de ecuaciones no lineal asociado suele ser inviable. Además, incluso en los casos donde sea posible resolverlo, el sistema puede tener múltiples soluciones, por lo que habría que identificarlas y evaluarlas todas para determinar la óptima.

Dada esta realidad, el procedimiento habitual es recurrir a rutinas numéricas, o algoritmos, que aproximan las soluciones de interés. Otro aspecto para tener en cuenta es que, desde un punto de vista práctico, es poco probable que un único método de optimización pueda resolver eficazmente todos los problemas de optimización no lineal. En consecuencia, resulta necesario disponer de un amplio abanico de técnicas, así como de un conocimiento básico de las mismas para poder adaptarlas (o combinarlas) teniendo en cuenta las particularidades del problema a resolver. El objetivo de esta sección es describir varias de estas técnicas y, en especial, el descenso por gradiente (GD, por sus siglas en inglés). Puesto que el interés de este TFG gravita en torno a problemas con variables continuas y garantías teóricas, en la revisión que se realiza en esta sección no se discuten métodos de optimización discreta (como la programación entera y los métodos combinatorios) ni tampoco métodos heurísticos y metaheurísticos.

Evidentemente, si se dispusiera de recursos computacionales ilimitados, cualquier problema podría resolverse mediante búsqueda exhaustiva, definiendo un conjunto de soluciones candidatas y evaluándolas todas en paralelo. En la práctica, sin embargo, la inmensa mayoría de los métodos de optimización son iterativos. Esto significa que el método genera una secuencia (potencialmente infinita) de soluciones aproximadas y, cuando el método termina, escoge como salida una de estas soluciones (normalmente

la más reciente de la secuencia). La idea clave en estos métodos es que la siguiente iteración (solución candidata) se forma combinando: i) la información local de la función a optimizar en la vecindad de la iteración actual, y ii) los valores de las iteraciones anteriores.

Normalmente, el objetivo es actualizar la iteración de forma que se reduzca la función objetivo. De ahí que esta familia de métodos se denomine a menudo *métodos de descenso*. Un aspecto crítico a la hora de diseñar un método de descenso es cómo elegir la dirección de descenso (es decir, qué tipo de información local de la función a optimizar se explota). Teniendo esto en cuenta, los métodos pueden clasificarse en [33]:

- Métodos de orden cero, que usan solo los valores del objetivo y las restricciones y no usan sus derivadas.
- Métodos de primer orden, que usan los valores y los gradientes de la función objetivo y de las restricciones.
- Métodos de segundo orden, que usan los valores, los gradientes y las matrices Hessianas (es decir, matrices de derivadas de segundo orden) de la función objetivo y de las restricciones.

En teoría, podrían considerarse métodos de orden superior. Sin embargo, rara vez se utilizan en la práctica debido a problemas tanto de precisión como de complejidad computacional. Entre los métodos mencionados, los de primer orden, y en particular el descenso de gradiente, son los más utilizados. Esto se debe a su eficacia a la hora de encontrar la solución óptima, su facilidad de cálculo y su robustez frente a errores o ruido. En la siguiente sección se define en detalle el algoritmo de descenso de gradiente y se analizan algunas de sus propiedades.

Descenso de gradiente

El descenso de gradiente es un método de descenso para la optimización matemática sin restricciones. Es un algoritmo iterativo de primer orden para encontrar un mínimo local de una función multivariable diferenciable. Si la función a optimizar se denota como $f(x)$, el índice de iteración como t y la solución en la iteración t como x^t, el descenso de gradiente implementa la siguiente actualización:

$$x^{t+1} = x^t - \eta_t \nabla_x f(x^t), \tag{2.9}$$

donde $\eta_t > 0$ es el tamaño del paso (o tasa de aprendizaje, en el contexto de ciencia de datos) y $\nabla_x f(x^t)$ es el gradiente de f evaluado en x^t. El algoritmo necesita como "entradas" el punto de partida x^0 y la regla de tamaño del paso. La actualización en (2.9) se ejecuta para un número finito de iteraciones. Los criterios de parada más habituales incluyen haber alcanzado un número máximo de iteraciones, que el cambio en x sea menor que un umbral predefinido o que el cambio en el valor objetivo sea menor que un umbral predefinido.

La lógica detrás de este algoritmo de descenso es que el gradiente $\nabla_x f(x)$ es un vector que apunta en la dirección del ascenso más pronunciado de la función f. Restando una fracción η del gradiente, el algoritmo mueve la solución actual x^t en la dirección del descenso más pronunciado, reduciendo así el valor de la función f.

Las preguntas clave a la hora de implementar un algoritmo de descenso de gradiente son: i) cómo elegir el tamaño de los pasos; ii) si converge y, en caso afirmativo, hacia dónde; y iii) si converge, a qué velocidad. Una respuesta detallada y rigurosa a estas preguntas excede el alcance de este documento. Por ello, en las páginas siguientes se discuten brevemente algunas de estas cuestiones, haciendo más hincapié en las que serán importantes en el contexto del algoritmo que se propone en el capítulo siguiente.

En cuanto a la selección del tamaño del paso, las opciones más comunes son:

1. **Tamaño de paso constante y suficientemente pequeño:** Se utiliza un tamaño de paso fijo η a lo largo de las iteraciones. Este enfoque es de fácil implementación, pero requiere un ajuste cuidadoso del tamaño del paso. En concreto, debe asegurarse que el tamaño del paso sea lo suficientemente pequeño para garantizar la convergencia y, al mismo tiempo, que no sea tan pequeño que haga que la convergencia se vuelva excesivamente lenta.

2. **Tamaño de paso decreciente:** El tamaño del paso se reduce con el tiempo según un esquema específico, como $\eta_t = \frac{\eta_0}{t}$ o $\eta_t = \frac{\eta_0}{\sqrt{t}}$. Un requisito común para esquemas de tamaño de paso decreciente es que los tamaños de paso no sean sumables pero sus cuadrados sí lo sean, es decir, $\sum_{t=1}^{\infty} \eta_t = \infty$ y $\sum_{t=1}^{\infty} \eta_t^2 < \infty$. Esto asegura la convergencia permitiendo que el tamaño del paso disminuya.

3. ***Line search*:** En cada iteración, se utiliza un procedimiento de búsqueda para encontrar un tamaño de paso óptimo η_t que disminuya suficientemente la función objetivo. Este método puede elegir adaptativamente el tamaño del paso basado en el comportamiento de la función en cada iteración. Mientras que el movimiento en cada iteración es de mayor calidad, la complejidad computacional por iteración es también más alta.

En cuanto a la convergencia, la propiedad más importante del descenso de gradiente es su garantía de converger a un mínimo local, siempre que se elija adecuadamente el tamaño del paso. Además, si la función objetivo es convexa, se garantiza la convergencia al óptimo global, ya que en una función convexa todos los mínimos locales son globales.

La velocidad de convergencia depende de las propiedades de la función objetivo y de cómo se elija el tamaño del paso. Por ejemplo, cuando el objetivo es fuertemente convexo y Lipschitz suave, el descenso de gradiente converge linealmente con un tamaño de paso fijo, siempre que el tamaño del paso se elija como $\eta = \frac{2}{\alpha + l_1}$, donde α es la constante de convexidad fuerte y l_1 es la constante de Lipschitz del gradiente. En este caso, el error disminuye exponencialmente rápido hasta alcanzar el valor óptimo. Para funciones convexas pero no fuertemente convexas, la convergencia es generalmente sublineal, lo que significa que el error disminuye a una tasa proporcional a $\frac{1}{t}$, donde t es el número de iteración. Propiedades menos estrictas, como la no convexidad o la falta de suavidad de Lipschitz, llevan a garantías de convergencia más débiles o requieren selecciones de tamaño de paso más avanzadas y utilizando otras técnicas de optimización.

En resumen, la selección del tamaño del paso y las propiedades de convergencia son aspectos cruciales del algoritmo de GD. Entender los compromisos existentes entre estos elementos y ajustar sus valores de forma adecuada resulta esencial para garantizar una optimización eficiente y efectiva, particularmente en el contexto de los problemas de BLO abordados en este TFG.

Variantes del GD

La sección se cierra discutiendo brevemente algunas variantes del descenso de gradiente. Como se ha mencionado anteriormente, los problemas de optimización son muy diversos, por lo que es poco probable que un único método se adapte a todos los escenarios. Las variantes del GD utilizadas en la práctica incluyen:

- **Descenso por subgradiente:** Este método se utiliza cuando la función objetivo no es diferenciable en todo su dominio. Para los puntos donde la función no es diferenciable, se utiliza un subgradiente (que es una generalización del gradiente [34]). La convergencia al óptimo global está garantizada si la función es convexa y el tamaño del paso se elige correctamente, aunque la velocidad de convergencia puede ser mucho más lenta que la del descenso del gradiente.

- **GD estocástico:** Este método se utiliza cuando la función objetivo (y, por lo tanto, el gradiente) implica un operador esperanza y se tiene acceso a una aproximación estocástica del gradiente.

La convergencia puede establecerse utilizando criterios probabilísticos (con probabilidad uno, en esperanza, en distribución...) y, dependiendo de la selección del paso, las iteraciones x_t no tienen por qué converger a un punto fijo, sino que se quedan oscilando en un vecindario de la solución óptima.

- **GD incremental:** Este método se emplea cuando la función objetivo implica una suma sobre un gran número de términos y el gradiente se calcula utilizando sólo un pequeño subconjunto de estos términos (en el extremo, únicamente uno).
- **Método de Newton:** Se utiliza cuando la función objetivo es convexa y dos veces diferenciable y, además, la matriz Hessiana es fácil de invertir. Aunque estrictamente hablando se trata de un método de segundo orden, puede verse como una variante del descenso de gradiente donde, en cada iteración, la dirección de descenso (es decir, el gradiente) se modifica multiplicándola por la inversa de la matriz Hessiana.

Esta lista no es, en absoluto, exhaustiva. Existen muchas otras versiones y mejoras (por ejemplo, métodos acelerados, métodos de impulso, esquemas de descenso coordinado por bloques, métodos proximales...) que ofrecen ventajas adicionales o que pueden aplicarse en escenarios en los que los métodos más clásicos no funcionan bien.

Por último, debe mencionarse también que existen algunas versiones del descenso de gradiente que pueden utilizarse para la optimización con restricciones. La más sencilla y célebre es el descenso de gradiente proyectado (PGD, por sus siglas en inglés). En concreto, supóngase ahora que, además de optimizar la función $f(x)$, la solución también debe satisfacer la restricción $x \in \mathcal{C}$. Entonces, la actualización para el descenso de gradiente proyectado en la iteración t es

$$x^{t+1} = \mathcal{P}_{\mathcal{C}}\left(x^t - \eta_t \nabla_x f(x^t)\right), \tag{2.10}$$

donde $\mathcal{P}_{\mathcal{C}}(\mathbf{z})$ denota la proyección de $\mathbf{z}$ sobre el conjunto $\mathcal{C}$, y η_t es el tamaño del paso en la iteración t.

Los métodos de descenso de gradiente proyectado funcionan bien cuando el problema es convexo y el operador de proyección es fácil de calcular. Este es el caso, por ejemplo, cuando se trata de restricciones de caja. Sin embargo, cuando las restricciones no son lineales y acoplan las variables de optimización, implementar el operador de proyección necesario en cada iteración puede ser tan costoso como un nuevo problema de optimización. Esto supone que el descenso de gradiente proyectado no se considere una alternativa práctica en esos casos. En la siguiente sección se analiza cómo diseñar algoritmos para la optimización con restricciones que puedan manejar un mayor nivel de complejidad.

2.1.4. Métodos para problemas de optimización con restricciones

Existen diversos métodos para resolver problemas de optimización con restricciones, cada uno con sus propias técnicas y enfoques. En este documento se presentan los métodos de penalización, de barrera y duales. Los métodos de penalización y de barrera transforman el problema original (que tiene restricciones) en un problema relacionado que no tiene restricciones. Al seleccionar y ajustar adecuadamente los parámetros de penalización que definen el problema sin restricciones, es posible obtener soluciones que satisfagan las restricciones del problema original. Los métodos duales, por su parte, se centran en resolver el problema desde una perspectiva diferente, trabajando con la formulación dual del problema original para obtener soluciones óptimas de manera eficiente. Cada uno de estos métodos tiene sus propias ventajas y consideraciones, y la elección del método adecuado depende de la naturaleza específica del problema y de las restricciones que se manejan.

Métodos de penalización

Los métodos de penalización son una clase de técnicas utilizadas en optimización para lidiar con problemas con restricciones [35]. Estos métodos pertenecen a la familia de métodos aumentados, que

transforman un problema de optimización con restricciones en un problema sin restricciones mediante la incorporación de un término adicional en la función objetivo por cada una de las restricciones existentes.

Para ser más específicos, considérese el siguiente problema de optimización

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.a:} \quad & g^c(x) \leq 0, \end{aligned} \tag{2.11}$$

donde $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es la función objetivo y $g^c : \mathbb{R}^{d_x} \to \mathbb{R}^{d_c}$ representa d_c restricciones de desigualdad. Consideremos ahora una función escalar $\pi : \mathbb{R} \to \mathbb{R}^+$, que se define de forma que: i) π devuelve cero si su argumento de entrada es negativo o cero y ii) π devuelve un número positivo (y creciente) si su argumento de entrada es positivo. Un ejemplo clásico de esta función es $\pi(\cdot) = (\text{máx}(0, \cdot))^2$. Utilizando esta definición, consideremos ahora la función

$$P(x) := \sum_{i=1}^{d_c} \pi(g_i^c(x))^2 = \sum_{i=1}^{d_c} (\text{máx}(0, g_i^c(x)))^2, \tag{2.12}$$

donde, claramente, se tiene que, si x cumple las d_c restricciones, entonces $P(x) = 0$.

Teniendo en cuenta todo lo anterior, los métodos de penalización transforman el problema con restricciones en (2.11) en el siguiente problema sin restricciones

$$\min_{x} \; L(x, \gamma) = f(x) + \gamma P(x), \tag{2.13}$$

donde γ es un parámetro de penalización que controla la severidad de la penalización. El objetivo es ajustar γ de tal manera que la solución del problema sin restricciones (2.13) se acerque a la solución del problema original con restricciones (2.11). Para obtener la solución óptima de (2.11) es necesario resolver múltiples iteraciones del problema en (2.13), haciendo que, en cada una de ellas, el valor del parámetro γ cambie.

De forma más rigurosa, si t denota el índice de iteración y γ_t denota el valor del parámetro de penalización, en la iteración t-ésima se resuelve (2.13) con el objetivo $L(x, \gamma_t)$, dando lugar a la solución x^t. El valor inicial del parámetro de penalización se fija a $\gamma_0 = 0$ y, en cada iteración, dicho valor se incrementa hasta encontrar un valor γ_t que haga que se satisfagan todas las restricciones. Según [36, sec. 13.1, lema 2], para cualquier iteración $t < T - 1$ se cumple que $L(x^t, \gamma_t) \leq L(x^{t+1}, \gamma_{t+1}) \leq f(x^*)$. Por tanto, si $t \to \infty$ y el valor de γ_t crece al aumentar t, se tiene que $P(x^t) = 0$ y, por consiguiente, $L(x^t, \gamma_t) = f(x^t) \leq f(x^*)$, siendo x^t una solución factible. Dado que el mínimo de $f(x)$ se encuentra en x^*, queda demostrado que $x^t = x^*$. Esto supone que el método propuesto converge y, por lo tanto, se puede utilizar para obtener la solución óptima del problema (2.11).

Métodos de barrera

Un método de barrera o método de punto interior es un tipo de algoritmo utilizado en problemas de optimización con restricciones que busca soluciones óptimas iterando desde un punto inicial dentro del conjunto factible y moviéndose a través del interior de este conjunto. Para evitar que la solución salga del conjunto factible se aumenta el objetivo del problema de optimización con una función de penalización denominada función de barrera. Dicha función tiende a infinito en la frontera de la región factible del problema, evitando así que la solución propuesta abandone la región de factibilidad. Una de las funciones de barrera más utilizadas es el logaritmo, de tal forma que la función de penalización queda como

$$P(x) = -\sum_{i=1}^{d_c} \log(-g_i^c(x))^2, \tag{2.14}$$

donde $g^c(x) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_c}$ modela las d_c restricciones del problema a resolver. De este modo, la función objetivo modificada se convierte en

$$L(x, \kappa) = f(x) + \frac{1}{\kappa} P(x), \tag{2.15}$$

donde κ es el parámetro de penalización. En los métodos de barrera, es habitual introducir este parámetro dividiendo a la función de penalización. Normalmente, se toma un valor de κ suficientemente grande, de tal forma que la penalización sea mínima en puntos lejanos a la restricción. De cualquier manera, la ventaja de este método es que se puede resolver utilizando una técnica de búsqueda sin restricciones. Para encontrar la solución, se comienza en un punto perteneciente a la región factible y luego se busca desde ese punto, utilizando un método de GD u otro método iterativo de descenso aplicable a problemas sin restricciones. Dado que el valor de la función objetivo (2.15) se aproxima al infinito cerca del límite de la región factible del problema definido por (2.11)-(2.12), la técnica de búsqueda mantendrá la solución automáticamente dentro de la región factible, garantizando que se satisfagan las restricciones.

Métodos duales: multiplicadores, lagrangiano y condiciones KKT

La teoría dual en optimización se centra en transformar un problema de optimización, etiquetado como problema primal, en otro problema, conocido como el problema dual. Este enfoque no solo proporciona una nueva perspectiva sobre el problema original, sino que también puede ser utilizado para encontrar soluciones más fácilmente o para proporcionar información adicional sobre las propiedades de las soluciones. En el núcleo de la teoría dual se encuentran los multiplicadores de Lagrange, la función Lagrangiana, la función dual, el problema dual y las condiciones de Karush-Kuhn-Tucker (KKT). Cada uno de estos conceptos se describen a continuación de forma breve:

- **Multiplicadores de Lagrange:** Los multiplicadores de Lagrange son coeficientes que se introducen para convertir un problema de optimización con restricciones en un problema sin restricciones. Estos multiplicadores se asocian con cada una de las restricciones del problema, permitiendo que las restricciones se incorporen directamente en la función objetivo. Supongamos un problema de optimización con restricciones de la forma:

$$\begin{aligned} \min_{x} \quad & f(x) && \text{(2.16a)} \\ \text{s.a.:} \quad & g^c(x) \leq 0 && \text{(2.16b)} \\ & h^c(x) = 0, && \text{(2.16c)} \end{aligned}$$

  donde recordamos que $f : \mathbb{R}^{d_x} \to \mathbb{R}$ es la función objetivo, $g^c : \mathbb{R}^{d_x} \to \mathbb{R}^{d_c}$ son las restricciones de desigualdad y $h^c : \mathbb{R}^{d_x} \to \mathbb{R}^{d_h}$ son las restricciones de igualdad. Los d_c multiplicadores de Lagrange asociados a las restricciones de desigualdad se recogen en el vector denotado por $\mu = [\mu_1, ..., \mu_{d_c}]^T$, mientras que los d_h multiplicadores asociados a las restricciones de igualdad se recogen en $\nu = [\nu_1, ..., \nu_{d_h}]^T$.

- **Lagrangiano:** El siguiente paso consiste en definir una nueva función que combine la función original objetivo con las distintas restricciones. Dicha función se conoce como lagrangiano (o función lagrangiana) y se define como una combinación lineal de la función objetivo y las restricciones, siendo el peso de cada una de las restricciones el multiplicador de Lagrange asociado a la misma. Matemáticamente, el lagrangiano se denota como $\mathcal{L}(x, \mu, \nu)$ y se define como:

$$\mathcal{L}(x, \mu, \nu) = f(x) + \langle \mu, g^c(x) \rangle + \langle \nu, h^c(x) \rangle. \tag{2.17}$$

  Nótese que el lagrangiano es una función que depende tanto de las variables primales como de las variables duales.

- **Función dual:** La función dual $\varphi(\mu, \nu)$ se define como el valor mínimo del Lagrangiano con respecto a x:

$$\varphi(\mu, \nu) = \min_x \mathcal{L}(x, \mu, \nu) = \min_x f(x) + \langle \mu, g^c(x) \rangle + \langle \nu, h^c(x) \rangle. \tag{2.18}$$

  Si utilizamos la notación $x^*(\mu, \nu) = \arg\min_x \mathcal{L}(x, \mu, \nu)$, entonces la función dual puede escribirse de forma alternativa como $\varphi(\mu, \nu) = \mathcal{L}(x^*(\mu, \nu), \mu, \nu)$. Una propiedad importante de la función dual es que su (sub-)gradiente puede expresarse como $\nabla_\mu \varphi(\mu, \nu) = g^c(x^*(\mu, \nu))$ y $\nabla_\nu \varphi(\mu, \nu) = h^c(x^*(\mu, \nu))$.

- **Problema dual:** El problema dual consiste en maximizar la función dual con respecto a los multiplicadores de Lagrange:

$$\max_{\mu \geq 0, \nu} \varphi(\mu, \nu). \tag{2.19}$$

  El problema dual tiene una serie de ventajas. La primera es que la función dual es siempre cóncava, con lo que un algoritmo de ascenso será capaz de encontrar el valor óptimo de los multiplicadores de Lagrange. La segunda es que, si el problema primal original es convexo, la solución óptima del problema primal en (2.16) puede encontrarse como $x^* = x^*(\mu^*, \nu^*)$, donde (μ^*, ν^*) es la solución del problema dual en (2.19).

- **Condiciones KKT:** Las condiciones KKT son un conjunto de condiciones necesarias (y en algunos casos suficientes) para que una solución sea óptima en un problema de optimización no lineal con restricciones. Estas condiciones consideran tanto restricciones de igualdad como de desigualdad. Las condiciones KKT incluyen:

  1. **Condiciones de estacionariedad:**

$$\nabla_x \mathcal{L}(x^*, \mu^*, \nu^*) = \nabla_x f(x^*) + \langle \mu^*, \nabla_x g^c(x^*) \rangle + \langle \nu^*, \nabla_x h^c(x^*) \rangle = 0. \tag{2.20}$$

  2. **Condiciones primales de factibilidad:**

$$g^c(x^*) \leq 0 \tag{2.21a}$$
$$h^c(x^*) = 0. \tag{2.21b}$$

  3. **Condiciones duales de factibilidad:**

$$\mu_i^* \geq 0, \quad i = 1, ..., d_c. \tag{2.22}$$

  4. **Condiciones de complementariedad:**

$$\mu_i^* g_i^c(x^*) = 0, \quad i = 1, ..., d_c. \tag{2.23}$$

  Estas condiciones, de forma conjunta, dan lugar a un sistema de ecuaciones y desigualdades en las variables (x^*, μ^*, ν^*). La solución óptima del problema (2.16) debe satisfacer dicho sistema. Expresado de forma alternativa, la solución de (2.16) pertenece al conjunto de soluciones que satisface (2.20)-(2.23).

La teoría dual, junto con los multiplicadores de Lagrange, la función lagrangiana, la función dual, el problema dual y las condiciones KKT, proporciona un marco muy versátil para abordar problemas de optimización con restricciones. Al transformar el problema original y aplicar estas condiciones, se puede obtener una visión más profunda de la estructura del problema y, en muchos casos, soluciones más eficientes y robustas. En un problema de optimización con restricciones, la solución se encuentra en un punto de silla del lagrangiano, donde se cumplen dos propiedades simultáneamente: i) ser un mínimo

con respecto a las variables de optimización x y ii) ser un máximo con respecto a los multiplicadores μ y ν. Para resolver problemas de optimización mediante la teoría dual, es habitual recurrir a métodos duales o primales-duales. El método dual formula el problema dual, centrándose en encontrar primero el valor óptimo de los multiplicadores de Lagrange y, una vez conseguido esto, obtener el valor óptimo de las variables primales minimizando $\mathcal{L}(x, \mu^*, \nu^*)$ con respecto a x. Por otro lado, el método primal-dual intenta buscar un punto de silla de la función lagrangiana, iterando simultáneamente sobre las variables primales y duales. La siguiente sección proporciona más detalles sobre estos dos métodos.

Métodos numéricos para la optimización dual

Basándose en la teoría dual, los métodos dual y primal-dual son dos de las técnicas más habituales para resolver problemas de optimización con restricciones. En un problema de optimización, la formulación dual transforma el problema original en otro problema (el problema dual), cuyas variables corresponden a los multiplicadores de Lagrange asociados con las restricciones del problema original. Como ya se ha mencionado, resolver el problema dual puede proporcionar información valiosa sobre las propiedades del problema original, y en muchos casos, permite encontrar soluciones óptimas de manera más eficiente.

- **Método dual:** Para implementar el método dual y resolver problemas de optimización, se emplea un enfoque iterativo. Este método ajusta las variables y los multiplicadores de Lagrange de manera sistemática hasta alcanzar la convergencia. En cada iteración, se evalúa la función dual mediante la minimización del lagrangiano y se ejecuta un paso de ascenso por gradiente en los multiplicadores de Lagrange. La expresión del gradiente del problema dual con respecto a los multiplicadores de Lagrange viene dada por

$$\nabla_\mu \varphi(\mu, \nu) = \langle \nabla_\mu (x^*(\mu, \nu)), \nabla_x f(x^*(\mu, \nu)) + \langle \mu, \nabla_x g^c(x^*(\mu, \nu)) \rangle \rangle + g^c(x^*(\mu, \nu)) \tag{2.24a}$$
$$\nabla_\nu \varphi(\mu, \nu) = \langle \nabla_\nu (x^*(\mu, \nu)), \nabla_x f(x^*(\mu, \nu)) + \langle \nu, \nabla_x h^c(x^*(\mu, \nu)) \rangle \rangle + h^c(x^*(\mu, \nu)) \tag{2.24b}$$

  donde ∇_μ representa el jacobiano[2] de $x^*(\mu, \nu)$ con respecto a μ y, por lo tanto, $\nabla_\mu(x^*(\mu, \nu))$ evaluado en un valor particular de (μ, ν) genera como salida una matriz de tamaño $d_x \times d_c$. Puesto que $x^*(\mu, \nu)$ representa un mínimo del lagrangiano, se cumple que $\nabla_x \mathcal{L}(x^*(\mu, \nu), \mu, \nu) = 0$. Por lo tanto, la expresión del gradiente de la función dual respecto con a los multiplicadores de Lagrange se simplifica a

$$\nabla_\mu \varphi(\mu, \nu) = g^c(x^*(\mu, \nu)) \tag{2.25a}$$
$$\nabla_\nu \varphi(\mu, \nu) = h^c(x^*(\mu, \nu)), \tag{2.25b}$$

  que coincide con las expresiones proporcionadas tras la definición de la función dual en (2.18). El método dual se implementa siguiendo los pasos que se indican a continuación:

  1. Inicializar μ^t y ν^t para $t = 0$.
  2. Obtener el valor de la variable primal como

$$x^{t+1} = \arg\min_x \mathcal{L}(x, \mu^t, \nu^t). \tag{2.26}$$

     Este problema de optimización puede resolverse con cualquier método de resolución de problemas de optimización sin restricciones, por ejemplo un método *iterativo* de GD.

[2] Sea z un vector genérico y $\phi(z)$ una función genérica de z. Con un ligero abuso de notación, la expresión $\nabla_z \phi(z)$ se utilizará tanto para denotar el jacobiano como el gradiente. Si ϕ es una función vectorial, $\nabla_z \phi(z)$ será el jacobiano, esto es, una matriz cuyo número de filas corresponde con la dimensión de la imagen de ϕ y cuyo número de columnas se corresponde con la dimensión del dominio de ϕ. Así, la entrada (i, j) del jacobiano contendrá la derivada parcial de $\phi_i(z)$ con respecto a z_j. Por otro lado, si ϕ es una función escalar, $\nabla_z \phi(z)$ será el gradiente, esto es, un vector columna recolectando las derivadas parciales de ϕ con respecto a las entradas de z.

3. Obtener μ^{t+1}, ν^{t+1} como:

$$\mu^{t+1} = \text{máx}(0, \mu^t + \eta_{\mu,t} g^c(x^{t+1})) \tag{2.27a}$$
$$\nu^{t+1} = \nu^t + \eta_{\nu,t} h^c(x^{t+1}), \tag{2.27b}$$

donde $\eta_{\mu,t}$ y $\eta_{\nu,t}$ son los tamaños de paso en cada iteración. Nótese que, para los multiplicadores asociados a restricciones de desigualdad, se implementa un subgradiente proyectado, puesto que se necesita garantizar que $\mu \geq 0$.

4. Repetir 2) y 3) hasta convergencia.

- **Método primal-dual:** El método primal-dual es una técnica que realiza una búsqueda simultánea de las variables primales y duales. De forma general puede entenderse como un algoritmo que intenta encontrar un punto de silla de la función lagrangiana $\mathcal{L}(x, \mu, \nu)$. En cada iteración, se actualizan tanto las variables primales como las duales, minimizando sobre x y maximizando sobre (μ, ν). Los mayores beneficios de un método primal-dual están asociados a un menor coste computacional y mayor rapidez de convergencia. No obstante, su convergencia es más delicada que la de los métodos duales, exigiendo habitualmente propiedades adicionales a las funciones de trabajo y requiriendo una selección de los tamaños de paso más cuidadosa. Los métodos primales-duales son especialmente útiles en problemas de alta dimensionalidad donde se pueden aprovechar las estructuras específicas de la función objetivo y de las restricciones propuestas. Un método primal-dual se implementa siguiendo estos pasos:

  1. Inicializar x^t, μ^t y ν^t para $t = 0$.
  2. Actualización primal. Obtener x^{t+1} como:

$$x^{t+1} = x^t - \eta_{x,t} \nabla_x \mathcal{L}(x^t, \mu^t, \nu^t), \tag{2.28}$$

  donde $\eta_{x,t}$ es el tamaño de paso en cada iteración.

  3. Actualización dual. Obtener μ^{t+1} y ν^{t+1} como:

$$\mu^{t+1} = \text{máx}(0, \mu^t + \eta_{\mu,t} g^c(x^{t+1})) \tag{2.29a}$$
$$\nu^{t+1} = \nu^t + \eta_{\nu,t} h^c(x^{t+1}), \tag{2.29b}$$

  donde $\eta_{\mu,t}$ y $\eta_{\nu,t}$ son los tamaños de paso en cada iteración.

  4. Repetir 2) y 3) hasta convergencia.

  Nótese que la mayor diferencia entre el método primal-dual y el método dual está en el paso 2. Mientras que en el método primal-dual x^{t+1} se actualiza utilizando un simple paso por gradiente [cf. (2.28)], en el método dual se actualiza mediante la minimización exacta del lagrangiano [cf. (2.26)].

2.2. Fundamentos de la BLO

La BLO es un área interesante y compleja dentro del campo de la optimización matemática. Los problemas de optimización binivel son cruciales para modelar procesos de toma de decisiones jerárquicos donde las decisiones a tomar en un nivel superior influyen (impactan) en las decisiones a tomar en un nivel inferior. En un problema de optimización binivel, una tarea de optimización se encuentra anidada dentro de otra, reflejando escenarios del mundo real donde las decisiones se toman en diferentes niveles de una jerarquía. Esta estructura es particularmente relevante para problemas que involucran a múltiples actores, como un líder (nivel superior) cuyas decisiones influyen en un seguidor (nivel inferior), y viceversa. La interacción entre las decisiones de los líderes y las reacciones de los seguidores

refleja una complejidad inherente en la estructura de estos problemas. Además, es habitual el conflicto entre los objetivos de los distintos actores. Estos problemas son naturalmente complejos ya que incluso cuando los problemas en ambos niveles son convexos, el problema conjunto no lo es. Esto significa que no existe un algoritmo conocido en tiempo polinómico capaz de resolver todos los casos de problemas de BLO de manera eficiente. Las principales fuentes de esta complejidad incluyen la estructura anidada del problema y la no convexidad de las condiciones KKT, lo que lleva a la existencia de múltiples óptimos locales y dificulta la optimización global [37]. A lo largo de esta sección se formalizará matemáticamente el concepto de optimización binivel, analizando paso a paso cada uno de sus componentes. También se describirán los rasgos más importantes que caracterizan los problemas de BLO, permitiendo clasificarlos dentro de distintos grupos.

2.2.1. Definición y modelo básico

Un problema de optimización binivel se define formalmente como un problema de optimización que tiene otro problema de optimización como restricción. Puesto que se tienen dos niveles, pueden definirse dos tomadores de decisiones: el líder y el seguidor. Las decisiones del líder están representadas por la variable $x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$, mientras que las decisiones del seguidor están representadas por la variable $y \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$. Cada nivel tiene su propia función objetivo y restricciones. El líder busca minimizar su función objetivo $f : \mathbb{R}^{d_x \times d_y} \to \mathbb{R}$, mientras que el seguidor busca minimizar su función objetivo $g(x, y) : \mathbb{R}^{d_x \times d_y} \to \mathbb{R}$. Así, el problema de BLO puede formularse como:

$$\min_{x \in \mathcal{X}} \quad F(x) = f(x, y^*(x)) \tag{2.30a}$$

$$\text{s.a:} \quad y^*(x) = \arg\min_{y \in \mathcal{Y}} \{g(x, y) \quad \text{s.a: } g^c(x, y) \leq 0\} \tag{2.30b}$$

$$f^c(x, y) \leq 0, \tag{2.30c}$$

donde $f^c : \mathbb{R}^{d_x \times d_y} \to \mathbb{R}^{d_{fc}}$ y $g^c : \mathbb{R}^{d_x \times d_y} \to \mathbb{R}^{d_{gc}}$ definen las d_{fc} y d_{gc} restricciones en los niveles superior e inferior, respectivamente. En la figura 2.4 se ilustra la estructura de un problema de optimización binivel, donde para cada valor de las variables del nivel superior x se resuelve el nivel inferior sobre las variables y para encontrar la respuesta óptima. Es decir, para cada valor de las variables x cambiará el problema de optimización del nivel inferior. Así, se obtendrá una solución óptima para las variables y que será distinta para cada valor de x.

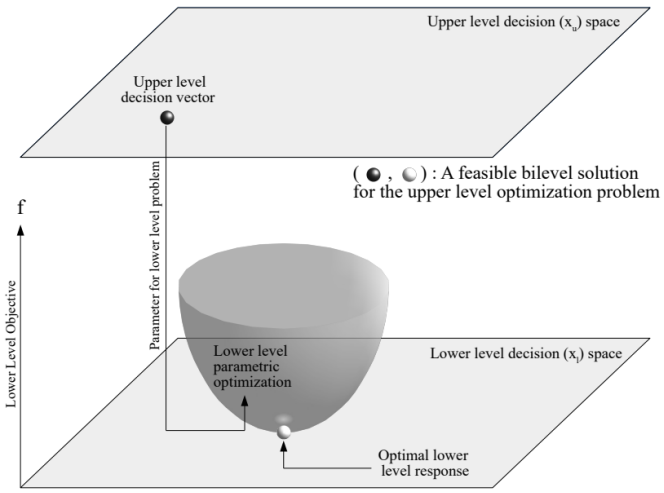


Figura 2.4: Ilustración del problema del nivel inferior en función de las variables de decisión del nivel superior. Figura tomada de [38].

Dependiendo de la convexidad y propiedades del problema en el nivel inferior $g(x, y)$, la solución $y^*(x)$ puede ser única o múltiple. En el caso de que existan varias soluciones para el nivel inferior, se necesita un criterio para identificar la solución final. En definitiva, en función del número de soluciones del nivel inferior y, en caso de que haya más de una, la solución que se escoja, se pueden distinguir tres versiones del problema BLO: optimista, pesimista y *singleton*.

- **Enfoque optimista:** Cuando el problema del nivel inferior $g(x, y)$ carece de convexidad estricta, no es posible asumir que exista un único y^* que alcance el valor mínimo de $g(x, y^*)$ para un x dado. Dicho en otras palabras, si el óptimo del problema inferior no es único, existe un conjunto de soluciones óptimas, denotado como $\Psi(x) \subseteq \mathcal{Y}$. Esto supone un desafío para la optimización del nivel superior, puesto que al evaluar la calidad de la variable x la función $f(x, y^*(x))$ puede tomar valores distintos en función de la solución $y^*(x) \in \Psi(x)$ que se escoja. Para eliminar la incertidumbre, en el enfoque optimista se asume que, entre los valores de y que pertenecen a $\Psi(x)$ se escoge como $y^*(x)$ el que minimiza la función objetivo del nivel superior. Esta suposición es razonable en escenarios en los que el actor del nivel inferior está dispuesto a ayudar al actor del nivel superior. Matemáticamente, este enfoque supone que el problema se formula como

$$\min_{x \in \mathcal{X}} \; F(x) = f(x, y^*(x)) \tag{2.31a}$$
$$\text{s.a: } y^*(x) = \arg \min_{y \in \Psi(x)} f(x, y)$$
$$\Psi(x) = \arg \min_{y \in \mathcal{Y}} \{g(x, y) \quad \text{s.a: } g^c(x, y) \leq 0\} \tag{2.31b}$$
$$f^c(x, y) \leq 0, \tag{2.31c}$$

  donde, tal y como se ha explicado, $\Psi(x)$ es el conjunto de soluciones óptimas al problema del nivel inferior para un x determinado.

- **Enfoque pesimista:** Al igual que en el caso anterior, nos enfrentamos a un caso en el que el minimizador del problema inferior para un x dado no es único. La diferencia en este caso es que se asume que el actor del nivel inferior escoge como solución $y^*(x)$ la que resulta más dañina para el nivel superior, es decir, la que maximiza $f(x, y^*(x))$. Matemáticamente, este comportamiento se modela como

$$\min_{x \in \mathcal{X}} \; F(x) = f(x, y^*(x)) \tag{2.32a}$$
$$\text{s.a: } y^*(x) = \arg \max_{y \in \Psi(x)} f(x, y)$$
$$\Psi(x) = \arg \min_{y \in \mathcal{Y}} \{g(x, y) \quad \text{s.a: } g^c(x, y) \leq 0\} \tag{2.32b}$$
$$f^c(x, y) \leq 0, \tag{2.32c}$$

- ***Singleton:*** Cuando el problema inferior tiene una solución única (esto ocurre, por ejemplo, si la función $g(x, y)$ es estrictamente convexa en y), el conjunto de soluciones del problema del nivel inferior $\Psi(x)$ vendrá dado por una única solución $y^*(x)$. En este caso no hay ambigüedad posible y el problema puede mantener su formulación original [cf. (2.30)]. Este TFG se centra en problemas BLO que se clasifican dentro de este grupo.

Como ya se ha mencionado, una dificultad inherente a los problemas de BLO es que, aunque el problema del nivel inferior sea estrictamente convexo (de forma que $y^*(x)$ es única y se puede encontrar de forma sencilla) y la función $f(x, y)$ sea conjuntamente convexa en (x, y), en general no es cierto que la función $f(x, y^*(x))$ sea convexa en x. A modo de ejemplo, la figura 2.5 refleja la evolución de la función objetivo del nivel inferior para distintos valores de x, a la vez que pone de manifiesto la no convexidad del nivel superior. Por ello, las próximas secciones discuten varias metodologías para abordar problemas de BLO, incluyendo técnicas que los transforman en un problema de un único nivel.

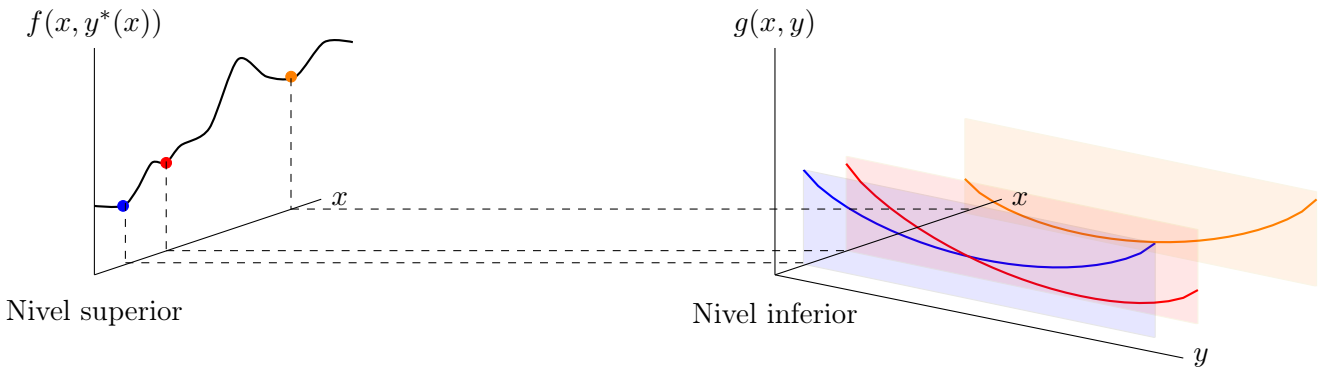


Figura 2.5: Evolución del nivel inferior en función de las variables del nivel superior y función objetivo en el nivel superior.

2.3. Métodos para la solución de problemas de BLO

Dada la complejidad de los problemas de optimización binivel, en la literatura se han desarrollado diversas estrategias (métodos) de solución. Las estrategias clásicas se han utilizado para abordar problemas con una estructura matemática favorable. Estas estrategas incluyen métodos basados en las condiciones KKT, GD y funciones de penalización. Por otra parte, en los últimos años se han desarrollado algoritmos evolutivos que permiten lidiar con problemas más complejos, habitualmente con variables discretas o categóricas, aunque no proporcionan garantías de optimalidad. A continuación se describen varias estrategias, poniendo mayor énfasis en las más relevantes para el desarrollo de este TFG.

Reducción a problema de un único nivel

Cuando se aborda un problema de BLO cuyo nivel inferior tiene estructura (especialmente cuando es convexo), el problema del nivel inferior se puede reemplazar por sus condiciones KKT. Esto transforma el problema de BLO en un problema de optimización de un único nivel con un mayor número de variables y restricciones. Más concretamente, en el nuevo problema de un único nivel: i) las condiciones KKT del nivel inferior (tanto las asociadas al lagrangiano como las restricciones de complementariedad) se incorporan como restricciones y ii) se optimiza conjuntamente sobre x, y y los multiplicadores asociados al nivel inferior. Si bien esto da lugar a un problema de un único nivel en el que se pueden utilizar técnicas de optimización clásicas, es importante tener en cuenta que el problema de nivel único resultante no suele ser fácil de resolver. De hecho, salvo cuando el problema del nivel inferior es cuadrático, las condiciones KKT asociadas al nivel inferior exigen que funciones no lineales (provenientes de los gradientes de g y g^c) sean iguales a cero, dando lugar a restricciones de igualdad no lineales que, por definición, no son convexas. Además, las condiciones de complementariedad del nivel inferior, debido a su naturaleza combinatoria, añaden otra capa de dificultad para la resolución del problema de un único nivel, requiriendo en muchos casos la introducción de variables enteras.

En el caso específico de los problemas de BLO con estructura cuadrática-cuadrática o lineal-cuadrática, la restricción asociada al lagrangiano del nivel inferior también permanece lineal. Como resultado, el problema de optimización de un único nivel se convierte en un programa lineal de enteros mixtos (MILP, por sus siglas en inglés). Para resolver estos problemas, se han utilizado técnicas como los métodos de *branch&bound* (B&B) [39, 40, 41]. No obstante, debe tenerse en cuenta que la complejidad de los métodos B&B escala exponencialmente con el número de variables enteras involucradas. Por este motivo, no es posible aplicar esta estrategia a problemas reales que involucren gran cantidad de variables y restricciones.

Métodos de descenso

Tal y como se presentó en la sección 2.1.3, para la mayoría de funciones objetivo, aplicar una dirección de descenso iterativamente en el proceso de optimización converge a, al menos, un mínimo local. Consideremos en primer lugar un problema binivel sin restricciones y recordemos que, de acuerdo a la definición en (2.30a), $F(x)$ representa el valor de la función objetivo del nivel superior cuando se fija x y se evalúa en $y^*(x)$. Al no haber restricciones, el gradiente de $F(x)$ puede utilizarse como dirección de descenso para la búsqueda de un x óptimo. Específicamente, teniendo en cuenta que $F(x) = f(x, y^*(x))$ y aplicando la regla de la cadena, el gradiente $\nabla_x F(x) \in \mathbb{R}^{d_x}$ tiene la expresión

$$\nabla_x F(x) = \nabla_x f(x, y^*(x)) + \langle \nabla_x (y^*(x)), \nabla_y f(x, y^*(x)) \rangle, \tag{2.33}$$

donde $\nabla_x(y^*(x))$ es el jacobiano de la función vectorial $y^*(x)$ con respecto a x. Nótese que el segundo término en (2.33) viene de la aplicación de la regla de la cadena, multiplicando una matriz de tamaño $d_y \times d_x$ (jacobiano) por un vector de tamaño $d_y \times 1$ (gradiente). En la mayoría de los casos, no existe una expresión cerrada para $y^*(x)$, por lo que el valor de sus derivadas parciales (i.e., de las entradas de $\nabla_x(y^*(x))$ no puede obtenerse utilizando métodos de derivación explícita. No obstante, puesto que la función $y^*(x)$ se corresponde con la solución de las ecuaciones optimalidad del nivel inferior, las derivadas parciales de $y^*(x)$ pueden obtenerse mediante el teorema de derivación implícita. Más específicamente, al estar considerando que no existen restricciones en el nivel inferior, las condiciones de optimalidad que definen $y^*(x)$ son simplemente $\nabla_y g(x, y^*) = 0$ y, por consiguiente, aplicando el teorema de derivación implícita se obtiene que

$$\nabla_x(y^*(x)) = \left(\nabla^2_{yy} g(x, y^*(x))\right)^{-1} \nabla^2_{yx} g(x, y^*(x)), \tag{2.34}$$

donde $\nabla^2_{yy} g(x, y)$ y $\nabla^2_{yx} g(x, y)$ son bloques de la matriz hessiana de la función $g(x, y)$. Sustituyendo (2.34) en (2.33) se obtiene

$$\nabla_x F(x) = \nabla_x f(x, y^*(x)) + \langle \left(\nabla^2_{yy} g(x, y^*(x))\right)^{-1} \nabla^2_{yx} g(x, y^*(x)), \nabla_y f(x, y^*(x)) \rangle. \tag{2.35}$$

Este método se conoce como GD implícito (IGD, por sus siglas en inglés) y es uno de los métodos más populares en BLO [42, 43].

Cuando en el nivel inferior hay restricciones (supongamos, por sencillez y sin pérdida de generalidad, que únicamente las del tipo $g^c(x, y) \leq 0$), la obtención del jacobiano de $y^*(x)$ con respecto a x es algo más complicada, puesto que hay que tener en cuenta el valor de los multiplicadores de Lagrange del nivel inferior, μ, y asegurar que la solución $y^*(x)$ sea factible. Los dos pasos clave para este caso son: i) definir un vector aumentado para el nivel inferior que considere tanto las variables primales como las variables duales y ii) aplicar el teorema de la función implícita utilizando las condiciones KKT [44]. Por lo tanto, en problemas binivel con restricciones en el nivel inferior la dirección de descenso viene determinada por

$$\nabla_x F(x) = \nabla_x f(x, y^*(x)) + \langle \nabla_x (y^*(x)), \nabla_y f(x, y^*(x)) \rangle, \tag{2.36}$$

donde el jacobiano $\nabla_x(y^*(x))$ se obtiene tal y como se explica a continuación. En primer lugar, se escriben las condiciones KKT del nivel inferior

$$\begin{pmatrix} \nabla_y \mathcal{L}_g(x, y^*, \mu^*) \\ g^c_+(x, y^*) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{2.37}$$

donde $\mathcal{L}_g(x, y, \mu)$ es el lagrangiano del problema del nivel inferior y $g^c_+(x, y)$ es el conjunto de restricciones activas del problema del nivel inferior. En segundo lugar, se aplica el teorema de derivación implícita al sistema de ecuaciones en (2.37), derivando con respecto al vector (y^*, μ^*). Esto da lugar a

$$\begin{pmatrix} \nabla_x(y^*(x)) \\ \nabla_x(\mu^*(x)) \end{pmatrix} = \begin{pmatrix} \nabla^2_{yy} \mathcal{L}_g(x, y^*(x), \mu^*(x)) & \nabla_y g^c_+(x, y^*(x)) \\ \nabla_y (g^c_+(x, y^*(x)))^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla^2_{yx} \mathcal{L}_g(x, y^*(x), \mu^*(x)) \\ \nabla_x g^c_+(x, y^*(x)) \end{pmatrix}, \tag{2.38}$$

El tercer y último paso consiste en sustituir expresión de asociada a las d_y primeras filas del bloque superior de la matriz resultante en (2.38) en la ecuación (2.36).

Cuando el problema del nivel inferior no sea diferenciable, el cálculo de los gradientes y hessianos en (2.38) no es posible. No obstante, también existen técnicas para lidiar con problemas no diferenciables. Un ejemplo es el algoritmo recientemente propuesto en [45] que propone en primer lugar una aproximación a $y^*(x)$ y, posteriormente, utilizan las fórmulas en (2.38).

Independientemente de la técnica utilizada, es importante tener en cuenta que el uso del teorema de la función implícita implica invertir una matriz que: i) para que exista, la función objetivo deber ser dos veces diferenciable, ii) solo es de rango completo bajo ciertos requisitos, y iii) puede ser de un tamaño muy considerable. Esto supone que, en un amplio abanico de casos, sea necesario recurrir a otro tipo de métodos para estimar bien el valor de $\nabla_x(y^*(x))$ o bien directamente el valor de $\nabla_x F(x)$. Algunos de estos métodos se discuten de manera breve a continuación.

Métodos de función de penalización

Tal y como se explicó en el caso de optimización de un único nivel con restricciones, los métodos de penalización definen un nuevo problema en el que el objetivo original se combina con un nuevo término, que siempre es mayor o igual que cero y que está relacionado con la restricción. Si la restricción se satisface, la penalización vale cero. En caso contrario la penalización se activa, tomado mayores valores cuanto mayor sea el nivel de violación de las restricciones. Para el caso de la BLO, se considera un escenario sin restricciones y se denota como γ al coeficiente que refleja el peso de la penalización. El problema penalizado de un solo nivel en dicho escenario es

$$\min_{x,y} f(x,y) + \gamma P(x,y), \tag{2.39}$$

donde la función de penalización $P(x,y)$ está asociada al problema del nivel inferior. En la literatura existen varias opciones para definir esta función de penalización [13, 46, 47]. Para el caso concreto en el que el problema inferior no tiene restricciones, una función de penalización común es $p(x,y) = \|\nabla_y g(x,y)\|^2$. De este modo, si el valor de γ es muy elevado, la solución de (2.39) tenderá a hacer que $\nabla_y g(x,y) \to 0$, garantizando que se satisfagan las condiciones de optimalidad del nivel inferior. Naturalmente, la norma 2 en la expresión anterior puede sustituirse por cualquier otra norma. Asimismo, si el problema en el nivel inferior tiene restricciones, la penalización se aplicará sobre un vector que contenga el gradiente del lagrangiano del nivel inferior, así como el resto de las condiciones KKT. En el contexto de métodos de penalización para BLO, también es común definir la función de penalización como $p(x,y) = g(x,y) - v(x)$, donde $v(x)$ es una aproximación de $\min_y g(x,y)$ conocida como *función de valor*. De esta forma, $p(x,y) = 0$ sólo si $y = \arg\min_z g(x,z)$, siendo mayor que cero en cualquier otro caso. Al igual que cuando se penaliza la norma del gradiente, existen algoritmos que modifican la función de penalización utilizada (por ejemplo, aplicando una transformación no lineal y creciente a $g(x,y) - v(x)$), de forma que se pueda modular la sensibilidad del algoritmo a la violación de la condición de optimalidad. El algoritmo diseñado en este TFG pertenece a la familia de métodos penalizados utilizando la función de valor, por lo que varios de estos aspectos se discutirán en más detalle en el capítulo 3.

Algoritmos evolutivos

Los algoritmos evolutivos han emergido como una estrategia popular para resolver problemas de BLO, principalmente debido a su capacidad para manejar problemas complejos con variables discretas y carecientes de estructura matemática favorable. Dentro de este grupo de algoritmos, destacan los métodos evolutivos anidados [48, 49], donde se realiza una búsqueda exhaustiva en el nivel superior y se resuelve el problema de optimización de nivel inferior para cada solución explorada. Sin embargo, estos métodos son computacionalmente muy costosos e inviables para problemas de gran escala.

Otro tipo de algoritmos evolutivos ampliamente utilizados en el contexto de BLO son los métodos de metamodelado [50]. Estos métodos son comunes en problemas de optimización donde la evaluación de la función objetivo original es costosa. Un metamodelo es una aproximación simplificada del modelo real, entrenado a partir de una pequeña muestra del mismo. En optimización binivel, el metamodelado, especialmente cuando se usa con algoritmos basados en poblaciones, ayuda a manejar la complejidad inherente de estos problemas. Se pueden aplicar técnicas iterativas para aproximar localmente el modelo real durante las iteraciones. Dos enfoques clave incluyen la aproximación del conjunto de reacciones [51, 52, 53] y la función de valor óptimo del nivel inferior [54], que pueden reducir el problema binivel a uno de nivel único.

2.4. Revisión histórica del estado del arte

Una vez descritos en detalle los elementos que componen un problema de BLO y las técnicas más habituales para abordarlos, se procede a realizar una revisión histórica del estado del arte. Este análisis proporciona un contexto sobre la evolución y el estado actual de los métodos de solución para problemas de BLO.

La primera formulación de un problema de optimización binivel se remonta a más de 50 años atrás [2]. Sin embargo, los trabajos sobre juegos líder-seguidor de Von Stackelberg [55, 1] ya identifican los principales elementos de los problemas BLO. Una discusión temprana sobre problemas binivel se encuentra en [56], donde se reconoce la dificultad intrínseca de estos problemas y sus diferencias con los problemas de un solo nivel. Los autores de [56] observaron que incluso con variables continuas y funciones objetivo y restricciones lineales, el conjunto factible de los problemas binivel puede ser no convexo y disconexo. De hecho, los resultados formales de complejidad indican que esta clase simplificada de problemas binivel es fuertemente NP[3]-difícil . En [56] también se propuso uno de los primeros algoritmos para resolver problemas BLO lineales, similar al método símplex. Estudios posteriores, como [57], profundizaron en este tipo de algoritmos.

En 1981, [40] diseñó un nuevo método para problemas binivel convexos-cuadráticos. La estructura cuadrática del nivel inferior da lugar a condiciones KKT lineales, habilitando la sustitución del problema de optimización inferior por un sistema de ecuaciones lineales. No obstante, dado que solo se deben considerar las restricciones activas, es necesario introducir variables binarias para seleccionar las restricciones activas. De esta forma, el método de [40] resulta en un problema MIP de un solo nivel que puede ser abordado por *solvers* estándar. Otros trabajos, como [41, 58, 59, 60], profundizaron en esta metodología, siendo aún hoy en día una estrategia popular para resolver problemas de BLO convexos en el nivel superior y cuadráticos en el nivel inferior. Posteriormente, [61, 62, 63, 64, 65] desarrollaron las condiciones de optimalidad de primer y segundo orden para programas binivel que involucran problemas sin restricciones convexos-cuadráticos.

Además de los enfoques basados en KKT, se han propuesto varios métodos de descenso para resolver problemas de optimización binivel. Los trabajos en [44, 66] adaptaron la noción de descenso más pronunciado al caso de los programas binivel. Recientemente, los enfoques basados en descenso por gradiente han sido objeto de estudio. Una línea de trabajo se centra en utilizar variantes del método de IGD, donde la idea clave es calcular (aproximar) el gradiente del problema BLO (referido en la literatura como hipergradiente) mediante el teorema de la función implícita –véase [42], así como la discusión en torno a la ecuación (2.35) proporcionada en esta memoria–, evitando el cálculo explícito de la solución del nivel inferior y facilitando por tanto la optimización del problema del nivel superior. IGD se ha utilizado principalmente para problemas BLO sin restricciones [29, 43, 67, 68, 69], aunque también existen trabajos para resolver problemas de BLO con restricciones desacopladas [70, 71, 72]. En cuanto al tiempo de cómputo, la convergencia en tiempo finito de los métodos de IGD se estableció por primera vez en [43] para problemas de nivel inferior fuertemente convexos y sin restricciones. Trabajos posteriores, como [67, 68, 69, 73, 74, 75, 76, 77, 78, 79], mejoraron las tasas de convergencia, en gran

[3]NP: *Nondeterministic polynomial time.*

parte de los casos, realizando suposiciones adicionales con respecto a la estructura y propiedades del problema.

En cuanto a enfoques penalizados, los primeros intentos se produjeron en los años 80 en los estudios realizados en [13, 46]. Estos trabajos reemplazan el problema del nivel inferior por un término de penalización en la función objetivo del nivel superior, obteniendo un problema de optimización de un solo nivel [80, 81, 82, 83, 84]. El término de penalización se introduce con el objetivo de asegurar la optimalidad del problema del nivel inferior. Otros estudios que se basan en el uso de funciones de penalización para problemas BLO lineales son [85, 86, 72, 87, 88].

Aunque se han logrado avances significativos para problemas de BLO sin restricciones, el análisis para BLO con restricciones es más limitado. Las restricciones de nivel superior de la forma (2.30c) fueron consideradas en [68, 73, 89]. Para BLO con restricciones desacopladas en el nivel inferior, se han propuesto algoritmos en [70, 71, 72]. Sin embargo, la literatura sobre BLO con CCs es escasa, a pesar de que estos problemas modelan muchos escenarios reales. En los últimos años, se ha suscitado un interés creciente en estos problemas, como se observa en estudios recientes [90, 91, 92, 93]. Los detalles de BLO con CCs se profundizarán en la sección introductoria del capítulo 3.

Paralelamente, se han desarrollado algoritmos evolutivos para abordar problemas de optimización con una estructura matemática desfavorable debido a su capacidad para afrontar problemas complejos. Algunos trabajos se basan en métodos de enjambre [94], mientras que otros utilizan metamodelos para resolver una versión simplificada y tratable del problema original [95].

La revisión realizada es, inevitablemente, limitada y sesgada por la temática de este TFG. Los lectores interesados en una descripción más completa del estado del arte de la BLO son remitidos a los tutoriales y revisiones realizados en [38, 96, 97].

Capítulo 3

Diseño y caracterización del algoritmo BLOCC

Como ya se ha explicado, la BLO puede entenderse como una resolución jerárquica de dos problemas de optimización en el que un problema de optimización (el problema del nivel inferior) está anidado dentro de otro problema de optimización (el problema del nivel superior). En las páginas precedentes han realizado una revisión de los trabajos más relevantes en el ámbito de la BLO, detallando también la evolución histórica del estado del arte. Dicha revisión ha dejado patente que los esquemas existentes se centran en problemas binivel bien sin restricciones o bien con restricciones no acopladas. Las CCs implican la presencia de restricciones en el nivel inferior que dependen tanto de las variables del nivel superior como de las variables del nivel inferior, añadiendo complejidad al problema. No obstante, en un gran número de aplicaciones reales, las variables del nivel superior e inferior presentan dependencias que deben ser tenidas en cuenta. El hecho de que los problemas BLO con CCs sean relevantes desde un punto de vista práctico y, al mismo tiempo, difíciles de resolver desde un punto de vista analítico justifica el diseño de algoritmos especializados para este tipo de problemas. En este capítulo se presenta un nuevo algoritmo para lidiar con problemas de BLO con CCs mediante una formulación penalizada de un único nivel.

La estructura de este capítulo está organizada de la siguiente manera. En la sección 3.1 se revisa la literatura existente en el ámbito de BLO con CCs. En la sección 3.2 se presenta la formulación del problema BLO con CCs en el nivel inferior. El valor de la función objetivo del problema del nivel superior de esta sección (cuando se resuelve de forma óptima el problema del nivel inferior) se denota como $F(x)$. Además, aprovechando el teorema de dualidad de Lagrange, se: i) introduce la función $F_\gamma(x)$ como una reformulación del problema presentado al inicio de la sección basada en una función de penalización y ii) cuantifica la relación entre $F(x)$ y $F_\gamma(x)$, demostrando su equivalencia matemáticamente. En la sección 3.3 se introduce y caracteriza BLOCC, el algoritmo de primer orden propuesto en este TFG para abordar problemas de BLO con CCs. El algoritmo BLOCC se centra en minimizar $F_\gamma(x)$. Para ello, se establece la continuidad y suavidad de Lipschitz de $F_\gamma(x)$ bajo ciertas suposiciones y se desarrolla un esquema novedoso para calcular su gradiente. Asimismo, se caracteriza de forma teórica la convergencia del algoritmo propuesto y, además, se acota su complejidad computacional cuando las CCs son convexas o afines. Finalmente, la sección 3.4 realiza una revisión del algoritmo presentado y lo aplica a un ejemplo sencillo, con el fin de ganar intuición e ilustrar mejor su comportamiento.

3.1. Introducción

El principal reto cuando se resuelven problemas de BLO es el acoplamiento existente entre los problemas de nivel superior e inferior. Para el tipo de problemas binivel estudiados en este capítulo, dicho acoplamiento tiene lugar tanto a través de la función objetivo como a través de restricciones

del nivel inferior que involucran a variables de ambos niveles de forma simultánea. La presencia de estas restricciones imposibilita la aplicación directa de los métodos clásicos de BLO, requiriendo bien la adaptación de los mismos o bien el diseño de nuevos métodos. Esta sección realiza una breve revisión de los trabajos existentes en el ámbito de BLO con CCs y sirve de motivación para la formulación que se propone en la sección 3.2.

Métodos anteriores han empleado técnicas de descenso por IGD y funciones de penalización para resolver problemas de BLO. Sin embargo, IGD se ha utilizado principalmente para problemas de BLO sin restricciones [29, 43, 67, 68, 69] o con restricciones desacopladas [70, 71, 72]. A pesar de los avances alcanzados por los métodos IGD en problemas de BLO sin restricciones o restricciones desacopladas, su uso en problemas con CCs enfrenta desafíos significativos. El método AiPOD (*Alternating implicit projected stochastic gradient descent*) [91] abordó la resolución de problemas de BLO con CCs, pero exclusivamente de igualdad, lo que limita su aplicabilidad a una gama más amplia de problemas BLO. El método de minimización alternada generalizada, GAM (*Gradient approximation method*), desarrollado en [45] está basado en técnicas de IGD; véase (2.38). Aunque más versátil que los anteriores, el método carece de garantías de convergencia en tiempo finito. Esto significa que, si bien puede aplicarse a una variedad más amplia de problemas, no hay garantía de que encuentre una solución óptima en un período de tiempo razonable. La falta de garantías de convergencia supone una gran desventaja, especialmente en aplicaciones prácticas donde es crítico encontrar una solución óptima.

Por otro lado, los métodos basados en funciones de penalización también son comunes para reformular problemas de BLO sin restricciones mediante problemas de un único nivel [80, 81, 82, 83, 84]. Sin embargo, estos métodos enfrentan dificultades cuando se aplican a problemas de BLO con CCs, ya que estas implican una fuerte interdependencia entre ambos niveles. Por el momento, la literatura existente en la reformulación de problemas de BLO con CCs haciendo uso de funciones de penalización se reduce a [92]. En este trabajo se desarrolla LV-HBA (*Lagrangian value function-based hessian-free bilevel algorithm*), un método que no necesita calcular el Hessiano del nivel inferior y que ofrece garantías convergencia en tiempo finito. Este método evita el coste computacional de invertir la matriz Hessiana en (2.38), siendo más eficiente que GAM [45] en la mayoría de escenarios. Sin embargo, en cada una de las iteraciones del algoritmo LV-HBA es necesario calcular una proyección al espacio factible del nivel inferior, es decir, en cada iteración la estimación de la solución debe moverse al punto más cercano del conjunto de factibilidad definido por las CCs. Este es un proceso que es intensivo en términos computacionales y difícil de realizar con precisión, especialmente a medida que aumenta el número de variables y restricciones, lo que limita su aplicabilidad, particularmente cuando se trata de problemas con una alta dimensionalidad o cuando las restricciones no son conjuntamente convexas.

En resumen, aunque en la literatura existen varios métodos para abordar problemas BLO con CCs, todos ellos presentan ciertas limitaciones. Los retos asociados al acoplamiento entre variables en las restricciones y a la complejidad computacional continúan siendo obstáculos significativos, lo que motiva el desarrollo de algoritmos más eficientes y robustos para resolver de manera efectiva problemas de BLO con CCs. Desde esta perspectiva, el objetivo en este capítulo es desarrollar un algoritmo eficiente que resuelva los problemas de BLO con CCs, evitando realizar proyecciones conjuntas de las variables de los niveles superior e inferior sobre dichas restricciones.

En la siguiente sección se formaliza el problema de BLO con CCs y se propone una reformulación de dicho problema que evita, por un lado, el cómputo de la inversa del Hessiano como en GAM y, por otro lado, la implementación de proyecciones conjuntas sobre las CCs como en LV-HBA, dando lugar a un esquema de optimización más robusto y eficiente. En la tabla 3.1 se recogen los algoritmos para resolver problemas de BLO con CCs mencionados en esta sección. Para cada uno de ellos, se mencionan los requisitos exigidos a las CCs para garantizar la convergencia del algoritmo, así como la complejidad computacional del mismo en función del parámetro de precisión ϵ. La complejidad computacional se expresa mediante la cota superior asintótica[1] $\mathcal{O}$. La utilización del parámetro ϵ es habitual en el ámbito

[1]Formalmente, se dice que una función $\varphi(n)$ está en $\mathcal{O}(f(n))$ si existe una constante positiva c tal que se cumple $\varphi(n) \leq cf(n)$ para cualquier valor de n. De forma más intuitiva, la cota superior asintótica $\mathcal{O}(f(n))$ es una función que

de la optimización para determinar la distancia entre la solución a la que converge el algoritmo y la solución óptima al problema que se desea resolver. Como se observa, el algoritmo BLOCC propuesto en este TFG permite resolver una gama más amplia de problemas de BLO con CCs, puesto que impone unos requisitos más generales sobre las propiedades matemáticas de las CCs $g^c(x, y)$.

Algoritmo	Restricciones en el nivel inferior	Primer orden	Complejidad
BLOCC	$g^c(x, y) \leq 0$ convexo en y y LICQ	Sí	$\mathcal{O}(\epsilon^{-2.5})$;
LV-HBA [92]	$g^c(x, y) \leq 0$ conjuntamente convexo en $x \times y$	Sí	$\mathcal{O}(\epsilon^{-3})$
GAM [45]	$g^c(x, y) \leq 0$ conjuntamente convexo en $x \times y$ y LICQ	No	-
BVFSM [90]	$g^c(x, y) \leq 0$ con varios requisitos	Sí	-
AiPOD [91]	$g^c(x, y) = Ay - b(x) = 0$	No	$\mathcal{O}(\epsilon^{-1.5})$

Tabla 3.1: Comparación del algoritmo propuesto en este TFG (BLOCC) con algoritmos existentes en el estado del arte: LV-HBA [92], BVFSM (*Bilevel value function based sequential minimization*) [90], AiPOD [91] y GAM [45]. Para cada algoritmo se presenta su orden y complejidad computacional en función del parámetro de precisión ϵ, que hace referencia a la distancia entre la solución óptima al problema de optimización y la solución a la que converge el algoritmo. Los algoritmos GAM y BVFSM no ofrecen garantías de convergencia en tiempo finito.

3.2. Reformulación penalizada del problema BLO con restricciones

Esta sección comienza formalizando el problema de BLO con CCs en (3.1). Posteriormente, el problema se reformula aumentando el objetivo con: i) un término de penalización asociado con el valor óptimo del problema del nivel inferior y ii) un término lagrangiano asociado con las CCs. El resultado de esta reformulación es un problema que puede relacionarse de forma rigurosa con el problema original (3.1) y que, al mismo, tiempo presenta una estructura más favorable. Esta estructura se aprovechará en la secciones siguientes para diseñar un algoritmo de GD con garantías de convergencia.

El problema de BLO con CCs en el nivel inferior que se aborda en este capítulo se formula como

$$\min_{x \in \mathcal{X}} \quad F(x) = f(x, y^*(x)) \tag{3.1a}$$

$$\text{s.a.:} \qquad y^*(x) = \arg\min_{y \in \mathcal{Y}} \big\{ g(x, y) \quad \text{s.a: } g^c(x, y) \leq 0 \big\}, \tag{3.1b}$$

donde $f(x, y)$, $g(x, y)$ y $g^c(x, y)$ son la función objetivo del nivel superior, la función objetivo del nivel inferior y las CCs del nivel inferior, respectivamente. Las restricciones no acopladas en el nivel superior se representan como $x \in \mathcal{X}$ y las del nivel inferior como $y \in \mathcal{Y}$. Por simplicidad, el método que se presenta en este capítulo no entrará en detalles sobre cómo satisfacer dichas restricciones, pudiendo utilizarse multitud de esquemas de optimización de un único nivel para tales efectos.

Es importante destacar que para poder establecer la relación entre el problema original (3.1) y la versión reformulada (3.2), deberán realizarse una serie de suposiciones. Aunque estas serán enunciadas de forma rigurosa más adelante, sí cabe señalar que los resultados presentados son válidos para escenarios donde: i) el objetivo de nivel inferior, $g(x, y)$, es fuertemente convexo en y y ii) las restricciones $g^c(x, y)$ son convexas en y.

Específicamente, siendo $\gamma > 0$ el valor de la constante de penalización y denotando $\mu \in \mathbb{R}^{d_c}_{\geq 0}$ los multiplicadores de Lagrange de las restricciones de nivel inferior en (3.1), se propone la formulación penalizada

describe el crecimiento máximo de otra función $\varphi(n)$ a medida que su argumento n tiende a infinito, es decir, se cumple que $\varphi(n) \leq cf(n)$ cuando n tiende a infinito. Esta notación es esencial en el análisis de algoritmos ya que proporciona un límite superior sobre el tiempo de ejecución, permitiendo evaluar su eficiencia.

$$\min_{x\in\mathcal{X}} \quad F_\gamma(x) := \max_{\mu\in\mathbb{R}^{d_c}_{\geq 0}} \min_{y\in\mathcal{Y}} f(x,y) + \underbrace{\gamma(g(x,y)-v(x))}_{\text{término de penalización}} + \underbrace{\langle \mu, g^c(x,y)\rangle}_{\text{término de Lagrange}} \tag{3.2a}$$

$$\text{donde} \quad v(x) := \min_{y\in\mathcal{Y}} g(x,y) \;\; \text{s. a: } y \in \mathcal{Y}(x) := \{y :\; g^c(x,y) \leq 0\}, \tag{3.2b}$$

donde, para simplificar la notación, se ha definido el conjunto factibilidad asociado a las CCs como $\mathcal{Y}(x)$. Tal y como se ha indicado, (3.2) considera dos nuevos términos en la función objetivo. En primer lugar el término de penalización, que está formado por el *optimality gap* del nivel inferior $g(x,y)-v(x)$ multiplicado por la constante γ. Nótese que la definición de la función $v(x)$, conocida como función de valor, se proporciona en (3.2b) y garantiza la satisfacción de las restricciones. Asimismo, debido a las suposiciones de suavidad de Lipschitz sobre las funciones de trabajo, la penalización explícita sobre el valor de la función $(g(x,y)-v(x))$ puede entenderse también como una penalización implícita sobre la distancia $\|y-y_g^*(x)\|$, donde $y_g^*(x)$ denota el valor de y que resuelve la minimización en (3.2b) para un x dado. El segundo término, $\langle \mu, g^c(x,y)\rangle = \sum_{i=1}^{d_c} \mu_i g_i^c(x,y)$, penaliza la violación de las CCs.

La formulación en (3.2) presenta dos diferencias con respecto al estado del arte. En primer lugar, con respecto a métodos de BLO sin CCs, la definición de la función de valor es diferente, puesto que incluye las restricciones. En segundo lugar, con respecto a los métodos de BLO con CCs, la formulación propuesta, gracias al término de Lagrange en el objetivo, permite diseñar algoritmos que no hagan uso del gradiente implícito ni tengan que proyectar la variable y para que cumpla las CCs. La importancia de estos dos aspectos se discute con mayor detalle a continuación.

1. Encontrar una función de penalización adecuada para problemas de BLO con CCs es más complejo que para el caso sin restricciones. Nótese que la clave para poder utilizar métodos de GD no es tener acceso a la función de valor $v(x)$, sino a su gradiente. Para problemas de BLO sin restricciones, la expresión de $\nabla v(x)$ es igual a $\nabla_x g(x, y_g^*(x))$ como se demuestra en [71, 72, 83, 84]. Sin embargo, esta expresión no es válida cuando hay CCs involucradas. De forma intuitiva, el nuevo gradiente deberá tener en cuenta el valor de los multiplicadores de Lagrange (que miden cómo de sensible es el valor óptimo ante un cambio infinitesimal de la restricción) así como la variación del valor de la restricción con respecto a x. Para ilustrar cómo de diferentes pueden ser estos gradientes, considérese un sencillo ejemplo donde $g(x,y) = (y-2x)^2$ y $g^c(x,y) = 3x-y$, la solución óptima es $y_g^*(x) = 3x$, siendo $v(x) = x^2$ con $\nabla v(x) = 2x$. Por el contrario, el gradiente calculado usando la expresión para BLOs sin restricciones es $\nabla_x g(x, y_g^*(x)) = -4x$, lo que no coincide con $\nabla v(x)$. Esto se representa de forma gráfica en la figura 3.1, donde se compara el gradiente $\nabla v(x)$ calculado según $\nabla_x g(x, y_g^*(x))$ frente al valor real del gradiente. La figura muestra que, para el problema de ejemplo, la dirección obtenida con la expresión $\nabla_x g(x, y_g^*(x))$ es ortogonal a la verdadera dirección de descenso de $v(x)$. Esto se debe a que la expresión $\nabla_x g(x, y_g^*(x))$ ignora (los multiplicadores de Lagrange asociados a) las restricciones que están activas en la solución óptima. La expresión exacta de $\nabla v(x)$ cuando existen CCs es uno de los principales resultados que se describen en la próxima sección.

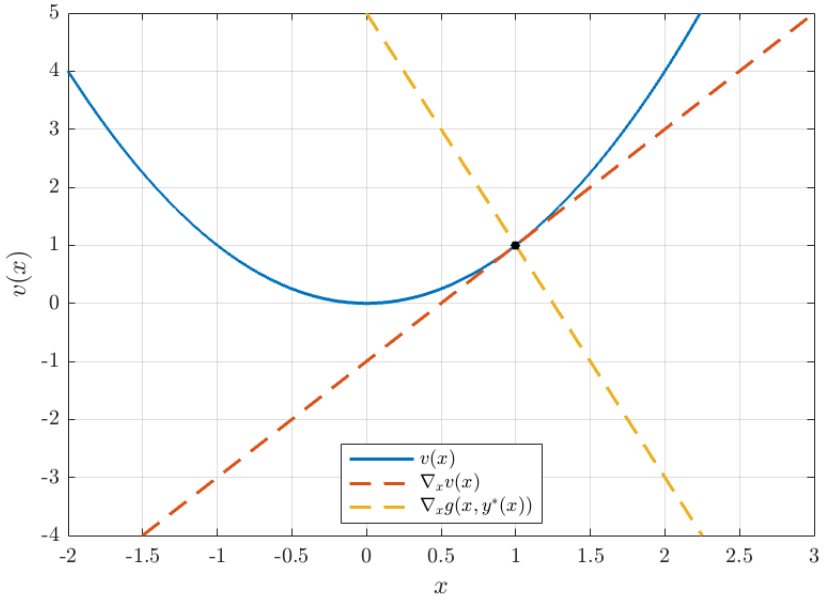


Figura 3.1: Cálculo de $\nabla v(x)$ en presencia de CCs. El ejemplo considerado es $v(x) = \text{mín}_y\, g(x,y)$ s.a: $g^c(x,y) \leq 0$ con $g(x,y) = (y-2x)^2$ y $g^c(x,y) = 3x - y$, de forma que $v(x) = x^2$. La línea azul representa la función de valor $v(x)$ en presencia de CCs, la línea roja discontinua representa el gradiente $\nabla v(x)$ calculado teniendo en cuenta los multiplicadores de Lagrange y, finalmente, la línea amarilla discontinua representa el gradiente $\nabla_x g(x, y_g^*(x))$ calculado según la expresión dada en [72, 84], que no tiene en cuenta las CCs. Se observa no solo que ambos gradientes son diferentes sino que, para el ejemplo planteado, la dirección obtenida por $\nabla_x g(x, y_g^*(x))$ es, de hecho, ortogonal a $\nabla v(x)$, el gradiente de la función de valor que tiene en cuenta las CCs.

2. Realizar una proyección conjunta de las variables de los niveles superior e inferior sobre las CCs del problema conlleva un elevado coste computacional. Si se extiende directamente la formulación propuesta en [72, 84] sería necesario proyectar conjuntamente las variables (x, y) sobre el conjunto de CCs $g^c(x,y) \leq 0$ para asegurar la factibilidad de la solución propuesta en cada paso del algoritmo. Salvo en unos pocos casos en los que existen expresiones cerradas, la proyección requiere resolver un problema de optimización. Además, la dificultad de la proyección se incrementa (muy) considerablemente cuando las CCs no son conjuntamente convexas en (x, y). Esto supone que el método de penalización propuesto en este TFG no sólo incurra en una menor complejidad computacional, sino que también permita trabajar con restricciones más generales (concretamente, se necesita convexidad en y, pero no convexidad conjuntamente en (x, y)).

Pese a las ventajas comentadas, la formulación propuesta presenta varios retos. Entre los más evidentes se encuentran ser capaces de relacionar la distancia $g(x,y) - v(x)$ con la distancia $\|y - y_g^*(x)\|$ y encontrar una forma de resolver los subproblemas de maximización-minimización (max-min) en (3.2a) en un tiempo razonable. Para hacer frente a estos desafíos, el primer paso consiste en identificar una serie de propiedades favorables que se pedirán a las funciones $f(x,y)$, $g(x,y)$ y $g^c(x,y)$. Dichas propiedades incluirán convexidad o suavidad de Lipschitz. Esta última es crucial para diseñar algoritmos basados en descenso por gradiente porque asegura que la función no evoluciona de forma abrupta, permitiendo una optimización más estable y eficiente.

3.2.1. Validez de la reformulación

En esta sección se proporciona un resultado formal que caracteriza de forma rigurosa la equivalencia entre la solución del problema original en (3.1) y la solución del problema aproximado en (3.2). Antes de caracterizar matemáticamente la relación, es necesario definir las suposiciones bajo las cuales la caracterización proporcionada es válida.

Suposición 1 (Continuidad Lipschitz) *Las funciones f, ∇f, ∇g, g^c y ∇g^c son respectivamente $l_{f,0}$, $l_{f,1}$, $l_{g,1}$, $l_{g^c,0}$, y $l_{g^c,1}$-Lipschitz conjuntamente sobre (x, y). Además, la función g es $l_{g_x,0}$-Lipschitz en x.*

Suposición 2 (Convexidad en y) *Para cualquier $x \in \mathcal{X}$ dado, $g(x, y)$ y $g^c(x, y)$ son α_g-fuertemente convexos y convexos en y, respectivamente.*

Suposición 3 (Factibilidad del Dominio) *Los dominios $\mathcal{X} \subset \mathbb{R}^{d_x}$ y $\mathcal{Y} \subset \mathbb{R}^{d_y}$ son no vacíos, cerrados y convexos. Para cualquier $x \in \mathcal{X}$, $\mathcal{Y}(x) := \{y \in \mathcal{Y} : g^c(x, y) \leq 0\}$ es no vacío.*

Suposición 4 (Calificación de Restricciones) *Para cualquier $x \in \mathcal{X}$ dado, las restricciones g^c satisfacen la condición de independencia LICQ en y.*

Las suposiciones sobre la continuidad Lipschitz (suposición 1) y la fuerte convexidad de g en y (suposición 2) son convencionales en la literatura [43, 69, 73, 79, 84, 91]. Además, solo se requiere que $g^c(x, y)$ sea convexo en y, en lugar de ser conjuntamente convexo en (x, y), como en [92], o lineal, como en [70, 91]. La suposición 3 se refiere a la convexidad y cierre del dominio, lo cual también es típico en el estado del arte. Finalmente, la suposición 4 es una condición común de calificación de restricciones (véase la sección 2.1.2).

El primer elemento clave para proponer una reformulación mediante una función de penalización para BLO con CCs en el nivel inferior es encontrar una función de penalización adecuada y que permita resolver el problema original. Concretamente, la función ha de regular la distancia entre la solución propuesta por la reformulación para las variables del nivel inferior y y la solución óptima original $y_g^*(x)$. Es decir, se busca controlar el valor de $\|y - y_g^*(x)\|$. En el lema 1 se demuestra que $g(x, y) - v(x)$ es una opción adecuada.

Enunciado del lema 1

Lema 1 *Considérese que las suposiciones 2-4 se satisfacen y que la función $v(x)$ se define como en* (3.2b)*, entonces se tiene que:*

1. $g(x, y) - v(x) \geq \frac{\alpha_g}{2}\|y - y_g^*(x)\|^2$, *para todo* $y \in \mathcal{Y}(x)$.
2. $g(x, y) = v(x)$ *si y solo si* $y = y_g^*(x)$, *para todo* $y \in \mathcal{Y}(x)$.

Prueba del lema 1

El desafío técnico de probar este lema radica en demostrar la propiedad de crecimiento cuadrático (primer resultado del lema). Para BLO sin CCs, esto se cumple naturalmente ya que el objetivo de nivel inferior $g(x, y)$ es fuertemente convexo en y. Para BLO con CCs, se demuestra por:

Lema 2 *Sea $\mathcal{Q} \in \mathbb{R}^{d_q}$ un conjunto cerrado, convexo y no vacío. $h : \mathbb{R}^{d_q} \to \mathbb{R}$ es una función fuertemente convexa en $\mathcal{Q}$ y $h^c : \mathbb{R}^{d_q} \to \mathbb{R}^{d_c}$ una función convexa en $\mathcal{Q}$ con d_c finito, de tal forma que el conjunto $\{q \in \mathcal{Q} : h^c(q) \leq 0\}$ es no vacío. Entonces se cumple que el problema de optimización* $\min_{q\in\mathcal{Q},h^c(q)\leq 0} h(q)$*:*

1. *Tiene una única solución óptima q^*.*
2. *Cuando se cumple la LICQ, los multiplicadores de Lagrange óptimos al resolver el problema de optimización*
   $\max_{\mu\in\mathbb{R}^{d_c}_{\geq 0}} \min_{q\in\mathcal{Q},h^c(q)\leq 0} h(q) + \langle \mu, h^c(q)\rangle$ *son únicos (véase su prueba en [98]).*

De acuerdo al lema 2, para cada valor de x, los multiplicadores de Lagrange óptimos asociados a las restricciones del nivel inferior $\mu^*(x)$ son únicos. Por tanto, se tiene

$$\min_{y\in\mathcal{Y}:g^c(x,y)\leq 0} g(x, y) \quad \Leftrightarrow \quad \min_{y\in\mathcal{Y}} g(x, y) + \langle \mu^*(x), g^c(x, y)\rangle. \tag{3.3}$$

Puesto que $g(x, y)$ es α_g-fuertemente convexo en y y $g^c(x, y)$ es convexo en y para cualquier $x \in \mathcal{X}$, se confirma que $g(x, y) + \langle \mu^*(x), g^c(x, y)\rangle$ es al menos α_g-fuertemente convexo en y. Por el teorema 2 en [99], el resultado 1 en el lema 1 es verdadero. Teniendo que el resultado 1 en el lema 1 es cierto y que los multiplicadores de Lagrange son únicos (tal y como se demuestra en [98]), se concluye que el resultado 2 en el lema 1 también es cierto.

□

Utilizando la relación entre la penalización $g(x, y) - v(c)$ y la distancia $\|y - y^*_g(x)\|$ obtenida en el lema 1, puede establecerse una relación entre la solución de la reformulación en (3.2) y una solución aproximada al problema original (3.1). Dicha relación se formaliza en el teorema 1, que se enuncia a continuación.

Enunciado del teorema 1

Teorema 1 (Equivalencia) *Considérese que f es Lipschitz continua en y, ∇f es $l_{f,1}$-Lipschitz en (x, y) según la suposición 1, y que las suposiciones 2-4 se cumplen. Entonces, resolver el problema ϵ-aproximado de* (3.1)*:*

$$\min_{(x,y)\in\{\mathcal{X}\times\mathcal{Y}:g^c(x,y)\leq 0\}} f(x, y) \quad s.t. \quad \|y - y^*_g(x)\|^2 \leq \epsilon, \tag{3.4}$$

es equivalente a resolver la reformulación de penalización primal-dual en (3.2) *con $\gamma = \mathcal{O}(\epsilon^{-0.5})$ y $\gamma > \frac{l_{f,1}}{\alpha_g}$.*

Prueba del teorema 1

Se sabe por el lema 1 que $g(x,y) - v(x) \geq \frac{\alpha_g}{2}\|y - y_g^*(x)\|^2$ y $g(x,y) = v(x)$ si y solo si $y = y_g^*(x)$. Esta cota cuadrática de distancia a la solución óptima y^* sigue la definición 1 en [72]. Bajo la condición de suavidad de Lipschitz de $f(x,y)$ con respecto a y, encontrar las soluciones del problema ϵ-aproximado en (3.4) es equivalente a encontrar las soluciones de su reformulación de penalización:

$$\min_{(x,y)\in\{\mathcal{X}\times\mathcal{Y}:g^c(x,y)\leq 0\}} f(x,y) + \gamma(g(x,y) - v(x)) \tag{3.5}$$

con $\gamma = \mathcal{O}(\epsilon^{-0.5})$ según los teoremas 1 y 2 en [72].
Además, encontrar conjuntamente la solución para (x,y) en (3.5) es equivalente a encontrar las soluciones en

$$\min_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}(x)} f(x,y) + \gamma(g(x,y) - v(x)). \tag{3.6}$$

La prueba de esta equivalencia es la siguiente. Supóngase que $(x_0,y_0) \in \{\mathcal{X}\times\mathcal{Y} : g^c(x,y) \leq 0\}$ es una solución de (3.5). Supóngase que para cualquier $x \in \mathcal{X}$, $y_F^*(x) \in \arg\min_{y\in\mathcal{Y}(x)} f(x,y) + \gamma(g(x,y) - v(x))$. Se sabe que para cualquier $x \in \mathcal{X}$, $y \in \mathcal{Y}(x)$, se cumple que

$$\begin{aligned} f(x_0,y_0) + \gamma(g(x_0,y_0) - v(x_0)) \leq & f(x, y_F^*(x)) + \gamma(g(x, y_F^*(x)) - v(x)) \\ \leq & f(x,y) + \gamma(g(x,y) - v(x)). \end{aligned}$$

Esto significa que cualquier solución de (3.5) es una solución de (3.6). Por otro lado, supóngase que $x_0 \in \mathcal{X}$, $y_F^*(x_0) \in \mathcal{Y}(x_0)$ es una solución de (3.6). Se sabe que para cualquier $(x,y) \in \{\mathcal{X}\times\mathcal{Y} : g^c(x,y) \leq 0\}$,

$$\begin{aligned} f(x_0, y_F^*(x_0)) + \gamma(g(x_0, y_F^*(x_0)) - v(x_0)) \leq & f(x, y_F^*(x)) + \gamma(g(x, y_F^*(x)) - v(x)) \\ \leq & f(x,y) + \gamma(g(x,y) - v(x)). \end{aligned}$$

Esto significa que cualquier solución de (3.6) es una solución de (3.5).
Además, se tiene que $f(x,y)$ es $l_{f,1}$-suave y $g(x,y)$ es α_g-fuertemente convexo en y. Por las definiciones de convexidad fuerte y suavidad de Lipschitz, para $x \in \mathcal{X}$ fijo y cualquier $y_1, y_2 \in \mathcal{Y}$,

$$\begin{aligned} & f(x,y_1) + \gamma(g(x,y_1) - v(x)) - f(x,y_2) + \gamma(g(x,y_2) - v(x)) \\ =& f(x,y_1) - f(x,y_2) + \gamma(g(x,y_1) - g(x,y_2)) \\ \geq& \langle \nabla_y f(x,y_2), y_1 - y_2\rangle - \frac{l_{f,1}}{2}\|y_1 - y_2\|^2 + \gamma\langle \nabla_y g(x,y_2), y_1 - y_2\rangle + \gamma\frac{\alpha_g}{2}\|y_1 - y_2\|^2 \\ =& \langle \nabla_y f(x,y_2) + \gamma\nabla_y g(x,y_2), y_1 - y_2\rangle + \frac{\gamma\alpha_g - l_{f,1}}{2}\|y_1 - y_2\|^2. \end{aligned} \tag{3.7}$$

Esto prueba que $f(x,y) + \gamma(g(x,y) - v(x))$ es $(\gamma\alpha_g - l_{f,1})$-fuertemente convexo en y. Además, según la suposición 2, la restricción $g^c(x,y)$ es convexa en y. Por el lema 2 el problema $\min_{y\in\mathcal{Y}(x)} f(x,y) + \gamma(g(x,y) - v(x))$ es equivalente a su forma dual [100]

$$\max_{\mu\in\mathbb{R}^{d_c}_{\geq 0}} \min_{y\in\mathcal{Y}} f(x,y) + \gamma(g(x,y) - v(x)) + \langle \mu, g^c(x,y)\rangle. \tag{3.8}$$

Al establecer $\gamma = \mathcal{O}(\epsilon^{-0.5})$, resolver la reformulación de penalización formulada en (3.5) es equivalente a resolver el problema original ϵ-aproximado (3.4).

□

En resumen, el teorema 1 demuestra que el problema binivel (3.1) y su reformulación penalizada (3.2) son aproximadamente equivalentes, siempre y cuando se cumplan ciertas condiciones sobre las funciones que forman el problema original.

Hay dos aspectos importantes que permiten esta equivalencia. Primero, el problema del nivel inferior, con función objetivo $g(x, y)$, es α_g-fuertemente convexo con respecto a las variables del nivel inferior y. Segundo, el problema del nivel superior, con función objetivo $f(x, y)$, es $l_{f,1}$-Lipschitz suave con respecto a las variables del nivel superior x e inferior y. Estas propiedades aseguran que la reformulación del problema binivel (3.2) sea convexa respecto a y si se selecciona un valor del parámetro γ tal que se cumpla $\gamma\alpha_g \geq l_{f,1}$, lo que significa que esta reformulación se puede resolver de manera sencilla para valores dados de las variables del nivel superior x.

Para entender mejor este teorema, consideremos una función $g(x, y)$ que es α_g-fuertemente convexa en y para cualquier valor de x. Esto significa que la curvatura[2] de la función siempre es positiva en el dominio $\mathcal{Y}(x)$ y, además, el valor mínimo de la curvatura está determinado por α_g. Asimismo, una función $f(x, y)$ que es $l_{f,1}$-Lipschitz suave en (x, y) cumple que la curvatura máxima en todo el dominio $\mathcal{X} \times \mathcal{Y}(x)$, en valor absoluto, está acotada por $l_{f,1}$. Por último, la función de valor $v(x)$, que se define como el mínimo de $g(x, y)$ sobre $y \in \mathcal{Y}(x)$, constante en (no depende de) y. En consecuencia, la expresión $f(x, y) + \gamma(g(x, y) - v(x))$ será convexa sobre y para cualquier valor fijo de x si $\gamma\alpha_g \geq l_{f,1}$.

El teorema 1 también establece que la solución al mínimo sobre $y \in \mathcal{Y}(x)$ de la función $f(x, y) + \gamma(g(x, y) - v(x))$ será aproximadamente el mismo que la solución al mínimo sobre $y \in \mathcal{Y}(x)$ de $g(x, y)$ para todo valor de $x \in \mathcal{X}$, y que la distancia entre ambas soluciones será menor mientras mayor sea el valor del parámetro γ.

Por tanto, el método para resolver la reformulación (3.2) consiste en dividir la minimización conjunta en (x, y) en dos problemas separados: i) un problema de minimización sobre x, llamado problema exterior y ii) un problema de minimización sobre y restringido a $\mathcal{Y}(x)$ (es decir, restringido a $g^c(x, y) \leq 0$), llamado problema interior. Para el problema interior en y, se elige γ de tal manera que se cumpla $\gamma\alpha_g - l_{f,1} > 0$. Esto asegura que el objetivo en (3.2a) sea fuertemente convexo en y. Además, la convexidad de $g^c(x, y)$ en y hace que se cumpla el teorema de dualidad fuerte [31], lo que permite adoptar una reformulación primal-dual max-min para (3.2).

3.2.2. Caracterización de las propiedades de la reformulación

En la sección anterior se caracterizó la equivalencia entre el problema BLO original (3.1) y la reformulación (3.2). Además, se caracterizaron varias propiedades de (3.2) con respecto a las variables del nivel inferior y cuando se congelan variables del nivel superior x. Estas propiedades incluyen la convexidad con respecto a y, lo que permite obtener de forma sencilla el valor de $y_F^*(x)$, la solución óptima de la reformulación, para cualquier valor dado de x. El objetivo de esta sección es caracterizar las propiedades de la reformulación conforme varía el valor de x. Para ello, será crítico caracterizar $\nabla v(x)$ y demostrar que tiene ciertas propiedades favorables. De forma más específica, se demuestra la suavidad de Lipschitz para la reformulación con respecto a x. Esto permitirá emplear un algoritmo de GD para converger a un mínimo local de la función $F_\gamma(x)$. De acuerdo a la definición de $F_\gamma(x)$ en (3.2a), evaluar $F_\gamma(x)$ en un x determinado exige encontrar la solución del problema max-min interno:

$$(\mu_F^*(x), y_F^*(x)) := \arg \max_{\mu \in \mathbb{R}^{d_x}_{\geq 0}} \min_{y \in \mathcal{Y}} \underbrace{f(x, y) + \gamma(g(x, y) - v(x)) + \langle \mu, g^c(x, y) \rangle}_{=: L_F(\mu, y; x)}. \tag{3.9}$$

La unicidad de $y_F^*(x)$ y $\mu_F^*(x)$ está garantizada bajo las suposiciones 2 y 4 por el lema 2. Por lo tanto, $F_\gamma(x)$ en (3.2a) se evalúa utilizando las soluciones óptimas únicas vía

$$F_\gamma(x) = L_F(\mu_F^*(x), y_F^*(x); x). \tag{3.10}$$

[2]la curvatura de una función en un punto mide cómo cambia la dirección de la tangente a la función en ese punto, reflejando la tasa de cambio de su pendiente. Matemáticamente, se determina usando la segunda derivada de la función: si esta es siempre positiva, la función es convexa, mientras que si es siempre negativa, la función es cóncava.

De manera similar, la función de valor puede evaluarse como $v(x) = L_g(\mu_g^*(x), y_g^*(x); x)$, donde

$$(\mu_g^*(x), y_g^*(x)) := \arg \max_{\mu \in \mathbb{R}^{d_x}_{\geq 0}} \min_{y \in \mathcal{Y}} \underbrace{g(x,y) + \langle \mu, g^c(x,y) \rangle}_{=: L_g(\mu, y; x)}. \tag{3.11}$$

La presencia del término $-\gamma v(x)$ en la función $F_\gamma(x)$ hace que la minimización con respecto a x sea más difícil puesto que: i) $-v(x)$ puede ser cóncava (véase lema 4.24 en [100]), por lo que es poco probable que la función $F_\gamma(x)$ sea convexa y ii) es difícil encontrar una expresión cerrada para $v(x)$. No obstante, puesto que la mayoría de los problemas de BLO son no convexos (y NP-difíciles), si el propósito es encontrar un mínimo local vía GD, el interés no se centra tanto en $v(x)$ sino en $\nabla v(x)$. En este contexto, se proporciona a continuación el lema 4 que no solo establece que $v(x)$ es diferenciable, sino que también proporciona una expresión analítica para su gradiente $\nabla v(x)$. De forma equivalente al lema 4, el gradiente $\nabla F_\gamma(x)$ se puede obtener mediante el lema 5. Antes de introducirlos, se introduce un resultado más general, recogido en el lema 3, cuya validez se demuestra en el teorema 4.24 en [101].

Enunciado del lema 3

Lema 3 *Considérense los conjuntos $\mathcal{Y}$ e $\{y \in \mathcal{Y} : h^c(x,y) \leq 0\}$, ambos no vacíos, cerrados y convexos. Considérense también las funciones $h(x,y)$ conjuntamente suave en (x,y), fuertemente convexa en y y Lipschitz continua en x, así como $h^c(x,y)$ convexa en y y Lipschitz continua en x. Entonces se tiene que*

$$v_h(x) = \min_{y \in \mathcal{Y}} h(x,y) \quad s.a: \quad h^c(x,y) \leq 0$$

es diferenciable y que su gradiente es

$$\nabla v_h(x) = \nabla_x h(x, y_h^*(x)) + \langle \mu_h^*(x), h^c(x, y_h^*(x)) \rangle, \tag{3.12}$$

donde $(y_h^*(x), \mu_h^*(x))$ es único.

Utilizando este lema, se puede obtener el gradiente de la función de valor en (3.2b) como sigue.

Enunciado del lema 4

Lema 4 (Lema tipo Danskin para $v(x)$) *Considérese que se cumplen las suposiciones 1-4, y sea $B_g < \infty$ una constante tal que $\|\mu_g^*(x)\| < B_g$ para todo $x \in \mathcal{X}$. Entonces, se cumple que:*

1. *Las funciones $y_g^*(x)$ y $\mu_g^*(x)$ definidas en (3.11) son L_g-Lipschitz para una constante finita $L_g \geq 0$.*
2. *La función de valor $v(x)$ definida en (3.2b) es $l_{v,1}$-suave donde $l_{v,1} \leq (l_{g,1} + B_g l_{g^c,1})(1 + L_g) + l_{g^c,0} L_g$, y*

$$\nabla v(x) = \nabla_x g(x, y_g^*(x)) + \langle \mu_g^*(x), \nabla_x g^c(x, y_g^*(x)) \rangle. \tag{3.13}$$

Prueba del lema 4

El problema $\min_{y\in\mathcal{Y}} g(x,y)$ s.t. $g^c(x,y)\leq 0$ se ajusta al contexto del lema 3 tomando $h(x,y)=g(x,y)$ y $h^c(x,y)=g^c(x,y)$. Por lo tanto, es posible computar el valor de la derivada (3.13). Además, para cualesquiera $x_1,x_2\in\mathcal{X}$,

$$\begin{aligned}
&\|\nabla v(x_1)-\nabla v(x_2)\|\\
=&\|\nabla_x g(x_1,y_g^*(x_1))+\langle\mu_g^*(x_1),\nabla_x g^c(x_1,y_g^*(x_1))\rangle-\nabla_x g(x_2,y_g^*(x_2))\\
&-\langle\mu_g^*(x_2),\nabla_x g^c(x_2,y_g^*(x_2))\rangle\|\\
\overset{(a)}{\leq}&\|\nabla_x g(x_1,y_g^*(x_1))-\nabla_x g(x_2,y_g^*(x_2))\|\\
&+\|\langle\mu_g^*(x_1),\nabla_x g^c(x_1,y_g^*(x_1))\rangle-\langle\mu_g^*(x_1),\nabla_x g^c(x_2,y_g^*(x_2))\rangle\|\\
&+\|\langle\mu_g^*(x_1),\nabla_x g^c(x_2,y_g^*(x_2))\rangle-\langle\mu_g^*(x_2),\nabla_x g^c(x_2,y_g^*(x_2))\rangle\|\\
\overset{(b)}{\leq}&(l_{g,1}+B_g l_{g^c,1})(\|x_1-x_2\|+\|y_g^*(x_1)-y_g^*(x_2)\|)+l_{g^c,0}\|\mu_g^*(x_1)-\mu_g^*(x_2)\|\\
\overset{(c)}{\leq}&((l_{g,1}+B_g l_{g^c,1})(1+L_g)+l_{g^c,0}L_g)\|x_1-x_2\|,
\end{aligned}$$

donde (a) sigue la desigualdad triangular; (b) se apoya en la continuidad Lipschitz de ∇g, g^c y ∇g^c en x, y el límite superior para $\|\mu_g^*(x)\|$; y (c) utiliza la continuidad Lipschitz de $y_g^*(x)$ y $\mu_g^*(x)$ del teorema 2.16 en [102]. Se concluye que $v(x)$ es $l_{v,1}$-suave donde $l_{v,1}\leq(l_{g,1}+B_g l_{g^c,1})(1+L_g)+l_{g^c,0}L_g$.

□

La suposición de la existencia del límite superior para el multiplicador de Lagrange en el lema 4 es habitual (véase teorema 1.6 en [102] y [92]).

Por último, utilizando la expresión de $\nabla v(x)$, se puede obtener el gradiente de la función $F_\gamma(x)$ (3.2b) como sigue.

Enunciado del lema 5

Lema 5 (Teorema tipo Danskin para $F_\gamma(x)$) *Considérese que se cumplen las condiciones en el lema 4. Supóngase, además, que $\gamma>\frac{l_{f,1}}{\alpha_g}$ y que existe $B_F<\infty$ tal que $\|\mu_F^*(x)\|<B_F$ para todo $x\in\mathcal{X}$. Entonces, se tiene que:*

1. *Las funciones $y_F^*(x)$ y $\mu_F^*(x)$ definidas en (3.9) son L_F-Lipschitz para alguna constante $L_F\geq 0$.*
2. *La función $F_\gamma(x)$ definida en (3.2a) es $l_{F,1}$-suave con $l_{F,1}\leq(l_{f,1}+\gamma l_{g,1}+B_F l_{g^c,1})(1+L_F)+\gamma l_{v,1}+l_{f^c,0}L_F$, y*

$$\nabla F_\gamma(x)=\nabla_x f(x,y_F^*(x))+\gamma\left(\nabla_x g(x,y_F^*(x))-\nabla v(x)\right)+\langle\mu_F^*(x),\nabla_x g^c(x,y_F^*(x))\rangle. \quad (3.14)$$

Prueba del lema 5

Por las suposiciones 1 y 2, $f(x,y)$ es $l_{f,1}$-suave y $g(x,y)$ es α_g-fuertemente convexa en y. Se tiene que $f(x,y)+\gamma(g(x,y)-v(x))$ es $(\gamma\alpha_g - l_{f,1})$-fuertemente convexa cuando $\gamma > \frac{l_{f,1}}{\alpha_g}$ como se concluyó en (3.7). Además, la restricción $g^c(x,y)$ es convexa en y según la suposición 2. De esta manera, el problema

$$\min_{y\in\mathcal{Y}} \; f(x,y)+\gamma(g(x,y)-v(x)) \qquad \text{s.t. } g^c(x,y)\leq 0$$

es equivalente a

$$F_\gamma(x) = \max_{\mu\in\mathbb{R}^{d_c}_{\geq 0}} \min_{y\in\mathcal{Y}} \; f(x,y)+\gamma(g(x,y)-v(x))+\langle \mu, g^c(x,y)\rangle.$$

Considerando la suavidad de Lipschitz de $v(x)$ presentada en el lema 4, todas las suposiciones en el lema 3 están satisfechas. Por lo tanto, es posible computar la expresión dada en (3.14). Además, para cualesquiera $x_1, x_2 \in \mathcal{X}$,

$$\begin{aligned}
&\|\nabla F(x_1) - \nabla F(x_2)\| \\
=&\|\nabla_x f(x_1, y_F^*(x_1)) + \gamma\left(\nabla_x g(x_1, y_F^*(x_1)) - \nabla v(x_1)\right) + \langle \mu_F^*(x_1), \nabla_x g^c(x_1, y_F^*(x_1))\rangle \\
&- \nabla_x f(x_2, y_F^*(x_2)) - \gamma\left(\nabla_x g(x_2, y_F^*(x_2)) - \nabla v(x_2)\right) - \langle \mu_F^*(x_2), \nabla_x g^c(x_2, y_F^*(x_2))\rangle\| \\
\overset{(a)}{\leq}&\|\nabla_x f(x_1, y_F^*(x_1)) - \nabla_x f(x_2, y_F^*(x_2))\| + \gamma\|\nabla_x g(x_1, y_F^*(x_1)) - \nabla_x g(x_2, y_F^*(x_2))\| \\
&+ \gamma\|\nabla v(x_1) - \nabla v(x_2)\| + \|\langle \mu_F^*(x_1), \nabla_x g^c(x_1, y_F^*(x_1))\rangle - \langle \mu_F^*(x_1), \nabla_x g^c(x_2, y_F^*(x_2))\rangle\| \\
&+ \|\langle \mu_F^*(x_1), \nabla_x g^c(x_2, y_F^*(x_2))\rangle - \langle \mu_F^*(x_2), \nabla_x g^c(x_2, y_F^*(x_2))\rangle\| \\
\overset{(b)}{\leq}&(l_{f,1} + \gamma l_{g,1} + B_F l_{g^c,1})(\|x_1 - x_2\| + \|y_F^*(x_1) - y_F^*(x_2)\|) + \gamma l_{v,1}\|x_1 - x_2\| \\
&+ l_{g^c,0}\|\mu_F^*(x_1) - \mu_F^*(x_2)\| \\
\overset{(c)}{\leq}&((l_{f,1} + \gamma l_{g,1} + B_F l_{g^c,1})(1 + L_F) + \gamma l_{v,1} + l_{f^c,0} L_F)\|x_1 - x_2\|,
\end{aligned}$$

donde (a) sigue la desigualdad triangular; (b) se apoya en la continuidad Lipschitz de ∇f, ∇g, g^c y ∇g^c en x, y el límite superior para $\|\mu_F^*(x)\|$; y (c) usa la continuidad Lipschitz de $y_F^*(x)$ y $\mu_F^*(x)$ del teorema 2.16 en [102]. Se puede concluir que $F(x)$ es $l_{F,1}$-suave donde $l_{F,1} \leq (l_{f,1} + \gamma l_{g,1} + B_F l_{g^c,1})(1 + L_F) + \gamma l_{v,1} + l_{f^c,0} L_F$.

□

En resumen, de forma similar a [72], los teoremas tipo Danskin en los lemas 4 y 5 se basan en la continuidad Lipschitz de las soluciones en (3.9) y (3.11) respecto a las variables del nivel superior x. Esto, junto con la convexidad respecto a las variables del nivel inferior y, permite resolver la reformulación (3.2) para obtener una solución ϵ-aproximada del problema BLO con CCs (3.1). Además, como se ha demostrado, a diferencia de BLO sin CCs [72], el gradiente de la reformulación $F_\gamma(x)$ depende de los multiplicadores de Lagrange $\mu_g^*(x)$ y $\mu_F^*(x)$.

A partir de los resultados de la caracterización teórica de la reformulación (3.2), la siguiente sección diseña un método de gradiente primal-dual para resolver problemas de BLO con CCs con rigurosas garantías de convergencia. Este método aprovecha las propiedades de suavidad de Lipschitz para asegurar que cada iteración del algoritmo haga un progreso significativo hacia la solución óptima.

3.3. El algoritmo BLOCC

En esta sección se describe la contribución principal de este TFG, el algoritmo BLOCC, diseñado para resolver para problemas de optimización binivel con CCs de desigualdad (sección 3.3.1). Asimismo, se caracterizan de forma analítica varias de sus propiedades, obteniendo garantías teóricas de convergencia y optimalidad (sección 3.3.2). Finalmente, se propone un algoritmo primal-dual para resolver los problemas max-min internos y, en el caso concreto de que g^c sea afín en $y \in \mathcal{Y}$, se caracteriza su complejidad computacional (sección 3.3.3).

3.3.1. Descripción del algoritmo

En esta subsección, se presenta el algoritmo BLOCC, basado en PGD, para resolver el problema de minimización penalizado $F_\gamma(x)$. El algoritmo aprovecha las propiedades de diferenciabilidad y suavidad de Lipschitz de $F_\gamma(x)$, permitiendo una actualización eficiente de las variables del nivel superior x en cada iteración. La subsección se organiza de la siguiente manera:

- En primer lugar, se describe detalladamente el proceso iterativo del algoritmo, especificando los cálculos necesarios para evaluar las expresiones de los gradientes involucrados.
- A continuación, se explican los algoritmos de optimización max-min empleados para encontrar las soluciones aproximadas para las variables del nivel inferior, necesarias para estas evaluaciones de gradiente.
- Finalmente, se presenta el pseudocódigo del algoritmo, que resume sistemáticamente todos los pasos descritos.

El objetivo del algoritmo es resolver $\min_{x\in\mathcal{X}} F_\gamma(x)$. De acuerdo a las suposiciones realizas a lo largo del capítulo, el conjunto $\mathcal{X}$ es no vacío, cerrado y convexo y, además, $F_\gamma(x)$ es tanto diferenciable como Lipschitz suave (véase lema 5). Por ello, se propone abordar $\min_{x\in\mathcal{X}} F_\gamma(x)$ mediante un método iterativo basado en PGD. Concretamente, si t denota el índice de iteración, el valor de la variable x en la iteración t se actualiza de la siguiente manera:

$$x_{t+1} = \mathrm{Proj}_{\mathcal{X}}(x_t - \eta g_{F,t}), \tag{3.15}$$

donde η es el tamaño (constante) del paso y $g_{F,t}$ es una estimación de $\nabla F_\gamma(x_t)$.

En la sección 3.2, se obtuvieron las expresiones cerradas de $\nabla v(x)$ en (3.13) y $\nabla F_\gamma(x)$ en (3.14). Evaluar estas expresiones requiere encontrar $(y_g^*(x_t), \mu_g^*(x_t))$ y $(y_F^*(x_t), \mu_F^*(x_t))$, que son las soluciones del problema max-min $L_g(\mu, y; x_t)$ en (3.11) y $L_F(\mu, y; x_t)$ en (3.9), respectivamente. Puesto, que de forma general, los lagrangianos no tienen solución cerrada, las soluciones se obtienen mediante algoritmos iterativos y tendrán siempre un pequeño error, que será menor cuanto mayor sea el número de iteraciones. Para formalizar esto, T_g denota las iteraciones ejecutadas para resolver (3.11) de forma ϵ_g-aproximada y T_F denota las iteraciones para resolver (3.9) de forma ϵ_F-aproximada. Si denotamos como $(y_{g,t}^{T_g}, \mu_{g,t}^{T_g})$ y $(y_{F,t}^{T_F}, \mu_{F,t}^{T_F})$ las soluciones obtenidas, tenemos, por lo tanto que dichas soluciones satisfacen:

$$\left\|(y_{g,t}^{T_g}, \mu_{g,t}^{T_g}) - (y_g^*(x_t), \mu_g^*(x_t))\right\|^2 = \mathcal{O}(\epsilon_g), \quad \left\|(y_{F,t}^{T_F}, \mu_{F,t}^{T_F}) - (y_F^*(x_t), \mu_F^*(x_t))\right\|^2 = \mathcal{O}(\epsilon_F) \tag{3.16}$$

para precisiones de estimación objetivo $\epsilon_g, \epsilon_F > 0$. Aunque en la sección 3.3.3 se describe con más detalle el algoritmo de optimización max-min empleado, para el cómputo del gradiente aproximado g_F, t, basta con suponer que los valores de $y_{g,t}^{T_g}, \mu_{g,t}^{T_g}, y_{F,t}^{T_F}$ y $\mu_{F,t}^{T_F}$ están disponibles. A partir de estos valores, el gradiente de la función de valor $\nabla v(x_t)$ se aproxima vía (3.13) como

$$\nabla v(x_t) \approx g_{v,t} = \nabla_x g(x_t, y_{g,t}^{T_g}) + \langle \mu_{g,t}^{T_g}, \nabla_x g^c(x_t, y_{g,t}^{T_g}) \rangle. \tag{3.17}$$

Finalmente, utilizando el valor de $g_{v,t}$, el gradiente $\nabla F_\gamma(x_t)$ se aproxima vía (3.14) como

$$\nabla_x F(x_t) \approx g_{F,t} = \nabla_x f(x_t, y_{F,t}^{T_F}) + \gamma \left(\nabla_x g(x_t, y_{F,t}^{T_F}) - g_{v,t} \right) + \langle \mu_F^{T_F}, \nabla_x g^c(x_t, y_{F,t}^{T_F}) \rangle. \tag{3.18}$$

Sustituyendo la expresión (3.18) en (3.15), permite resolver el problema relajado mediante PGD.

Nótese que, cuando γ toma un valor grande, el valor de $y_{F,t}^{T_F}$ será muy similar al valor óptimo ϵ_F-aproximado del nivel inferior $y_{g,t}^{T_g}$, mientras que el valor de $\mu_{F,t}^{T_F}$ será muy similar a $\gamma \mu_{g,t}^{T_g} - \nabla_y f(x_t, y_{F,t}^{T_F})$. De esta forma, $g_{F,t} \approx \nabla_x f(x_t, y_{F,t}^{T_F}) - \langle \nabla_y f(x_t, y_{F,t}^{T_F}), \nabla_x g^c(x_t, y_{F,t}^{T_F}) \rangle$. Así, las variables del nivel superior x se actualizarán no solo teniendo en cuenta $\nabla_x f(x_t, y_{F,t}^{T_F})$, sino también el impacto que tendrían en la función objetivo del nivel superior posibles reacciones por parte de las variables del nivel inferior. Dicho impacto vendrá determinado por el producto entre el gradiente de las CCs $\nabla_x g^c(x_t, y_{F,t}^{T_F})$ y el gradiente de la función objetivo del nivel superior respecto a las variables del nivel inferior.

El método de PGD para encontrar la solución de mín $F_\gamma(x)$ se sintetiza en el algoritmo 1, donde la implementación de la función MaxMin se proporcionará en la sección 3.3.3.

Algoritmo 1: BLOCC

Entradas: conjuntos $\mathcal{X}$,$\mathcal{Y}$; tamaño del paso η, η_g, η_F; contadores T_g, T_F; parámetros de precisión ϵ_g,ϵ_F, parámetro γ

Inicialización: puntos iniciales x_0, $y_{g,0}$, $\mu_{g,0}$, $y_{F,0}$, $\mu_{F,0}$

para $t = 0, 1, \ldots, T-1$ **hacer**

  Calcular la solución ϵ_g-aproximada de (3.11) como $(y_{g,t}^{T_g}, \mu_{g,t}^{T_g}) = \mathsf{MaxMin}(T_g, T_y^g, \eta_g)$

  Calcular la solución ϵ_F-aproximada de (3.9) como $(y_{F,t}^{T_F}, \mu_{F,t}^{T_F}) = \mathsf{MaxMin}(T_F, T_y^F, \eta_F, \gamma)$

  Actualizar x_{t+1} mediante (3.15) con $g_{F,t}$ como $x_{t+1} = \mathrm{Proj}_{\mathcal{X}}(x_t - \eta g_{F,t})$

fin

Salidas: $(x_T, y_{g,T}^{T_g})$

3.3.2. Demostración de convergencia

Una vez descrito el algoritmo de PGD para minimizar $F_\gamma(x)$, el siguiente paso es caracterizar de forma analítica su convergencia a (un punto cercano a) un óptimo local. En esta sección, se presenta el resultado de convergencia del algoritmo mediante el teorema 2, demostrando que la calidad de la solución cuando el número de iteraciones T crece depende de los términos de precisión ϵ_g y ϵ_F, así como del parámetro γ. La métrica utilizada es la media aritmética del cuadrado del gradiente proyectado $G_\eta = \eta^{-1}(x_{t+1} - x_t)$. Se concluye con la demostración detallada de este resultado, destacando los pasos clave y las implicaciones para la implementación del algoritmo.

Enunciado del teorema 2

Teorema 2 *Considérese que se cumplen las suposiciones del lema 5 y que el algoritmo 1 se ejecuta con T_F y T_g suficientemente grandes de manera que $(y_{g,t}^{T_g}, \mu_{g,t}^{T_g})$ y $(y_{F,t}^{T_F}, \mu_{F,t}^{T_F})$ sean $\mathcal{O}(\epsilon_g)$ y $\mathcal{O}(\epsilon_F)$-óptimos en distancia al cuadrado, tal y como se establece en* (3.16)*. Por último, considérese que el valor del tamaño del paso satisface $\eta \leq \frac{1}{l_{F,1}}$, donde $l_{F,1}$ se define en el lema 5. Entonces, se tiene que:*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|G_\eta(x_t)\|^2 = \frac{1}{T\eta^2} \sum_{t=0}^{T-1} \|(x_{t+1} - x_t)\|^2 = \mathcal{O}(\gamma T^{-1} + \gamma^2 \epsilon_F + \gamma^2 \epsilon_g). \tag{3.19}$$

Prueba del teorema 2

Se define el término de sesgo $b(x_t)$ del gradiente $\nabla F_\gamma(x_t)$ como:

$$\begin{aligned}
b(x_t) :=& \nabla F_\gamma(x_t) - g_{F,t} \\
=& \Big(\nabla_x f(x_t, y_F^*(x_t)) + \gamma \big(\nabla_x g(x, y_F^*(x_t)) - \big(\nabla_x g(x_t, y_g^*(x_t)) + \langle \mu_g^*(x_t), \nabla_x g^c(x_t, y_g^*(x_t)) \rangle \big) \big) \\
& + \langle \mu_F^*(x_t), \nabla_x g^c(x_t, y_F^*(x_t)) \rangle \Big) \\
& - \Big(\nabla_x f(x_t, y_{F,t}^{T_F}) + \gamma \Big(\nabla_x g(x_t, y_{F,t}^{T_F}) - \Big(\nabla_x g(x_t, y_{g,t}^{T_g}) + \Big\langle \mu_{g,t}^{T_g}, \nabla_x g^c(x_t, y_{g,t}^{T_g}) \Big\rangle \Big) \Big) \\
& + \langle \mu_F^{T_F}, \nabla_x g^c(x_t, y_{F,t}^{T_F}) \rangle \Big).
\end{aligned}$$

De esta manera, se tiene

$$\begin{aligned}
\|b(x_t)\| \overset{(a)}{\leq} & \|\nabla_x f(x_t, y_{F,t}^{T_F}) - \nabla_x f(x_t, y_F^*(x_t))\| \\
& + \gamma \Big(\|\nabla_x g(x_t, y_{F,t}^{T_F}) - \nabla_x g(x_t, y_F^*(x_t))\| + \|\nabla_x g(x_t, y_{g,t}^{T_g}) - \nabla_x g(x_t, y_g^*(x_t))\| \\
& \qquad + \Big\| \langle \mu_g^*(x_t), \nabla_x g^c(x_t, y_g^*(x_t)) \rangle - \Big\langle \mu_g^*(x_t), \nabla_x g^c(x_t, y_{g,t}^{T_g}) \Big\rangle \Big\| \\
& \qquad + \Big\| \Big\langle \mu_g^*(x_t), \nabla_x g^c(x_t, y_{g,t}^{T_g}) \Big\rangle - \Big\langle \mu_{g,t}^{T_g}, \nabla_x g^c(x_t, y_{g,t}^{T_g}) \Big\rangle \Big\| \Big) \\
& + \|\langle \mu_F^{T_F}, \nabla_x g^c(x_t, y_{F,t}^{T_F}) \rangle - \langle \mu_F^*(x_t), \nabla_x g^c(x_t, y_{F,t}^{T_F}) \rangle\| \\
& + \|\langle \mu_F^*(x_t), \nabla_x g^c(x_t, y_{F,t}^{T_F}) \rangle - \langle \mu_F^*(x_t), \nabla_x g^c(x_t, y_F^*(x_t)) \rangle\| \\
\overset{(b)}{\leq} & l_{f,1} \|y_{F,t}^{T_F} - y_F^*(x_t)\| + \gamma \Big(l_{g,1} \|y_{F,t}^{T_F} - y_F^*(x_t)\| + l_{g,1} \|y_{g,t}^{T_g} - y_g^*(x_t)\| \\
& \qquad + l_{g^c,0} \|\mu_{g,t}^{T_g} - \mu_g^*(x_t)\| + B_g l_{g^c,1} \|y_{g,t}^{T_g} - y_g^*(x_t)\| \Big) \\
& + l_{g^c,0} \|\mu_{F,t}^{T_F} - \mu_F^*(x_t)\| + B_F l_{g^c,1} \|y_{F,t}^{T_F} - y_F^*(x_t)\| \\
\overset{(c)}{=} & (l_{f,1} + \gamma l_{g,1} + B_F l_{g^c,0}) \|y_{F,t}^{T_F} - y_F^*(x_t)\| + l_{g^c,0} \|\mu_{F,t}^{T_F} - \mu_F^*(x_t)\| \\
& + \gamma \Big((l_{g,1} + B_g l_{g^c,1}) \|y_{g,t}^{T_g} - y_g^*(x_t)\| + l_{g^c,0} \|\mu_{g,t}^{T_g} - \mu_g^*(x_t)\| \Big),
\end{aligned}$$

donde (a) utiliza la desigualdad triangular, (b) se basa en la continuidad Lipschitz de ∇f, ∇g, g^c, y ∇g^c en x, los límites superiores para $\|\mu_F^*(x)\|$ y $\|\mu_g^*(x)\|$ y la desigualdad de Cauchy-Schwartz, y (c) es por reordenamiento.

□

Además, según la desigualdad de Young

$$\begin{aligned}\|b(x_t)\|^2 \leq & 2\left((l_{f,1}+\gamma l_{g,1}+B_F l_{g^c,0})\|y_{F,t}^{T_F}-y_{F,t}^*\|+l_{g^c,0}\|\mu_{F,t}^{T_F}-\mu_{F,t}^*\|\right)^2 \\ & +2\gamma^2\left((l_{g,1}+B_g l_{g^c,1})\|y_{g,t}^{T_g}-y_g^*(x_t)\|+l_{g^c,0}\|\mu_{g,t}^{T_g}-\mu_g^*(x_t)\|\right)^2 \\ = & \mathcal{O}(\gamma^2\epsilon_F+\gamma^2\epsilon_g).\end{aligned}$$

Según el lema 5, $F_\gamma(x)$ es $l_{F,1}$-suave en $\mathcal{X}$. De esta manera, por la suavidad de Lipschitz, se tiene que

$$\begin{aligned}F(x_{t+1}) \leq & F(x_t)+\langle\nabla F(x_t), x_{t+1}-x_t\rangle+\frac{l_{F,1}}{2}\|x_{t+1}-x_t\|^2 \\ \leq & F(x_t)+\langle g_{F,t}, x_{t+1}-x_t\rangle+\frac{1}{2\eta}\|x_{t+1}-x_t\|^2+\langle b(x_t), x_{t+1}-x_t\rangle,\end{aligned} \tag{3.20}$$

donde la segunda desigualdad es por $\eta \leq \frac{1}{l_{F,1}}$ y $\nabla F(x_t) = g_{F_t} + b(x_t)$.
La proyección garantiza que x_{t+1} y x_t estén en $\mathcal{X}$. Siguiendo el lema 3.1 en [103], se tiene

$$\langle g_{F,t}, x_{t+1}-x_t\rangle \leq -\frac{1}{\eta}\|x_{t+1}-x_t\|^2.$$

Insertando esto de nuevo en (3.20), se tiene

$$\begin{aligned}F(x_{t+1}) \leq & F(x_t)-\frac{1}{2\eta}\|x_{t+1}-x_t\|^2+\langle b(x_t), x_{t+1}-x_t\rangle \\ \leq & F(x_t)-\frac{1}{2\eta}\|x_{t+1}-x_t\|^2+\eta\|b(x_t)\|^2+\frac{1}{4\eta}\|x_{t+1}-x_t\|^2 \\ = & F(x_t)-\frac{1}{4\eta}\|x_{t+1}-x_t\|^2+\eta\|b(x_t)\|^2,\end{aligned}$$

donde la segunda desigualdad se deduce de la desigualdad de Young. Por lo tanto, se obtiene

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\|G_\eta(x_t)\|^2 \leq & \frac{4}{\eta T}(F(x_0)-F(x_T))+\frac{4}{T}\sum_{t=0}^{T-1}\|b(x_t)\|^2 \\ = & \mathcal{O}(\eta^{-1}T^{-1})+O(\gamma^2\epsilon_F+\gamma^2\epsilon_g) \\ = & \mathcal{O}(\gamma T^{-1}+\gamma^2\epsilon_F+\gamma^2\epsilon_g)\end{aligned}$$

donde la última igualdad se debe al hecho de que $\eta = \mathcal{O}(\gamma^{-1})$, ya que $\eta \leq \frac{1}{l_{F,1}}$ y $l_{F,1} \leq (l_{f,1}+\gamma l_{g,1}+B_F l_{g^c,1})(1+L_F)+\gamma l_{v,1}+l_{f^c,0}L_F = \mathcal{O}(\gamma)$.

□

En resumen, el teorema demuestra la convergencia del algoritmo 1 a un punto donde el gradiente del problema de BLO con CCs se anula (se hace arbitrariamente pequeño). De forma más específica, la expresión (3.19) del teorema 2 establece que la solución del algoritmo BLOCC para las variables del nivel superior x converge en función del número de iteraciones T, el parámetro γ y los parámetros de precisión ϵ_F y ϵ_g. Como se deduce de la expresión (3.19), tras un número suficientemente grande de iteraciones, la diferencia promedio entre x_t y x_{t+1} queda acotada únicamente por el parámetro γ y los parámetros de precisión ϵ_F y ϵ_g. Usar el gradiente proyectado $G_\eta(x_t) = \eta^{-1}(x_{t+1}-x_t)$ como la métrica de convergencia para problemas de optimización con restricciones es común en el estado del arte [104]. El término $\mathcal{O}(\gamma^2\epsilon_F+\gamma^2\epsilon_g)$ surge de los errores de estimación asociados al algoritmo MaxMin y, tal y como se explicó en la sección anterior, se puede reducir si se aumenta el valor de T_F y T_g.

Una vez se ha demostrado la convergencia del algoritmo BLOCC para resolver problemas de BLO con CCs, es el momento de analizar su complejidad computacional. Los pasos más costosos del algoritmo 1 son los ejecutados en las líneas 3 y 4, donde se resuelven los problemas max-min del nivel inferior. Por ello, la complejidad asociada al algoritmo 1 está básicamente determinada por la complejidad (eficiencia) de los algoritmos MaxMin utilizados para resolver los problemas internos. La siguiente sección lleva a cabo el análisis de complejidad de los problemas internos.

3.3.3. Resolución del nivel inferior y complejidad computacional

El último paso para implementar el algoritmo BLOCC consiste en especificar cómo resolver los dos problemas max-min en las líneas 3 y 4 del algoritmo 1. Ese es precisamente el objetivo de esta sección, donde: i) se propone un algoritmo para resolver los problemas (3.9) y (3.11), y ii) se caracteriza su velocidad de convergencia para el caso en el que las CCs $g^c(x, y)$ sean afines con respecto a y.

Comenzando por el objetivo en i), se presenta a continuación el algoritmo MaxMin primal-dual de descenso-ascenso por gradiente, diseñado para resolver los subproblemas max-min en (3.9). y (3.11) de manera eficiente. El algoritmo propuesto garantiza la convergencia y proporciona cotas superiores para las distancias $||\mu_{g,t}^{T_g} - \mu_g^*(x_t)||$ y $||\mu_{F,t}^{T_F} - \mu_F^*(x_t)||$. Al igual que se hizo en la sección 3.3.1, en primer lugar se explican y justifican las decisiones de diseño del algoritmo y, posteriormente, se ofrece el pseudocódigo que resume de manera precisa los pasos a ejecutar por el algoritmo.

Para un x fijo, los problemas (3.9) y (3.11) son cóncavos en μ y fuertemente convexos en y. En consecuencia, se puede desarrollar algoritmos basados en el siguiente problema abstracto max-min:

$$\max_{\mu \in \mathbb{R}^{d_c}_{\geq 0}} \min_{y \in \mathcal{Y}} L(\mu, y), \tag{3.21}$$

donde $L(\mu, y)$ es fuertemente convexo en y y cóncavo en μ. La literatura existente proporciona algoritmos adaptados para este tipo de problemas con complejidades aproximadas $\mathcal{O}(\epsilon^{-0.5})$. Algoritmos notables se incluyen en [71, 72, 104]. Sin embargo, cuantificar la precisión de la estimación de $\nabla F_\gamma(x)$ y $\nabla v(x)$ requiere estimar $||\mu_{g,t}^{T_g} - \mu_g^*(x_t)||$ y $||\mu_{F,t}^{T_F} - \mu_F^*(x_t)||$, lo cual no es sencillo para los algoritmos mencionados. Por lo tanto, en este TFG se propone el uso del algoritmo 2, un esquema dual-primal de ascenso-descenso por gradiente, que garantiza convergencia y cotas superiores para $||\mu_{g,t}^{T_g} - \mu_g^*(x_t)||$ y $||\mu_{F,t}^{T_F} - \mu_F^*(x_t)||$ tras ejecutar T iteraciones.

Concretamente, se propone utilizar un algoritmo con dos bucles anidados en el que, en cada iteración $t = 1, ..., T$ del bucle externo, se actualiza la variable dual μ_t. Dicha actualización requiere resolver una optimización con respecto a la variable primal y, lo que se realizará en el bucle interno. En este sentido, se recuerda que la función dual evaluada en μ_t es

$$D(\mu_t) = \min_{y \in \mathcal{Y}} L(\mu_t, y) = L(\mu_t, y_{\mu_t}^*), \text{ donde } y_{\mu_t}^* = \arg\min_{y \in \mathcal{Y}} L(\mu_t, y). \tag{3.22}$$

Puesto que $L(\mu_t, y)$ es fuertemente convexo en y, el valor de $y_{\mu_t}^*$ puede obtenerse mediante un algoritmo de descenso por gradiente que convergerá de forma lineal [33]. Específicamente, en cada iteración $t_y = 1, ..., T_y$ del bucle interno, la variable y se actualiza vía PDG

$$y_{t,t_y+1} = \text{Proj}_{\mathcal{Y}}(y_{t,t_y} - \eta_1 \nabla_y L(\mu_t, y_{t,t_y})). \tag{3.23}$$

Tras correr T_y iteraciones, se obtiene la salida y_{t,T_y} que, para aligerar la notación, en adelante se denotará como y_{t+1}. Concluidas las iteraciones sobre la variable primal, el valor de μ se actualiza con un paso de ascenso de gradiente proyectado, dado por:

$$\mu_{t+1} = \text{Proj}_{\mathbb{R}^{d_c}_{\geq 0}}(\mu_t + \eta_2 \nabla_\mu L(\mu_t, y_{t+1})). \tag{3.24}$$

El número de iteraciones T necesario dependerá de la precisión requerida y de las propiedades de $L(\mu, y)$ con respecto a μ. Después de T iteraciones, se obtiene (y_T, μ_T) como una aproximación a $y^*(x), \mu^*(x)$.

La aplicación de este algoritmo a los problemas (3.9) y (3.11) genera como salidas $y_{F,t}^{T_F}, \mu_{F,t}^{T_F}$ y $y_{g,t}^{T_g}, \mu_{g,t}^{T_g}$, que son utilizadas en el algoritmo 1.

El algoritmo MaxMin descrito en esta sección se presenta de forma compacta en el algoritmo 2.

Algoritmo 2: Algoritmo de gradiente MaxMin para resolver (3.21)

Entradas: conjunto $\mathcal{Y}$; tamaños de paso η_1, η_2; contadores T, T_y; parámetro γ
Inicialización: puntos iniciales y_0, μ_0
para $t = 0, 1, \ldots, T-1$ **hacer**
  Establecer $y_{t,0} = y_t$
  para $t_y = 0, 1, \ldots, T_y - 1$ **hacer**
    Calcular $y_{t,t_y+1} = \text{Proj}_{\mathcal{Y}}(y_{t,t_y} - \eta_1 \nabla_y L(\mu_t, y_{t,t_y}))$
  fin
  Establecer $y_{t+1} = y_{t,T_y}$
  Calcular $\mu_{t+1} = \text{Proj}_{\mathbb{R}^{d_c}_{\geq 0}}(\mu_t + \eta_2 \nabla_\mu L(\mu_t, y_{t+1}))$
fin
Salidas: (y_T, μ_T)

Nótese que el algoritmo es un enfoque híbrido entre un algoritmo dual y un algoritmo primal-dual. Cuando T_y es grande, puede entenderse como un algoritmo dual puro, ya que calcula una aproximación para $y^*(\mu_t, x_t)$ para cada valor de μ. Por otro lado, es un algoritmo primal-dual puro cuando T_y es igual a 1, ya que y_t y μ_t se actualizan simultáneamente.

Tal y como se mencionó en el objetivo ii) al inicio de esta sección, presentado ya el algoritmo, solo resta discutir su velocidad de convergencia. Desde una perspectiva computacional, el algoritmo 2 será más eficiente para valores más pequeños de T_y. Sin embargo, las garantías de convergencia para $T_y = 1$ solo se mantienen bajo ciertas condiciones sobre $g^c(x, y)$. Para el caso especial donde $g^c(x, y)$ es afín en y y puede expresarse como

$$g^c(x, y) = g_1^c(x)^T y - g_2^c(x), \tag{3.25}$$

el algoritmo MinMax garantiza una convergencia lineal con $T_y = 1$, tal y como se formaliza en el teorema 3. La demostración del teorema 3 consiste básicamente en aplicar los resultados del teorema 4.47 en [102] a los problemas (3.9) y (3.11). Los lectores que tengan particular interés en la misma pueden consultar [30].

Enunciado del teorema 3

Teorema 3 (Convergencia lineal interna) *Considérese el problema BLO definido en* (3.1) *con* $\mathcal{Y} = \mathbb{R}^{d_y}$ *y* $g^c(x, y)$ *como en* (3.25). *Considérese, además, que se cumplen las suposiciones 1–4 y las condiciones en el teorema 2. Supóngase, asimismo, que para cualquier* $x \in \mathcal{X}$, *existen constantes* $s_{\text{mín}}$ *y* $s_{\text{máx}}$ *tales que* $0 < s_{\text{mín}} \leq \sigma_{\text{mín}}(g^c(x)) \leq \sigma_{\text{máx}}(g^c(x)) \leq s_{\text{máx}} < \infty$ [a]. *Si i) se cumple que* $\gamma > \frac{l_{f,1}}{\alpha_g}$, *ii) en el bucle interno se fijan los valores del número de iteraciones de forma que* $T_g = \mathcal{O}(\log(\epsilon_g^{-1})), T_F = \mathcal{O}(\log(\epsilon_F^{-1}))$ *y* $T_y^g = T_y^F = 1$ *y iii) se escogen tamaños de paso constantes de valores adecuados, entonces las iteraciones generadas por el algoritmo 2 satisfacen:*

$$\|\mu_{t,g}^{T_g} - \mu_g^*(x_t)\|^2 = \mathcal{O}(\epsilon_g) \quad y \quad \|y_{t,g}^{T_g} - y_g^*(x_t)\|^2 = \mathcal{O}(\epsilon_g)$$
$$\|\mu_{t,F}^{T_F} - \mu_F^*(x_t)\|^2 = \mathcal{O}(\epsilon_F) \quad y \quad \|y_{t,F}^{T_F} - y_F^*(x_t)\|^2 = \mathcal{O}(\epsilon_F).$$

[a] $\sigma_{\text{mín}}(g^c(x))$ y $\sigma_{\text{máx}}(g^c(x))$ indican los valores singulares mínimo y máximo de la matriz de restricciones $g^c(x)$.

Para describir el resultado de manera más informal supóngase que las tolerancias se fijan de forma que $\epsilon_g = \epsilon_F = \epsilon$. El teorema 3 establece que, en el caso de que las restricciones sean afines, un algoritmo primal-dual en el que tanto los multiplicadores como las variables primales se actualizan con un sencillo

algoritmo de gradiente converge en un número de iteraciones proporcional a $-\log(\epsilon)$. La presencia de restricciones afines (incluidas las lineales) es común en muchos problemas de ingeniería. El capítulo 4 presenta un ejemplo en el contexto de optimización de redes de transporte donde la suma de los flujos de los pasajeros (que son variables del nivel inferior) se limita por la capacidad de los enlaces de la red construida (que son variables del nivel superior). Esta restricción de capacidad es afín y, por lo tanto, el algoritmo BLOCC que utiliza un algoritmo primal-dual en el nivel inferior, convergerá de forma rápida.

3.4. Ejemplo didáctico y resumen ejecutivo

Esta última sección se dedica a i) presentar un ejemplo sencillo que permite entender mejor varias de las definiciones introducidas en el capítulo y ii) realizar un resumen ejecutivo de los resultados obtenidos a lo largo de las secciones precedentes.

3.4.1. Ejemplo didáctico

Para finalizar este capítulo, se presenta un ejemplo didáctico de una optimización BLO con CCs. El propósito que se persigue es doble. En primer lugar, se pretende ganar intuición sobre los problemas de BLO con CCs. En segundo lugar, se pretende demostrar que el algoritmo BLOCC converge a mínimos locales del problema bilineal. Para ello, se propone un problema en el que: i) tanto las variables del nivel superior como inferior son unidimensionales, lo que facilita la visualización de las funciones a optimizar, ii) el problema del nivel inferior tiene solución analítica, lo que permite obtener una expresión cerrada para la función $F(x)$ del nivel superior, y iii) las funciones objetivo y las restricciones satisfacen las suposiciones 1, 2, 3, y 4.

Teniendo todo esto en cuenta, el problema que se resuelve es

$$\min_{x\in[0,3]} \; F(x) = f(x, y_g^*(x)) = \frac{e^{-y_g^*(x)+2}}{2+\cos(6x)} + \frac{1}{2}\log((4x-2)^2+1) \tag{3.26a}$$

$$\text{con} \;\; y_g^*(x) \in \arg\min_y \; g(x,y) = (y-2x)^2, \tag{3.26b}$$

$$\text{s. a:} \quad y - x \leq 0, \tag{3.26c}$$

donde la CC simplemente establece que $y \leq x$. El problema (3.26) satisface todas las suposiciones requeridas por los teoremas 2 y 3. Además, el problema inferior (3.26b)-(3.26c) puede resolverse de forma analítica. Esto permite obtener la expresión de $y_g^*(x)$ y sustituirla en (3.26a), obteniendo así $F(x) = f(x, y_g^*(x))$ de forma explícita.

Así, pues el primer paso es la resolución del problema del nivel inferior. El objetivo en (3.26b) es fuertemente convexo y, de no existir la restricción, el óptimo estaría en el punto $y = 2x$. No obstante, la solución $y = 2x$ no cumple la restricción (3.26c). Cuando dicha restricción se tiene en cuenta, la condición KKT asociada a la restricción se activa, con lo que se tiene que el óptimo del nivel inferior es $y_g^*(x) = x$. Sustituyendo $y_g^*(x) = x$ en (3.26a), el problema BLO en (3.26) se reduce a optimizar

$$\min_{x\in[0,3]} \; F(x) = f(x, y_g^*(x)) = f(x,y)|_{y=x} = \frac{e^{-x+2}}{2+\cos(6x)} + \frac{1}{2}\log((4x-2)^2+1). \tag{3.27}$$

Por lo tanto, la aplicación del algoritmo 1 queda como

Algoritmo 3: BLOCC aplicado a ejemplo didáctico

Entradas: conjuntos $\mathcal{X}$,$\mathcal{Y}$; tamaño del paso η, η_g, η_F; contadores T_g, T_F; parámetros de precisión ϵ_g,ϵ_F, parámetro γ

Inicialización: puntos iniciales x_0, $y_{g,0}$, $\mu_{g,0}$, $y_{F,0}$, $\mu_{F,0}$

para $t = 0, 1, \ldots, T-1$ **hacer**

  Calcular la solución ϵ_g-aproximada $(y_{g,t}^{T_g}, \mu_{g,t}^{T_g}) = (x, 2x)$

  Calcular la solución ϵ_F-aproximada $(y_{g,t}^{T_g}, \mu_{g,t}^{T_g}) = (x, \frac{e^{-x+2}}{2+\cos(6x)} + \gamma 2x)$

  Actualizar x_{t+1} como $x_{t+1} = \text{Proj}_{[0,3]}(x_t - \eta(\frac{6\sin(6x)e^{-x+2}}{(2+\cos(6x))^2} + \frac{4(4x-2)}{(4x-2)^2+1} - \frac{e^{-x+2}}{2+\cos(6x)}))$

fin

Salidas: $(x_T, y_{g,T}^{T_g})$.

La figura 3.2 representa la función a optimizar del nivel superior así como los resultados obtenidos al aplicar el algoritmo BLOCC a este problema. En la subfigura de la izquierda se proporciona una representación en 3 dimensiones (correspondiente a la función $f(x, y)$) y en la subfigura de la derecha se proporciona una representación en dos dimensiones (correspondiente a la función $F(x) = f(x, y)|_{y=x}$). Más concretamente, en la subfigura izquierda se representan 3 elementos. El primero, correspondiente a la curva tridimesional, representa los valores de la función $f(x, y)$ en el intervalo $(x, y) \in [0, 3]^2$. El segundo, correspondiente a la línea roja discontinua, representa la intersección de la superficie $f(x, y)$ y el plano $y = x$. El tercero, correspondiente con los puntos rojos, son las soluciones a las que converge el algoritmo BLOCC con $\gamma = 5$ y 200 valores de inicialización diferentes de las variables x e y. Los resultados demuestran que: i) BLOCC efectivamente genera soluciones factibles (es decir, óptimas para el nivel inferior) ya que todos los puntos rojos satisfacen que $x = y$ y ii) los puntos rojos están en mínimos locales de la línea roja. Para interpretar mejor los resultados, la función unidimensional $F(x)$, resultante de la intersección de $f(x, y)$ e $y = x$, se representa en color azul en la subfigura de la derecha. Nótese que esta función se corresponde con la expresión proporcionada en (3.27). La subfigura muestra claramente que, dentro del intervalo $[0, 3]$, existen 4 mínimos locales. Al igual que en el caso de la subfigura de la izquierda, los puntos rojos representan las soluciones a las que converge el algoritmo BLOCC. Esta representación demuestra claramente que, en los 200 casos considerados, BLOCC converge a uno de los cuatro mínimos locales de la función $F(x)$, verificando la efectividad del método propuesto.

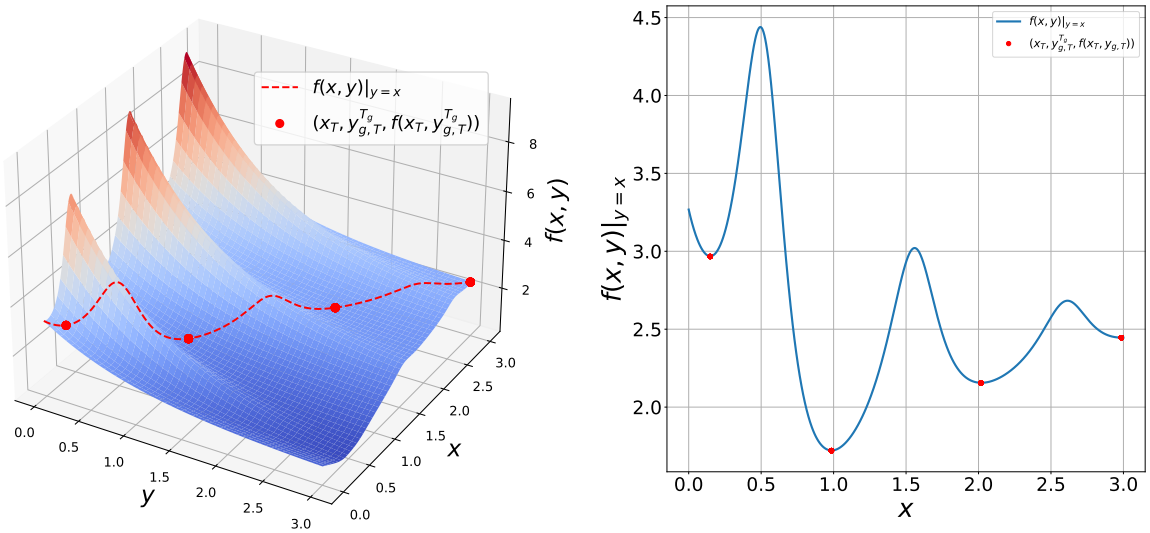


Figura 3.2: Resultados obtenidos tras la aplicación del algoritmo BLOCC al problema definido en (3.26). La figura de la izquierda representa la función objetivo del nivel superior. La línea $F(x) = f(x, y)|_{y=x}$ se representa tanto en la figura izquierda (línea roja punteada) como en la figura derecha (línea azul continua). Los puntos rojos en ambas figuras representan las soluciones a las que converge el algoritmo BLOCC para este problema.

3.4.2. Resumen ejecutivo

En este capítulo se ha propuesto el algoritmo BLOCC: un algoritmo con garantías de convergencia en tiempo polinómico para resolver problemas de BLO con CCs. Las ideas básicas detrás del diseño han sido: i) redefinir el problema original mediante una formulación que penaliza tanto el *gap* de optimalidad del problema inferior como la violación de las CCs y ii) resolver dicho problema mediante un algoritmo de PGD sobre las variables del nivel superior.

Para ello, tras presentar el problema BLO con CCs original (3.1), el primer paso ha consistido en presentar una reformulación de dicho problema (3.2), cuya función objetivo se ha denotado como $F_\gamma(x)$. Las diferencias principales de (3.2) con respecto a (3.1) son: i) la consideración de una función de penalización que depende de la función de valor $v(x)$, ii) la consideración de un término dual en el objetivo asociado a las CCs y iii) el uso de una formulación max-min para atacar el problema del nivel inferior.

Bajo ciertas suposiciones sobre las CCs y las funciones objetivo de los niveles superior e inferior, se ha: i) caracterizado la relación entre (3.1) y (3.2), ii) demostrado la convexidad de la reformulación con respecto a y y iii) demostrado la suavidad de Lipschitz de $F_\gamma(x)$ con respecto a x. Estas dos últimas propiedades han sido esenciales para abordar la resolución de (3.2) mediante el algoritmo BLOCC, cuyo pseudocódigo se proporciona en el algoritmo 1.

Una aspecto fundamental en el diseño del algoritmo BLOCC ha consistido en desacoplar las variables de los niveles superior e inferior mediante el uso de dos bucles anidados. En cada iteración del bucle externo, las variables del nivel superior se actualizan utilizando un algoritmo de PGD. El cálculo del gradiente de $F_\gamma(x)$ con respecto a x exige ejecutar el bucle interior en el que, para un valor determinado de x, se ejecutan los algoritmos MaxMin que hallan la solución de las variables primales y duales del nivel inferior (y, μ). El elemento crítico del algoritmo es la aproximación de $\nabla F_\gamma(x)$ a partir de (3.18), una expresión que considera tanto el impacto de x en el nivel superior como la reacción del nivel inferior ante cambios en x.

La parte final del capítulo se ha dedicado a caracterizar las propiedades del algoritmo BLOCC. Dos han sido los resultados principales. En primer lugar, se ha demostrado que, bajo las condiciones establecidas, el algoritmo BLOCC converge de manera eficiente a una solución cuyo gradiente es arbitrariamente pequeño (véase el enunciado del teorema 2). Asimismo, se ha caracterizado la complejidad computacional asociada a los subproblemas max-min internos (véase el enunciado del teorema 3), que son la mayor fuente de complejidad del algoritmo BLOCC.

El siguiente paso consiste en reforzar la caracterización del algoritmo BLOCC mediante su uso en aplicaciones reales, comparando su rendimiento y la calidad de la solución obtenida con otros métodos del estado del arte. Este es precisamente el objetivo del siguiente capítulo, donde se utiliza el algoritmo BLOCC en el contexto de optimización de hiperparámetros de arquitecturas de aprendizaje automático, así como en el contexto del diseño de redes de transporte en las que los usuarios toman decisiones de viaje de manera racional.

Capítulo 4

Experimentos

En el capítulo anterior, se ha propuesto un algoritmo novedoso para la resolución de problemas de BLO con CCs. En este capítulo se busca ilustrar la utilidad de dicho algoritmo en dos aplicaciones en el ámbito de la ingeniería. La primera aplicación, presentada en la sección 4.1, aborda el problema de la optimización de hiperparámetros de un algoritmo de aprendizaje automático. En la segunda aplicación, descrita en la sección 4.2, se presenta un problema de diseño (optimización) de la infraestructura de una red de transporte en presencia de un competidor y teniendo en cuenta las preferencias de los usuarios mediante un reparto logístico de la demanda. Tal y como se ha indicado en el capítulo introductorio el código utilizado para ejecutar los experimentos se encuentra disponible en `https://github.com/frealr/TFG-BLO`.

4.1. Optimización de hiperparámetros

La diseño óptimo de una arquitectura (modelo) de aprendizaje automático implica la selección (aprendizaje) de tanto una serie de parámetros como de hiperparámetros. Los valores óptimos de los parámetros se estiman utilizando datos de entrenamiento, mientras que los valores óptimos se estimas utilizando datos de validación. Aunque no siempre se define de forma explícita, la BLO constituye el marco formal para optimizar los hiperparámetros de un modelo de aprendizaje automático.

- En el *nivel inferior* de la BLO se optimizan los parámetros del modelo utilizando un conjunto de datos de entrenamiento. Este proceso implica ajustar los pesos y las variables internas del modelo para minimizar una función de coste predefinida, como la entropía cruzada en clasificación o el error cuadrático medio en regresión. El objetivo es encontrar los parámetros que mejor capturen las características subyacentes de los datos de *entrenamiento*, logrando así un buen desempeño en esta fase.
- Simultáneamente, en el *nivel superior* de la BLO, se optimizan los hiperparámetros del modelo utilizando un conjunto de datos de *validación*. Los hiperparámetros son variables que configuran el comportamiento del modelo y su proceso de entrenamiento, tales como la tasa de aprendizaje, el número de capas de una red neuronal o las constantes asociadas a los distintos regularizadores, entre otros. La optimización de hiperparámetros[1] se realiza para maximizar el rendimiento del modelo en el conjunto de validación, lo que garantiza que el modelo no solo se ajuste bien a los datos de entrenamiento, sino que también generalice adecuadamente a datos no vistos con anterioridad.

[1]El significado del término hiperparámetro, ha evolucionado a lo largo de los años, incluyendo en la actualidad variables tanto continuas como discretas, tales como el número de capas o el tipo de red neuronal. Este TFG se centra en optimización continua y, por lo tanto, el término hiperparámetro se circunscribe a variables como los pesos asociados a los distintos regularizadores.

Tradicionalmente, la búsqueda de hiperparámetros del nivel superior se realiza mediante una búsqueda exhaustiva, con técnicas como la búsqueda en rejilla (*grid search*) o la búsqueda aleatoria (*random search*). Esto implica que, en primer lugar, se define (crea) un conjunto de valores para cada uno de los hiperparámetros del modelo. Posteriormente, para cada una de las combinaciones de hiperparámetros, se resuelve el problema del nivel inferior, obteniendo los parámetros óptimos del modelo mediante técnicas como el GD estocástico. Finalmente, se evalúa cada uno de estos modelos entrenados en el conjunto de validación y se elige la combinación de hiperparámetros que ofrece mejores resultados.

Sin embargo, la búsqueda exhaustiva de hiperparámetros puede ser computacionalmente costosa y no siempre garantiza encontrar la combinación óptima, especialmente cuando el número de hiperparámetros supera la media docena. Por ejemplo, si tenemos 6 hiperparámetros distintos y queremos probar 5 valores posibles de cada uno de ellos, una búsqueda exhaustiva requiere resolver el problema del nivel inferior 5^6 veces, lo cual supone entrenar más de 4000 modelos diferentes. Aquí es donde la optimización binivel puede suponer una ventaja frente a los métodos de búsqueda clásicos. Al optimizar parámetros e hiperparámetros en niveles distintos pero interconectados, se disminuye radicalmente la complejidad computacional y, simultáneamente, mejora la capacidad de generalización, superando las limitaciones de las técnicas de búsqueda exhaustiva tradicionales.

Para ilustrar la búsqueda de hiperparámetros en el contexto de BLO y analizar el rendimiento del algoritmo BLOCC se va a considerar el entrenamiento de una SVM, una arquitectura de aprendizaje automático con restricciones. Las SVMs [105] son modelos ampliamente utilizados en tareas de clasificación y regresión debido a su robustez y efectividad en diferentes tipos de datos. Sin embargo, el rendimiento de una SVM depende en gran medida de la correcta configuración de sus hiperparámetros, más concretamente el parámetro de regularización. Utilizando la optimización binivel, se pueden ajustar estos hiperparámetros de manera más eficiente y precisa, asegurando que el modelo SVM implemente una configuración de parámetros e hiperparámetros eficiente y, por lo tanto, que mejore su rendimiento. A continuación, se detalla cómo se implementa esta técnica en el contexto de SVMs.

4.1.1. Formulación del problema

En el contexto de clasificación binaria, las SVMs son un modelo de aprendizaje automático que busca encontrar el hiperplano óptimo que separa las muestras de diferentes clases, tratando de maximizar el **margen**, definido como la distancia entre dicho hiperplano y las dos muestras más cercanas de cada clase al hiperplano.

Más concretamente, en el caso lineal[2] y para la resolución de un problema de clasificación binaria con etiquetas $\{-1, 1\}$ y datos de entrenamiento $\mathcal{D}_{tr} = \{(z_{tr,i}, l_{tr,i})\}_{i=1}^{I_{tr}}$, el hiperplano óptimo se puede encontrar resolviendo el siguiente problema de optimización

$$\min_{w,b,\xi} \ \frac{1}{2}\|w\|^2 + c\|\xi\|_1 \tag{4.1a}$$

$$\text{s.a: } (z_{tr,i}^\top w + b) l_{tr,i} \geq 1 - \xi_i \quad \forall\, i \in \{1, \ldots, I_{tr}\}, \tag{4.1b}$$

donde el conjunto $\mathcal{D}_{tr}$ contiene I_{tr} parejas de datos de entrenamiento, cada una de ellas compuesta por un vector de entrada $z_{tr,i} \in \mathbb{R}^F$ con F características y su etiqueta asociada $l_{tr,i} \in \{-1, 1\}$. Asimismo, $w \in \mathbb{R}^F$ y $b \in \mathbb{R}$ definen el hiperplano que separa las dos clases y $\xi \in \mathbb{R}^{I_{tr}}$ es un vector cuya entrada ξ_i contiene la violación de separación del hiperplano de la muestra i. La restricción en (4.1b) busca la correcta separación de las muestras. Si una muestra i no está al lado correcto del hiperplano, se activa la variable ξ_i asociada a dicha muestra, permitiendo una violación del mismo, a costa de penalizar la función objetivo. El objetivo en (4.1a) trata de maximizar el margen del hiperplano (que se puede

[2]Normalmente, para lidiar con hiperplanos de separación no lineales, los modelos SVM utilizan *kernels*. Estos son funciones no lineales que transforman los datos mediante una función no lineal a un espacio de alta dimensionalidad, separando mediante un hiperplano en dicho espacio. Esto provoca que el hiperplano de separación no sea lineal en el espacio original. No obstante, este TFG se centrará en el caso lineal.

calcular como $1/\|w\|$) a la vez que intenta minimizar la norma del vector ξ, cuyas entradas representan la violación (error de clasificación) de las muestras con respecto al hiperplano. El *trade-off* entre ambos términos en el objetivo viene determinado por el hiperparámetro c: un mayor valor de dicho hiperparámetro provocará que haya menos violaciones del hiperplano a coste de un menor margen, lo cual puede dar lugar a un sobreajuste a los datos de entrenamiento. Por el contrario, si c es pequeño, el margen será elevado, a coste de clasificar incorrectamente un mayor número de muestras. Para las SVMs de margen duro, no se tolera la clasificación incorrecta de los datos, lo cual puede suponer que el problema sea infactible si los datos no son separables por clases. Esto equivale a no considerar la variable ξ o establecer $c = \infty$ en (4.1). Por el contrario, para las SVMs de margen suave (que son las implementadas en la práctica), se permite que algunas muestras se clasifiquen de forma incorrecta.

Como se ha comentado, el hiperparámetro c busca balancear en el objetivo el margen del hiperplano y las violaciones al mismo. Para la determinación óptima de este parámetro mediante BLO, se propone la siguiente formulación

$$\min_{c\geq 0} \mathcal{L}_{\mathcal{D}_{val}}(w^*, b^*) = \sum_{(z_{val}, l_{val})\in\mathcal{D}_{val}} \exp\left(1 - \left(z_{val}^\top w^* + b^*\right) l_{val}\right) \tag{4.2a}$$

$$\text{con } w^*, b^*, \xi^* = \arg\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + c\|\xi\|_1 \tag{4.2b}$$

$$\text{s.a: } (z_{tr,i}^\top w + b) l_{tr,i} \geq 1 - \xi_i \quad \forall\, i \in \{1, \ldots, I_{tr}\}, \tag{4.2c}$$

donde el nivel inferior es exactamente igual al problema (4.1) y el nivel superior busca encontrar el hiperparámetro c óptimo mediante la minimización del error en el conjunto de validación $\mathcal{D}_{val} = \{(z_{val,i}, l_{val,i})\}_{i=1}^{I_{val}}$. Más concretamente, este error busca penalizar las violaciones a la separación del hiperplano en el conjunto de validación: si el término dentro de la exponencial es positivo (lo que implica que $(z_{val}^\top w^* + b^*) l_{val} \leq 1$), la muestra está incorrectamente clasificada y la exponencial toma un valor elevado. Por el contrario, si el término de la exponencial es negativo, la muestra está en el lado correcto del hiperplano y el coste asociado a esa muestra es muy pequeño [106].

Sin embargo, el problema en (4.2) no tiene CCs, por lo que no representa el escenario que se trata en este TFG. Por ello, se propone la siguiente reformulación del problema

$$\min_{c} \mathcal{L}_{\mathcal{D}_{val}}(w^*, b^*) = \sum_{(z_{val}, l_{val})\in\mathcal{D}_{val}} \exp\left(1 - \left(z_{val}^\top w^* + b^*\right) l_{val}\right) \tag{4.3a}$$

$$\text{con } w^*, b^*, \xi^* = \arg\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + \|c\|_1 \tag{4.3b}$$

$$\text{s.a: } (z_{tr,i}^\top w + b) l_{tr,i} \geq 1 - \xi_i \quad \forall\, i \in \{1, \ldots, I_{tr}\} \tag{4.3c}$$

$$\xi_i \leq c_i \qquad \forall\, i \in \{1, \ldots, I_{tr}\}. \tag{4.3d}$$

Al igual que en (4.2a), el objetivo del nivel superior (4.3a) es un coste de validación. La principal diferencia estriba en la variable del nivel superior, que ahora es un vector $c \in \mathbb{R}^{I_{tr}}$ donde cada entrada c_i representa el límite superior de la violación al hiperplano ξ_i de la muestra i del conjunto de entrenamiento. Intuitivamente, en (4.2) se quiere que ξ sea pequeño para permitir pocas violaciones, por lo que se incorpora su norma en (4.2b). En este caso se controla la magnitud de ξ utilizando un límite superior c en la CC (4.3d) y forzando a que dicho límite sea lo más pequeño posible, mediante la adición del término $\|c\|_1$ en el objetivo del nivel inferior (4.3b).

4.1.2. Resultados experimentales

En esta sección, se presentan los resultados de los experimentos llevados a cabo para evaluar el desempeño del algoritmo BLOCC para el diseño de un modelo SVM que resuelva los problemas de clasificación en dos bases de datos reales. Los detalles de los experimentos a realizados se discuten a continuación.

- Para cada experimento considerado, se evalúa. i) la función objetivo del nivel superior (conocida como función de coste o función de pérdidas), definida como $\sum_{(z_i,l_i)\in\mathcal{D}} \exp(1-(z_i^\mathsf{T} w^* + b^*)l_i)$ [cf. (4.2a) y (4.3a)] y la ii) exactitud, que es la proporción de muestras clasificadas correctamente. Estas métricas se calculan tanto para los conjuntos de validación ($\mathcal{D} = \mathcal{D}_{val}$) como de *test* ($\mathcal{D} = \mathcal{D}_{test}$).

- Adicionalmente, también se presentan resultados para dos algoritmos del estado del arte, LV-HBA [92] y GAM [45], ambos diseñados para resolver problemas de BLO con CCs de desigualdad. Al utilizar un mecanismo de gradiente implícito, el algoritmo GAM debe invertir una matriz. No obstante, a menudo la matriz asociada a (4.3) puede ser singular, lo que impide a GAM encontrar una solución. En consecuencia, en los resultados que se muestran en esta sección los algoritmos BLOCC y LV-HBA resuelven el problema en (4.3), mientras que GAM utiliza la formulación en (4.2).

Los resultados de los experimentos llevados a cabo se muestran en las 8 gráficas que componen las figuras 4.1 y 4.2. Cada una de las figuras está asociada a la evaluación de los algoritmos en un conjunto de datos: "diabetes" [107] en la figura 4.1 y "fourclass" [108] en la figura 4.2. En cada figura, las gráficas (a) y (c) contienen las métricas asociadas a los conjuntos de validación, mientras que las gráficas (b) y (d) muestran las métricas sobre los conjuntos de *test*. Además, con el fin de evaluar la velocidad de convergencia de los algoritmos, el eje horizontal de las gráficas representa el tiempo de cómputo, comenzando la representación de las métricas tras realizar la primera iteración de cada algoritmo. Para evaluar la robustez del algoritmo, se repiten los experimentos 10 veces, donde en cada simulación se escogen subconjuntos distintos de entrenamiento, validación y *test*. Las gráficas muestran tanto la media (línea sólida) como la desviación estándar (región sombreada) de la evolución de las métricas para las 10 realizaciones. Los valores de los parámetros LV-HBA y GAM se han escogido de forma que maximizan el rendimiento en las métricas presentadas para cada conjunto de datos, tal como se presenta en los experimentos de [92] y [45], respectivamente. Los valores de parámetros específicos utilizados para estos algoritmos, así como el resto de detalles de los escenarios simulados, se encuentran en el repositorio de código. Para el algoritmo BLOCC, se ha fijado $\eta = 1 \times 10^{-2}$ y $\gamma = 10$.

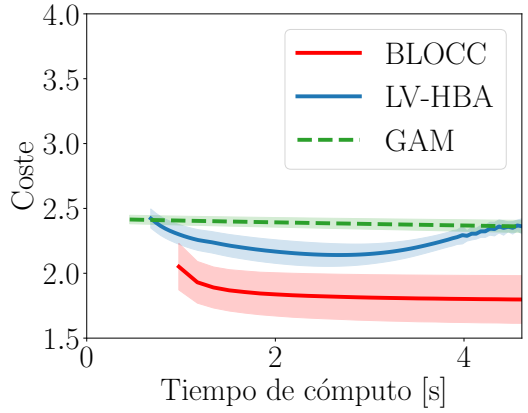
Focalizando el análisis, en primer lugar, en las dos gráficas que reportan resultados para la función de coste en el conjunto de validación (que es la métrica que los tres algoritmos considerados persiguen minimizar) se observa que, en ambas bases de datos (gráficas (a) de las figuras 4.1 y 4.2): i) BLOCC da lugar a la menor función de coste en ambos casos e ii) incluso teniendo en cuenta la desviación típica, que es ligeramente mayor en las métricas reportadas por el algoritmo BLOCC, el límite superior de BLOCC está por debajo del límite inferior de LV-HBA y GAM. En lo referente a la convergencia, se observa que BLOCC, tras el primer segundo de cómputo, converge de una forma rápida, dando lugar a un valor de coste medio razonablemente estable. Dicho comportamiento estable se mantiene en el algoritmo GAM, pero el valor medio de la función de coste es mayor. Esto contrasta con el comportamiento de LV-HBA, que pese a alcanzar un valor bajo de la función de coste relativamente rápido, transcurridos los 3 primeros segundos comienza a empeorar sus prestaciones e, incluso, a generar pequeñas oscilaciones. El comportamiento anómalo observado en LV-HBA se debe a su método de resolución del problema BLO. LV-HBA es un algoritmo iterativo que aborda un problema de un solo nivel en cada iteración. Este problema se define mediante una combinación lineal de los objetivos de los niveles superior e inferior del problema original. En cada iteración, la combinación lineal otorga un peso creciente al problema del nivel inferior. Por lo tanto, es lógico que los resultados "empeoren" en términos de la función objetivo del nivel superior en las iteraciones finales. Además, es importante recordar que una de las restricciones de un problema BLO es satisfacer la optimalidad en el nivel inferior, algo que en el caso de LV-HBA solo se logra cuando el algoritmo alcanza su convergencia.

Una vez analizadas las gráficas (a) de las figuras 4.1 y 4.2, el siguiente paso consiste en analizar la calidad de las soluciones generadas cuando se evalúan en la función de coste asociada al conjunto de test. Las gráficas (b) de las figuras 4.1 y 4.2 muestran la evolución temporal de estas métricas. En este caso, se observa que i) al evaluarse en datos nuevos que no se habían considerado durante

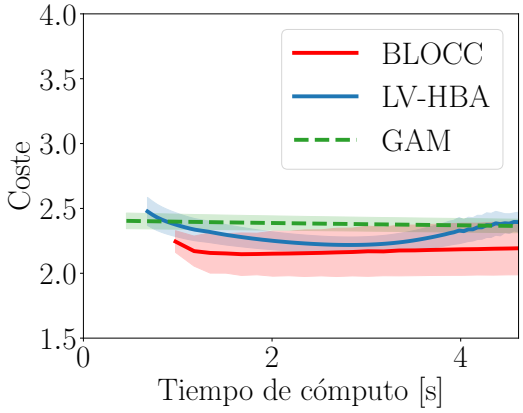
la optimización, las prestaciones de los tres algoritmos empeoran con respecto a las mostradas en las gráficas (a); ii) el algoritmo BLOCC sigue siendo el que, en media, obtiene mejores prestaciones, pero en este caso las zonas sombreadas que representan la desviación típica de los distintos algoritmos sí se solapan; y iii) las oscilaciones presentes en el algoritmo LV-HBA provocan que, en este caso, sea el que peor prestaciones acabe dando en el conjunto "diabetes", teniendo el algoritmo GAM un comportamiento significativamente más estable.

El último paso consiste en evaluar las prestaciones en términos de exactitud de clasificación, lo que se lleva a cabo en las gráficas (c) y (d) de las figuras 4.1 y 4.2, considerando la gráfica (c) el conjunto de validación y (d) el conjunto de *test*. En ambos casos se observa que: i) en media, los algoritmos BLOCC y LV-HBA alcanzan el mayor nivel de exactitud media, siendo ligeramente superior el algoritmo BLOCC en el conjunto de validación del conjunto "diabetes"; ii) las zonas sombreadas que representan la desviación típica se solapan y iii) las diferencias entre BLOCC y LV-HBA frente a GAM son notables, llegando a alcanzar niveles de exactitud superiores entre un 10-15 % (véanse las métricas para el conjunto de test de "fourclass" de la gráfica (d) en 4.2). También destaca el hecho que el algoritmo BLOCC converge más rápido, en poco más de un segundo, mientras que LV-HBA (en términos de optimalidad en el nivel inferior) y GAM (en términos de exactitud) lo hacen de una forma más pausada.

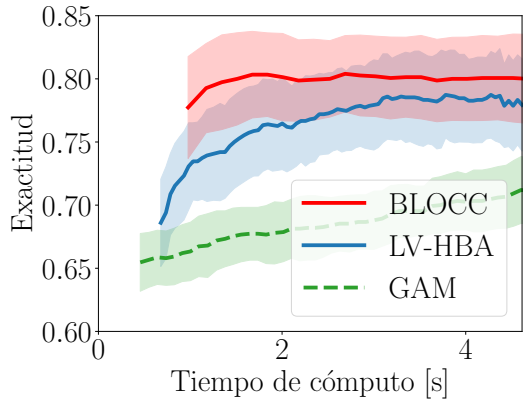
En suma, los resultados numéricos presentados en esta sección demuestran que i) en términos del problema de optimización planteado (selección de hiperparámetros para minimizar la función de coste en validación) el algoritmo BLOCC es el que ofrece mejores prestaciones en las dos bases de datos consideradas y ii) cuando se consideran otras métricas (función de coste en el conjunto de *test* o exactitud en los conjuntos de validación y *test*) el algoritmo BLOCC sigue dando lugar a las mejores prestaciones, si bien la distancia con las alternativas existentes en el estado del arte disminuye. A pesar de que, con el fin de entender mejor su comportamiento y limitaciones, el algoritmo BLOCC deba aplicarse a otros muchos escenarios relacionados con la selección de hiperparámetros, los resultados preliminares mostrados en esta sección pueden interpretarse como una primera validación del algoritmo diseñado en este TFG. Para los lectores interesados, en el repositorio de código se encuentran disponibles más experimentos asociados con distintas reformulaciones del problema de optimización resuelto en esta sección.



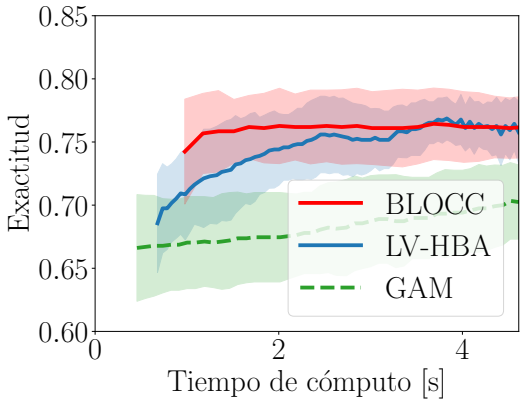
(a) Coste en datos de validación, conjunto de diabetes.



(b) Coste en datos de *test*, conjunto de diabetes.

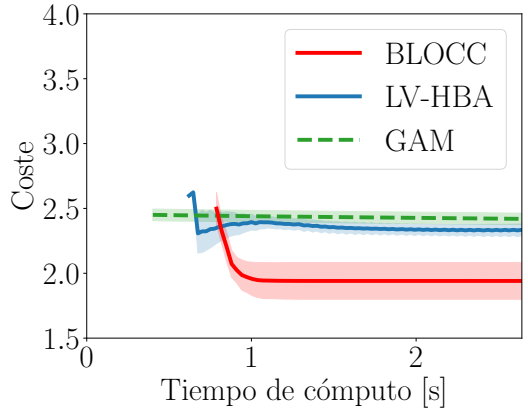


(c) Exactitud en datos de validación, conjunto de diabetes.

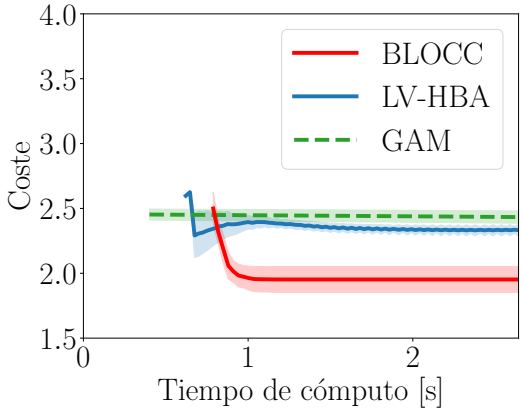


(d) Exactitud en datos de *test*, conjunto de diabetes.

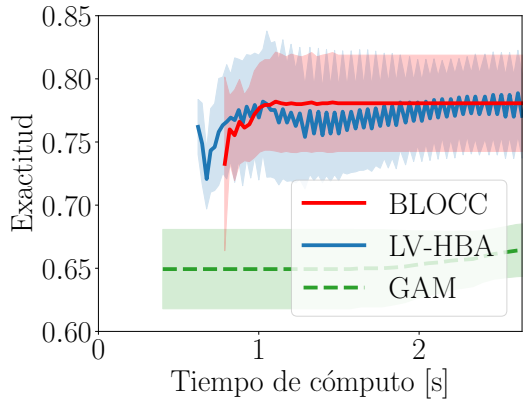
Figura 4.1: Resultados del experimento de optimización de hiperparámetros con un modelo SVM aplicado al dataset diabetes. Se prueban tres algoritmos BLO diferentes: BLOCC (este TFG), LV-HBA [92] y GAM [45], cada uno representado con un color diferente. En las gráficas (a) y (b) se evalúa la función de coste sobre los conjuntos de validación y *test*, respectivamente. En las gráficas (c) y (d) se evalúa la exactitud, que es la proporción de muestras clasificadas correctamente, para los conjuntos de validación y *test*. Finalmente, las líneas sólidas representan el valor medio de las 10 realizaciones consideradas de cada una de las métricas, y la región sombreada representa la $\pm$ desviación estándar.



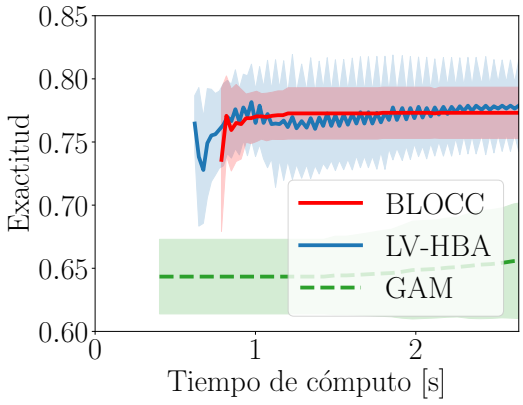
(a) Coste en datos de validación, conjunto de fourclass.



(b) Coste en datos de *test*, conjunto de fourclass.



(c) Exactitud en datos de validación, conjunto de fourclass.



(d) Exactitud en datos de *test*, conjunto de fourclass.

Figura 4.2: Resultados del experimento de optimización de hiperparámetros con un modelo SVM aplicado al dataset fourclass. Cada uno de los cinco gráficos representa una combinación de conjunto de datos y métrica. Se prueban tres algoritmos BLO diferentes: BLOCC (este TFG), LV-HBA [92] y GAM [45], cada uno representado con un color diferente. En las gráficas (a) y (b) se evalúa la función de coste sobre los conjuntos de validación y *test*, respectivamente. En las gráficas (c) y (d) se evalúa la exactitud, que es la proporción de muestras clasificadas correctamente, para los conjuntos de validación y *test*. Finalmente, las líneas sólidas representan el valor medio de las 10 realizaciones consideradas de cada una de las métricas, y la región sombreada representa la ± desviación estándar.

4.2. Aplicaciones a la planificación de redes de transporte

Esta sección aplica el algoritmo BLOCC a un problema de diseño de redes de transporte. Para ello, en la sección 4.2.1 se introducirá el problema que se quiere resolver, así como la formulación binivel asociada al mismo. A continuación se presentarán los resultados de la aplicación del algoritmo BLOCC y las alternativas anteriormente mencionadas (cuando sea posible) a una red de ejemplo de 3 nodos (sección 4.2.2), a una red de ejemplo de 9 nodos (sección 4.2.3) y a una red real en la ciudad de Sevilla (sección 4.2.4).

4.2.1. Introducción al problema

A continuación, se presenta formalmente el problema de diseño de la red de transporte. Se comienza con una descripción verbal y luego se introduce la notación asociada.

En la planificación de redes de transporte, el objetivo es diseñar (construir) una nueva red que conecte un conjunto $\mathcal{S}$ de estaciones para transportar pasajeros que deseen viajar desde un origen $o \in \mathcal{S}$ hasta un destino $d \in \mathcal{S}$ [109]. Los dos actores principales son el operador, que es responsable de construir y operar la red, y los pasajeros, que decidirán si usar la nueva red o un medio de transporte alternativo. Junto con $\mathcal{S}$, se le da al operador un conjunto (máximo) de enlaces a construir $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$.

Cualquier enlace en $\mathcal{A}$ conecta dos estaciones y tiene una capacidad decidida por el operador. Cuanto mayor sea la capacidad, mayor será el coste de construcción, pero también mayor será el número de usuarios que pueden viajar. Un enlace (i, j) puede construirse solo si $(i, j) \in \mathcal{A}$. Si la capacidad del enlace se establece en cero, entonces el enlace no se construye. Pasando ahora a los pasajeros, se considera un conjunto de mercados $\mathcal{K} \subseteq \mathcal{S} \times \mathcal{S}$. Cada par origen-destino (o, d) es un mercado diferente y cada mercado $(o, d) \in \mathcal{K}$ tiene su propio precio de viaje y su propia demanda. A menos que se indique lo contrario, se considera que existe demanda para todos los mercados, de modo que el conjunto $\mathcal{K}$ contiene todos los pares (o, d) tales que $o \neq d$.

Además, se asume que: i) solo existe un competidor (una red ya desplegada), ii) los pasajeros deciden si usar la red racionalmente, basándose en atributos del viaje como precios y tiempo de viaje, y iii) la demanda por mercado, los precios de los viajes y los tiempos de viaje por enlace son conocidos para todas las redes (nueva y competidora). Un aspecto clave en la literatura de transporte es cómo modelar ii). Los enfoques simplificados consideran una política de todo o nada, donde se define un nivel de utilidad para cada mercado y red (en este caso, la nueva red y la ya desplegada) y los pasajeros dentro del mismo mercado se modelan como un solo grupo, de modo que toda la demanda en un mercado se asigna al operador que, para ese mercado, proporciona la mayor utilidad a los pasajeros. Modelos más sofisticados tienen en cuenta la diversidad dentro de las preferencias de los pasajeros y la incertidumbre en las utilidades, lo que lleva a un modelo probabilístico bajo el cual los pasajeros eligen una de las redes con cierta probabilidad. El modelo más utilizado dentro de esta clase es el llamado modelo de elección *logit* [110, 111], que asigna probabilidades basadas en una función softmax de la forma

$$\frac{e^{u_i^{od}}}{e^{u_i^{od}} + e^{u_c^{od}}}, \tag{4.4}$$

con u_i^{od} siendo la utilidad de la red i y u_c^{od} la utilidad de la competencia. Este modelo de elección, que es no lineal y no convexo, es el considerado en este trabajo y juega un papel crítico en la formulación.

Para simplificar la exposición, se asumirá que solo existe un competidor y que el único atributo del viaje relevante para las decisiones de los usuarios es el tiempo de viaje. Ambas suposiciones pueden ser eliminadas de manera sencilla. La eliminación de la primera de ellas supone la adición de más términos en el denominador del reparto *logit* en (4.4). La segunda requiere cambiar la definición de las utilidades u_i^{od} y u_c^{od} para tener en cuenta otras variables además del tiempo, como el precio del viaje.

La motivación para una formulación bilevel en el contexto del diseño de redes de transporte es clara, ya que están involucrados dos actores con niveles de poder desiguales: el operador, que construye la red, y los pasajeros, que seleccionan la red a utilizar basándose en atributos del viaje que dependen de la topología que se ha construido. Además, el objetivo es maximizar el beneficio del operador mientras se tienen en cuenta las decisiones (racionales y basadas en la utilidad) tomadas por los pasajeros. Por lo tanto, el nivel superior del problema tiene en cuenta el objetivo del operador (minimizar los costes de construcción mientras se maximiza la cantidad de demanda atraída), siendo las variables de optimización la capacidad de cada uno de los enlaces. Por el contrario, el problema de nivel inferior tiene en cuenta el objetivo de los pasajeros (maximizar su utilidad de viaje), siendo las variables de optimización de los pasajeros la proporción (fracción) de la demanda que, para un mercado dado, utiliza cada enlace. Como quedará claro más adelante, esto equivale a diseñar, para cada mercado, tanto la ruta en la nueva red como el número total de pasajeros que usarán esa red. Tanto las variables de nivel superior como las de nivel inferior deben satisfacer múltiples restricciones. Más importante para el trabajo en cuestión, el nivel inferior necesita satisfacer un conjunto de restricciones de capacidad que garantizan que el número de pasajeros (agregado a través de los mercados) que utilizan un enlace nunca puede exceder la capacidad del enlace establecida por el operador. Esta última restricción, que es crítica en el diseño de transporte, acopla las variables de los niveles superior e inferior y, por lo tanto, motiva el uso de BLOCC para el problema en cuestión. La figura 4.3 proporciona un ejemplo de diseño de red de transporte. Específicamente, la figura representa: (a) los conjuntos de estaciones y enlaces, $\mathcal{S}$ y $\mathcal{A}$, junto con el tiempo de viaje en cada uno de ellos (entrada para el diseño), (b) la matriz de demanda para todos los mercados en $\mathcal{K}$ (entrada para el diseño), (c) la red construida y la

capacidad de la misma (salida del diseño) y (d) los pasajeros atendidos por la red construida (salida del diseño). La notación utilizada en la figura se introduce en los siguientes párrafos.

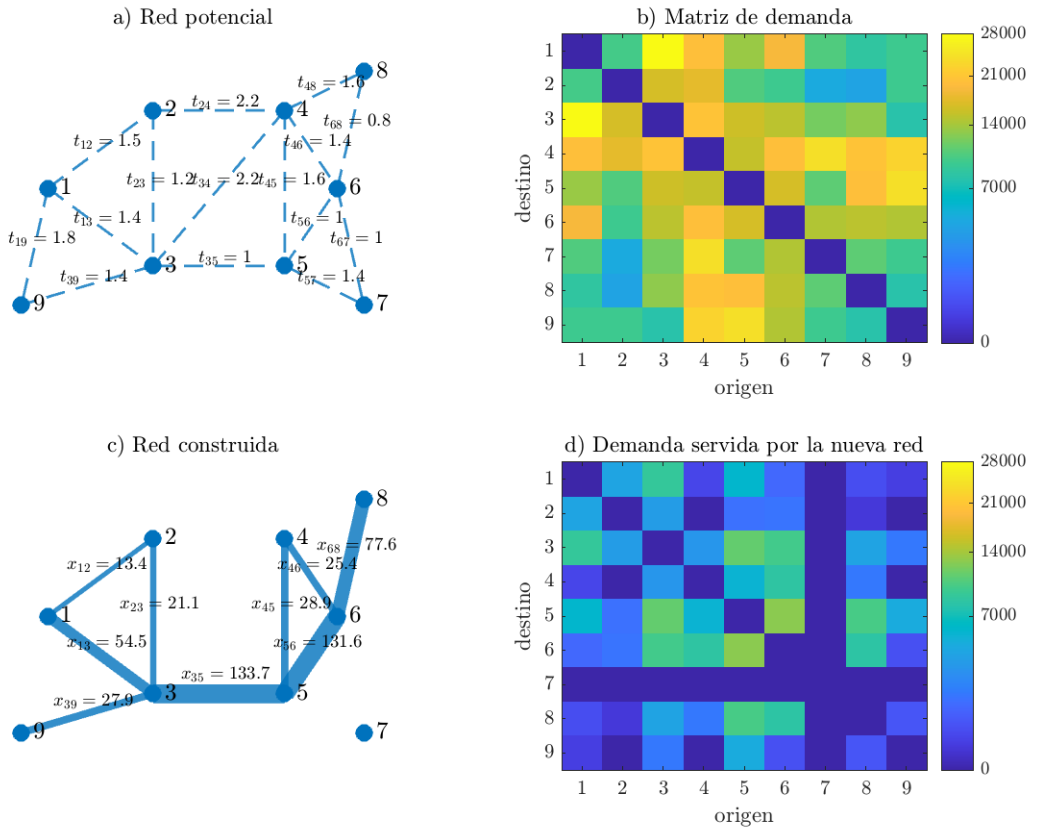


Figura 4.3: Ejemplo de diseño de una red de transporte. La figura (a) representa el conjunto de estaciones $\mathcal{S} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ y el conjunto de enlaces $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$. El número de nodos es $|\mathcal{S}| = 9$ y el número de enlaces es $|\mathcal{A}| = 30$ (15 segmentos con dos orientaciones cada uno). Nótese que $\mathcal{A}$ no es un grafo completo ya que, por ejemplo, $(1, 4) \notin \mathcal{A}$. El valor mostrado junto a cada enlace representa el tiempo de viaje. El mapa de calor en la figura (b) representa $\{w^{od}\}_{\forall (o,d) \in \mathcal{K}}$, la demanda existente entre todos los pares (o, d). Dado que se consideran todos los mercados, $|\mathcal{K}| = 9 \cdot 8 = 72$ valores son proporcionados (los que están en los elementos fuera de la diagonal del mapa de calor). La figura (c) representa la red construida, donde, para facilitar la visualización, se ha asumido que la capacidad es simétrica, de modo que $x_{ij} = x_{ji}\ \forall (i, j)$. En esta red se observa que: i) cada enlace tiene su propia capacidad, representada por el ancho del enlace que conecta los nodos, y ii) algunos enlaces no se construyen. Esto ocurre, por ejemplo, para el enlace $(2, 4)$ y para todos los enlaces que conectan el nodo 7 con el resto de la red. Finalmente, el mapa de calor en la figura (d) representa $\{w^{od} y^{od}\}_{\forall (o,d) \in \mathcal{K}}$, el número de pasajeros atendidos por la red construida. Cuando una estación (o un mercado) no es atendida por la nueva red, se tiene que el flujo asociado es $y^{od} = 0$, de modo que el número de pasajeros también es cero. Esto ocurre, por ejemplo, para todos los mercados que parten y llegan al nodo 7.

Matemáticamente, la capacidad del enlace $(i, j) \in \mathcal{A}$ se denota como x_{ij}. Las capacidades de todos los enlaces constituyen las variables de nivel superior. Las variables de nivel inferior, a menudo referidas en la literatura como flujos, son y^{od} para todos los $(o, d) \in \mathcal{K}$ y y_{ij}^{od} para todos los $(i, j) \in \mathcal{A}$ y $(o, d) \in \mathcal{K}$. La primera variable, y^{od}, representa la fracción de pasajeros del mercado (o, d) que viajará utilizando la nueva red. La segunda variable, y_{ij}^{od}, representa la fracción de pasajeros del mercado (o, d) que viajará utilizando el enlace (i, j) de la nueva red. Claramente, si $y_{ij}^{od} = 0$, entonces el enlace no pertenece a la ruta que los pasajeros siguen para viajar de o a d. Resumiendo, las variables de optimización son:

- $x_{ij} \in \mathbb{R}_{\geq 0}$, la capacidad construida para el enlace $(i, j) \in \mathcal{A}$. El número de variables de capacidad es $|\mathcal{A}|$. Si $x_{ij} = 0$, entonces el enlace no se construye.
- $y^{od} \in [0, 1]$, la fracción (proporción) de pasajeros del mercado $(o, d) \in \mathcal{K}$ que elige la nueva red

para su viaje. El número de variables de flujo es $|\mathcal{K}|$. Dado que se considera solo un competidor, $1 - y^{od}$ representa la fracción de pasajeros que usa la red incumbente. Si $y^{od} = 0$, entonces los pasajeros del mercado (o, d) no usan la nueva red.

- $y_{ij}^{od} \in [0, 1]$, la fracción de pasajeros del mercado $(o, d) \in \mathcal{K}$ que, al elegir la nueva red para viajar de o a d, utiliza el enlace $(i, j) \in \mathcal{A}$ en su ruta hacia el destino. El número de variables de flujo de enlace es $|\mathcal{A}||\mathcal{K}|$. Con esta definición, se infiere que $y_{ij}^{od} \leq y^{od}$. Además, si $y_{ij}^{od} = 0$, entonces el enlace (i, j) no pertenece a la ruta que los pasajeros siguen al viajar de o a d.

Para simplificar, se recopilan todas las variables de capacidad en x y todas las variables de flujo en y, de modo que:

- $x = \{x_{ij}\}_{\forall (i,j) \in \mathcal{A}}$ representa las variables de nivel superior a optimizar, y $\mathcal{X} = \mathbb{R}_{\geq 0}^{|\mathcal{A}|}$ representa el dominio de x.
- $y = \{y^{od}, \{y_{ij}^{od}\}_{\forall (i,j) \in \mathcal{A}}\}_{\forall (o,d) \in \mathcal{K}}$ representa las variables de nivel inferior a optimizar, y $\mathcal{Y} = [0, 1]^{|\mathcal{K}|} \times [0, 1]^{|\mathcal{A}||\mathcal{K}|}$ representa el dominio de y.

Además de las variables de optimización, el objetivo y las restricciones incluyen parámetros adicionales conocidos que, conjuntamente, describen el problema de optimización y su contexto. Estos son:

- w^{od}, la demanda total estimada (número de pasajeros) para el mercado $(o, d) \in \mathcal{K}$.
- m^{od}, los ingresos obtenidos por el operador de un pasajero en el mercado $(o, d) \in \mathcal{K}$.
- c_{ij}, el coste de construcción por pasajero asociado con el enlace $(i, j) \in \mathcal{A}$.
- t_{ij}, el tiempo de viaje para el enlace $(i, j) \in \mathcal{A}$.
- t_{ext}^{od}, tiempo de viaje en la red alternativa para los pasajeros en el mercado $(o, d) \in \mathcal{K}$.
- ω_t, el coeficiente asociado con el tiempo de viaje en la función de utilidad de los pasajeros.

Ahora, se está listo para introducir las formulaciones de los objetivos del problema BLO. Para el nivel superior, el operador de la red tiene como objetivo maximizar los beneficios y minimizar los costes de construcción, y por lo tanto su interés es

$$\min_{x \in \mathcal{X}} f(x, y) := -\left(\underbrace{\sum_{\forall (o,d) \in \mathcal{K}} m^{od} y^{od*}(x)}_{\text{beneficio}} \quad - \quad \underbrace{\sum_{\forall (i,j) \in \mathcal{A}} c_{ij} x_{ij}}_{\text{coste}} \right), \tag{4.5}$$

donde $y^{od*}(x)$ son los flujos de pasajeros óptimos de nivel inferior asociados con el diseño de red x.

El objetivo de nivel inferior requiere una explicación más detallada. Como ya se explicó, los usuarios establecen las variables de flujo de manera que se prefiera la red de transporte que ofrece una mayor utilidad, con las probabilidades dadas por un modelo *logit* (*softmax*, cf. (4.4)). Establecer el objetivo como una simple maximización de la utilidad lineal conduciría a una política de "todo o nada", que no se corresponde con el comportamiento observado en la práctica. Por otro lado, agregar una restricción que obligue a los flujos a seguir la política *softmax* lleva a una formulación no convexa. Por lo tanto, el enfoque considerado aquí es formular un objetivo dado por la transformada de Legendre del reparto de demanda *softmax* (véase [112, 113] para motivación adicional y garantías para este enfoque). Intuitivamente, esto significa que, en lugar de imponer el reparto *softmax* a priori, se formula un problema **convexo** cuyas condiciones KKT llevan al reparto *softmax*. Utilizando el hecho de que la transformada

de Legendre de una exponencial e^y es una función basada en la entropía negativa $y(\log(y)-1)$, el objetivo de nivel inferior es

$$g(x,y) := \min_{y\in\mathcal{Y}} -\Bigg(\underbrace{\sum_{(o,d)\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} w^{od}\omega_t t_{ij} y_{ij}^{od} + \sum_{(o,d)\in\mathcal{K}} w^{od}\omega_t t_{ext}^{od}(1-y^{od})}_{\text{utilidad de los pasajeros}} \tag{4.6}$$

$$- \underbrace{\sum_{(o,d)\in\mathcal{K}} w^{od} y^{od}(\log(y^{od})-1) - \sum_{(o,d)\in\mathcal{K}} w^{od}(1-y^{od})(\log(1-y^{od})-1)}_{\text{costes asociados a la transformada de Legendre}} \Bigg). \tag{4.7}$$

El primer término representa la utilidad de los pasajeros, que considera un parámetro de sensibilidad $\omega_t < 0$ y el tiempo de viaje t_{ij}. El segundo término es la transformada de Legendre de las utilidades presentes en el exponente de la distribución *softmax*. Dado que se tiene $y^{od} + (1-y^{od}) = 1$, se puede demostrar que al tomar la derivada de (4.7) con respecto a y^{od} y establecer el gradiente en cero se obtendrá el reparto *logit* deseado [113].

Habiendo introducido las variables de optimización, los parámetros y las funciones objetivo, a continuación se formula el problema de BLO, donde también se incorporan las restricciones adicionales requeridas

$$\min_{x\in\mathcal{X}} \; -\sum_{\forall(o,d)\in\mathcal{K}} m^{od} y^{od*} + \sum_{\forall(i,j)\in\mathcal{A}} c_{ij} x_{ij} \tag{4.8a}$$

$$\text{s.a: } (y^{od*}, y_{ij}^{od*}) = \arg\min_{y\in\mathcal{Y}} -\sum_{(o,d)\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} w^{od}\omega_t t_{ij} y_{ij}^{od} - \sum_{(o,d)\in\mathcal{K}} w^{od}\omega_t t_{ext}^{od}(1-y^{od}) \tag{4.8b}$$

$$+ \sum_{(o,d)\in\mathcal{K}} w^{od} y^{od}(\log(y^{od})-1) + \sum_{(o,d)\in\mathcal{K}} w^{od}(1-y^{od})(\log(1-y^{od})-1)$$

s.a:

$$\sum_{\forall j|(i,j)\in\mathcal{A}} y_{ij}^{od} - \sum_{\forall j|(j,i)\in\mathcal{A}} y_{ji}^{od} = \begin{cases} y^{od} & \text{si } i = o \\ -y^{od} & \text{si } i = d \\ 0 & \text{si } i \neq o \neq d \end{cases} \qquad \forall i, (o,d) \in \mathcal{S}\times\mathcal{K}, \tag{4.8c}$$

$$\sum_{\forall(o,d)\in\mathcal{K}} w^{od} y_{ij}^{od} \leq x_{ij} \qquad \forall (i,j)\in\mathcal{A}. \tag{4.8d}$$

En el problema formulado, (4.8a) es el objetivo de nivel superior (del operador) y (4.8b) es el objetivo de nivel inferior (de los pasajeros). Las igualdades en (4.8c) son las llamadas restricciones de conservación de flujo. Hay $|\mathcal{S}||\mathcal{K}|$ de estas restricciones, y solo involucran variables de nivel inferior. Las desigualdades en (4.8d) son las llamadas restricciones de capacidad. Hay $|\mathcal{A}|$ de estas restricciones, e involucran tanto variables de nivel superior como de nivel inferior, acoplando la optimización y motivando el uso de BLOCC. Nótese que, para redes con $|\mathcal{S}| = n$ estaciones (nodos): i) el número de CCs escala con $\mathcal{O}(n^2)$; y ii) cada una de las restricciones involucra $\mathcal{O}(n^2)$ variables. Por lo tanto, esto significa que incluso para redes de tamaño moderado (e.g. entre 30 y 50 nodos), se pueden tener miles de restricciones de acoplamiento que involucran millones de variables.

Hoja de ruta del experimento. Para proporcionar resultados numéricos que ilustren el comportamiento del algoritmo, se resuelve la optimización en (4.8) para tres escenarios:

1. El diseño de una red sintética simple de 3 nodos.
2. El diseño de una red sintética de 9 nodos, utilizada en la literatura de transporte anterior [109, 113].

3. El diseño de una red de metro real para la ciudad de Sevilla, España, con 24 nodos.

El primer escenario involucra 6 CCs y 48 variables, mientras que el último involucra alrededor de 100 CCs y 50,000 variables.

Para el escenario de la red de 3 nodos, se realizará un análisis comparativo contra los otros algoritmos del estado del arte para evaluar la eficacia del enfoque. En los otros dos escenarios, los algoritmos del estado del arte tienen dificultades para encontrar una solución; por lo tanto, para las redes de 9 nodos y Sevilla, se enfocará en proporcionar información sobre el rendimiento y comportamiento del algoritmo bajo diferentes configuraciones de parámetros, arrojando luz sobre la versatilidad y adaptabilidad del enfoque a redes de transporte del mundo real.

Antes de profundizar en la presentación y análisis de los resultados, son necesarias dos observaciones adicionales:

- Si bien uno de los objetivos de estos experimentos es comparar el algoritmo las prestaciones del algoritmo BLOCC con las de LV-HBA [92] y GAM [45], para el escenario en cuestión, el algoritmo GAM no puede implementarse, ya que la inversa de la matriz requerida en cada iteración para el problema en (4.8) no es tratable. De esta manera, solo se realizan los experimentos utilizando BLOCC y LV-HBA.

- El algoritmo BLOCC produce dos salidas para las variables de nivel inferior para cada iteración del algoritmo 1: $y_{F,t}^{T_F}$ y $y_{g,t}^{T_g}$. Si bien el Teorema 1 y el Teorema 2 garantizan que $(x_T, y_{F,T}^{T_F})$ es una solución ϵ-aproximada en el nivel inferior, el par $(x_t, y_{g,t}^{T_g})$ es estrictamente factible, lo que significa que $y_{g,t}^{T_g} = y_g^*(x_t)$. Dado que la factibilidad estricta es importante en aplicaciones que tratan con sistemas dinámicos (incluido el transporte), para comparar el desempeño del algoritmo usando ambas salidas, esta sección reportará los resultados utilizando como salida del nivel inferior tanto $y_{F,t}^{T_F}$ como $y_{g,t}^{T_g}$.

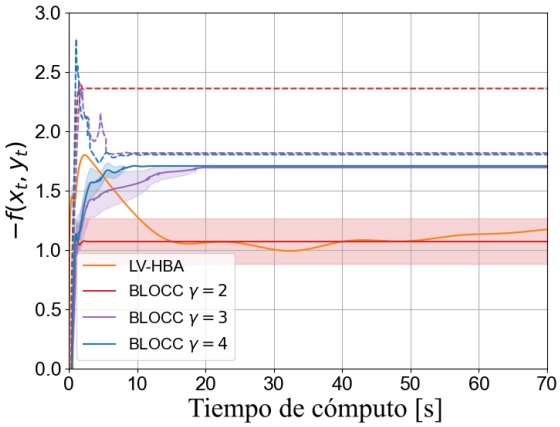


Figura 4.4: Evolución del objetivo de nivel superior negativo $-f(x_t, y_t)$ con el tiempo en un problema de diseño de una red de 3 nodos para 10 inicializaciones aleatorias de las variables de nivel superior. Las líneas sólidas representan el valor medio de $-f(x_t, y_{g,t}^{T_g})$ de las 10 realizaciones, y la región sombreada es la $\pm$ desviación estándar. Las líneas discontinuas representan el valor medio de $-f(x_t, y_{F,t}^{T_F})$ de las 10 realizaciones, y la región sombreada es la $\pm$ desviación estándar. Se presentan los resultados para 3 valores diferentes de γ (rojo, púrpura, azul) y el tamaño de paso se fija a $\eta = 1.6 \times 10^{-4}$. La línea de color naranja representa la evolución de $-f(x_t, y_t)$ para el algoritmo LV-HBA.

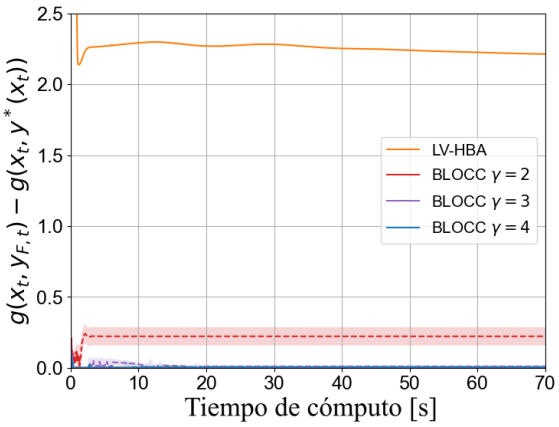


Figura 4.5: Pérdida de optimalidad en el problema de optimización de nivel inferior con el tiempo en un problema de diseño de una red de 3 nodos, $g(x_t, y_t) - g(x_t, y_t^*)$, para 10 inicializaciones aleatorias de las variables de nivel superior. Las líneas sólidas representan el valor medio de las 10 realizaciones, las líneas discontinuas representan el rendimiento de $g(x_t, y_{F,t}^{T_F}) - g(x_t, y_t^*)$ y la región sombreada es la $\pm$ desviación estándar. Se presentan resultados para 3 valores diferentes de γ (rojo, púrpura, azul) y el tamaño de paso se fija a $\eta = 1.6 \times 10^{-4}$. El color naranja representa la pérdida de optimalidad en el problema de nivel inferior $g(x_t, y_t) - g(x_t, y_t^*)$ para el algoritmo LV-HBA.

4.2.2. Resultados numéricos para la red de 3 nodos

	Mercado (o,d)					
Parámetro	(1,2)	(1,3)	(2,1)	(2,3)	(3,1)	(3,2)
demanda w^{od}	1	1	1	1	1	1
ingresos m^{od}	2	6	2	1	6	1
tiempo de viaje incumbente t_{ext}^{od}	3	3	3	3	3	3
	Enlace (i,j)					
Parámetro	(1,2)	(1,3)	(2,1)	(2,2)	(3,1)	(3,2)
coste de construcción del enlace c_{ij}	1	10	1	3	10	3
tiempo de viaje del enlace t_{ij}	1	10	1	2	10	2

Tabla 4.1: Valor de los parámetros para el escenario 1 (red de 3 nodos). El valor de ω_t se establece en -0.1.

En esta sección, se resuelve el problema formulado en (4.8) considerando una red con $|\mathcal{S}| = 3$ nodos (estaciones). Además, se asume que el grafo de enlaces potenciales $\mathcal{A}$ es completo, de modo que $|\mathcal{A}| = 6$ capacidades de enlace deben ser determinadas en el nivel superior (3 ejes con 2 direcciones cada uno). Finalmente, como en el resto del capítulo, se asume que existe demanda para todos los mercados. Esto implica que el conjunto $\mathcal{K}$ también es completo, conteniendo todos los posibles (en este caso, 6) pares origen-destino. Los valores de los parámetros clave se enumeran en la Tabla 4.1, con todos los detalles del escenario simulado disponibles en el repositorio de código en línea.

En cuanto a la configuración del algoritmo BLOCC, se establece el tamaño de paso en $\eta = 1.6 \times 10^{-4}$ y se analiza la convergencia del algoritmo para tres valores diferentes de γ, concretamente $\gamma \in \{2, 3, 4\}$. Se representa el negativo de los valores objetivo del nivel superior computándose tanto con $y_{g,t}$ como con $y_{F,t}$, es decir, $-f(x_t, y_{g,t}^{T_g})$ y $-f(x_t, y_{F,t}^{T_F})$. Los resultados se representan en las figuras 4.4 y 4.5.

La figura 4.4 muestra el comportamiento del algoritmo BLOCC. Específicamente, el eje horizontal representa el tiempo y el eje vertical representa la evolución de la función objetivo del nivel superior. Se aprecian siete líneas. La línea naranja representa la evolución de $-f(x_t, y_t)$ para el algoritmo LV-HBA. Las otras seis líneas son implementaciones del algoritmo BLOCC. Cada color representa un valor diferente de γ; las líneas sólidas representan $-f(x_t, y_{g,t}^{T_g})$ y las líneas discontinuas representan $-f(x_t, y_{F,t}^{T_F})$. Cada simulación se repite 10 veces (utilizando 10 diferentes inicializaciones aleatorias de las variables de nivel superior), y se muestran tanto los valores medios como las desviaciones estándar. Los resultados revelan que todas las versiones del algoritmo BLOCC convergen en menos de 10 segundos, mientras que LV-HBA muestra ligeras fluctuaciones incluso después de funcionar más de 50 segundos. Esto puede deberse a que el algoritmo LV-HBA requiere una proyección conjunta en $\{\mathcal{X} \times \mathcal{Y} : g^c(x, y) \leq 0\}$ en cada iteración, que involucra 42 variables y 6 CCs.

La figura 4.5 muestra la evolución de la brecha de optimalidad de nivel inferior, es decir, el valor de $g(x_t, y_t) - g(x_t, y^*(x_t))$ para los cuatro esquemas probados en este experimento (LV-HBA junto con BLOCC con 3 valores diferentes de γ). En cuanto a LV-HBA, se observa que la optimalidad de nivel inferior no se alcanza dentro de los 70 segundos de tiempo de ejecución. Por el lado de BLOCC, la optimalidad de nivel inferior se logra por construcción cuando $y_t = y_{g,T}^{T_g}$ (cf. ecuación (3.11)), de modo que la brecha en estos 3 casos es cero. Igualmente importante, cuando $y_t = y_{F,T}^{T_F}$, se observa que i) la optimalidad de nivel inferior se logra para $\gamma \geq 3$, y ii) para $\gamma = 2$ existe una brecha de optimalidad mayor a 0, pero es un orden de magnitud menor que para LV-HBA.

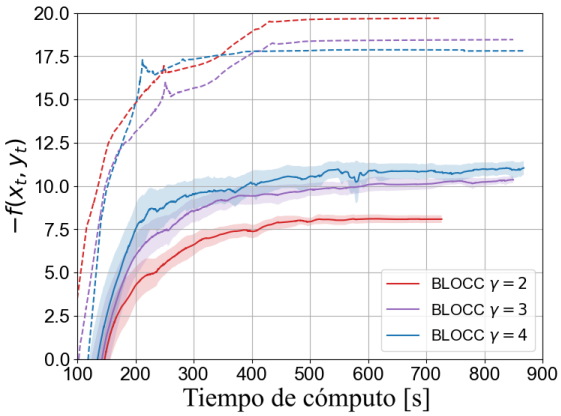


Figura 4.6: Evolución del objetivo de nivel superior negativo $-f(x_t, y_t)$ con el tiempo para un problema de diseño de red de 9 nodos para 10 inicializaciones aleatorias de las variables de nivel superior. Las líneas sólidas representan el valor medio de $-f(x_t, y_{g,t}^{T_g})$ de las 10 realizaciones, y la región sombreada es la $\pm$ desviación estándar. Las líneas discontinuas representan el valor medio de $-f(x_t, y_{F,t}^{T_F})$ de las 10 realizaciones, y la región sombreada es la $\pm$ desviación estándar. Tres valores diferentes de γ (rojo, púrpura, azul) están representados en el algoritmo, y tamaño de paso fijo $\eta = 1.6 \times 10^{-4}$.

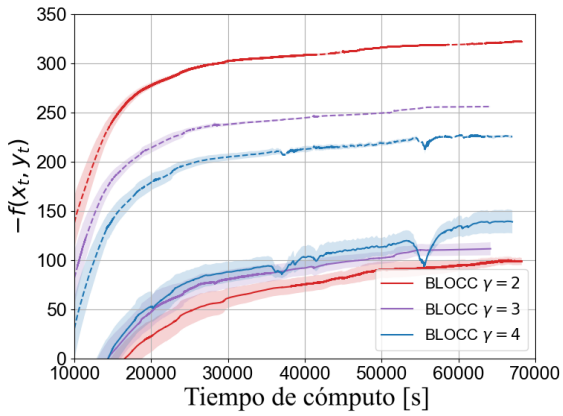


Figura 4.7: Evolución del objetivo de nivel superior negativo $-f(x_t, y_t)$ con el tiempo para un problema de diseño de red de metro en la ciudad de Sevilla, España, para 2 inicializaciones aleatorias diferentes de las variables de nivel superior. Las líneas sólidas representan el valor medio de $-f(x_t, y_{g,t}^{T_g})$ de las 2 realizaciones, y la región sombreada es la $\pm$ desviación estándar. Las líneas discontinuas representan el valor medio de $-f(x_t, y_{F,t}^{T_F})$ de las 2 realizaciones, y la región sombreada es la $\pm$ desviación estándar. Tres valores diferentes de γ (rojo, púrpura, azul) están representados en el algoritmo, y tamaño de paso fijo $\eta = 1.6 \times 10^{-4}$.

4.2.3. Resultados numéricos para la red de 9 nodos

En este caso, se considera la red en [114], véase también la figura 4.3, que tiene $|\mathcal{S}| = 9$ nodos y $|\mathcal{A}| = 30$ enlaces potenciales. Como antes, se considera que todos los mercados existen, de modo que $|\mathcal{K}| = 9 \cdot 8 = 72$. Los parámetros restantes se describen en la figura 4.3 y en el repositorio de código.

La figura 4.6 es la análoga de la figura 4.4 para el escenario de 9 nodos, mostrando el comportamiento del algoritmo BLOCC para $\eta = 1.6 \times 10^{-4}$ y $\gamma \in \{2, 3, 4\}$. Cada simulación se repite 10 veces (utilizando 10 diferentes inicializaciones aleatorias de las variables de nivel superior) y se muestran tanto los valores medios como las desviaciones estándar. Se observa que, dado que el número de variables y restricciones es casi un orden de magnitud mayor, el algoritmo requiere más tiempo para converger. Sin embargo, se observa que la convergencia del algoritmo BLOCC ocurre en un tiempo razonable (20-40 veces más que en el caso de prueba anterior de 3 nodos). En este escenario no se representan los resultados obtenidos con el algoritmo LV-HBA, puesto que el tiempo de cómputo necesario aumenta considerablemente al tener que realizar un paso de proyección sobre restricciones no triviales en cada iteración para un gran número de variables. En cuanto al valor óptimo, se observa que: i) la sensibilidad del valor óptimo (en estado estacionario) con respecto a γ no es demasiado grande; ii) las soluciones basadas en $y_{F,t}^{T_F}$ arrojan mejores valores de objetivo de nivel superior que las basadas en $y_{g,t}^{T_g}$; y iii) la brecha entre $f(x_T, y_{g,t}^{T_g})$ y $f(x_T, y_{F,t}^{T_F})$ disminuye a medida que γ aumenta. La observación ii) se debe al hecho de que $y_{F,t}^{T_F}$ no es factible (lo que significa que viola la optimalidad del nivel inferior), por lo que puede lograr un mejor objetivo de nivel superior. Además, el comportamiento observado en iii) es consistente con la discusión en la sección 3.2.1, con el valor de γ teniendo un impacto en la suboptimalidad de $y_{F,t}^{T_F}$ en el nivel inferior. Específicamente, valores más altos de γ empujan $y_{F,t}^{T_F}$ más cerca de $y_{g,t}^{T_g}$ y, por lo tanto, disminuyen la brecha de optimalidad del nivel inferior. Finalmente, se debe notar que para las líneas sólidas (asociadas con $y_{g,t}^{T_g}$), se obtiene que $f(x_t, y_{g,t}^{T_g}) = f(x_t, y^*(x_t))$. Esto implica que, si se necesita una solución que sea estrictamente óptima en el nivel inferior, los mayores valores de la función objetivo están asociados con valores más altos de γ.

4.2.4. Resultados numéricos para la red de Sevilla

En esta sección, el objetivo es demostrar el impacto práctico de BLOCC aplicándolo a un problema de diseño de red de transporte real. Específicamente, se aborda el diseño de una red de metro en la ciudad de Sevilla, que tiene aproximadamente un millón de habitantes, siendo el sistema de autobuses la red competidora de la red a construir. Los datos sobre la demanda, el número de estaciones y las ubicaciones se han tomado de [114]. La información sobre los costes de construcción, la capacidad y el tiempo de viaje se ha obtenido de operadores de tránsito rápido en España (véase referencias en [109, 113] para obtener detalles completos). Las autoridades de la ciudad consideraron $|\mathcal{S}| = 24$ ubicaciones para estaciones (potenciales). En cuanto a los enlaces, se hacen las siguientes suposiciones:

1. El enlace $(i, j) \in \mathcal{S} \times \mathcal{S}$ solo se considera como enlace potencial a construir si el nodo j es uno de los tres vecinos más cercanos a i, o viceversa. La distancia en este contexto se mide en términos de tiempo de viaje t_{ij}. Esta suposición limita el número de enlaces en cualquier estación a ser como máximo 3, lo cual no restringe en exceso la red si se tiene en cuenta que se tienen 24 estaciones.

2. Además, el enlace $(i, j) \in \mathcal{S} \times \mathcal{S}$ solo se considera como enlace potencial a construir si el tiempo de viaje t_{ij} es menor de 7 minutos. Esta también es una restricción que no restringe en exceso la formulación, ya el valor de 7 minutos se ha elegido de tal forma que permita que todas las estaciones estén conectadas con el resto de la red, incluidas aquellas que están más alejadas del centro de la ciudad (específicamente, el aeropuerto y el campus universitario [114], nodos 6 y 4).

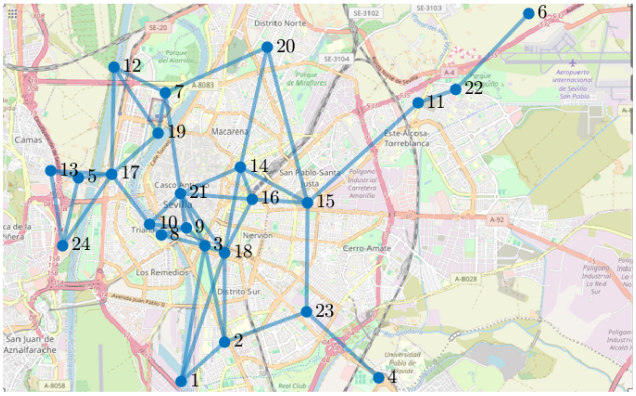


Figura 4.8: Topología de la red de Sevilla.

Bajo estas dos condiciones, el conjunto $\mathcal{A}$ de enlaces potenciales contiene $|\mathcal{A}| = 88$ enlaces. Los conjuntos de enlaces y estaciones, junto con sus ubicaciones reales, se muestran en la figura 4.8. Finalmente, se consideran todos los posibles mercados entre nodos, por lo que $|\mathcal{K}| = 24 \times 23 = 552$.

Siguiendo la narrativa en las secciones 4.2.2 y 4.2.3, la figura 4.7 presenta la evolución de la función objetivo de nivel superior con el tiempo para tres valores del parámetro $\gamma \in \{2, 3, 4\}$. El valor del tamaño de paso se ha establecido en $\eta = 1.6 \times 10^{-4}$, y se han considerado dos inicializaciones diferentes. En cuanto al comportamiento del algoritmo con respecto al valor de γ y la salida particular elegida ($y^{T_g}_{T,g}$ vs. $y^{T_F}_{T,F}$), los hallazgos son muy similares a los de la figura 4.6. Más concretamente, se encuentran brechas más pequeñas para valores más altos de γ, y, si se debe preservar la factibilidad en el nivel inferior (es decir, la consistencia con el nivel de preferencia del usuario), se logran mejores valores de objetivo para valores más altos de γ.

Sin embargo, la observación más importante está relacionada con el tiempo de ejecución. Específicamente, la figura 4.7 revela que, para este escenario del mundo real, el algoritmo BLOCC converge en 10-20 horas. Aunque esto es más de 1,000 veces mayor que el intervalo de convergencia para la red de 3 nodos, el número de variables y restricciones aquí es 100 veces mayor. Más importante aún, en el contexto del diseño de redes de transporte, los tiempos de optimización de 100 horas son ampliamente

aceptados incluso para formulaciones de nivel único. En resumen, los resultados numéricos demuestran que el algoritmo BLOCC propuesto es capaz de resolver problemas con un gran número de variables y CCs, lo que puede tener un valor práctico en aplicaciones del mundo real, como el estudiado en esta sección.

Capítulo 5

Conclusiones y líneas futuras

La memoria ha descrito y sintetizado varias las actividades clave asociadas a la ejecución de este TFG. Inicialmente, se ha llevado a cabo una revisión exhaustiva del campo de la optimización en general, introduciendo los conceptos básicos de un problema de optimización, así como la descripción de los métodos más comunes para su resolución. Posteriormente, se ha presentado la BLO, proporcionando una revisión del estado del arte que ha dado un contexto amplio sobre su evolución y aplicaciones. Una vez que se han asentado las bases, se ha abordado la resolución de problemas BLO con CCs mediante el diseño del algoritmo BLOCC. Este algoritmo, basado en una reformulación penalizada, ha demostrado ser eficaz en dos aplicaciones prácticas: la optimización de hiperparámetros y el diseño de redes de transporte. La validación teórica y experimental del algoritmo han subrayado su relevancia y aplicabilidad, posicionándolo como una herramienta robusta en el campo de la BLO.

Este capítulo tiene un doble propósito. En primer lugar, en la sección 5.1 se realiza un breve resumen de las tareas y contribuciones realizadas, enfatizando tanto los aspectos más novedosos como los que tienen un mayor impacto. En segundo lugar, en la sección 5.2 se identifican vías de trabajo futuro relevantes que permitirían continuar con la senda iniciada por este TFG.

5.1. Contribuciones, resultados e impacto

Este TFG recoge avances significativos en el campo de la BLO con CCs. A continuación, se presentan las principales contribuciones y un resumen de los resultados obtenidos:

- **Desarrollo del algoritmo BLOCC.** Se ha diseñado el algoritmo BLOCC, el cual reformula un problema de BLO con CCs en un problema de optimización de un único nivel que se resuelve de forma desacoplada para las variables del nivel superior x y las variables del nivel inferior y. Dicha reformulación se ha realizado utilizando funciones de penalización y utilizando métodos primales-duales. Una de las características innovadoras de BLOCC es que evita el cálculo de proyecciones, mejorando significativamente la eficiencia computacional en problemas con un gran número de variables y CCs. Además, este algoritmo proporciona soluciones óptimas en el nivel inferior en cada iteración, lo cual es crucial para un amplio rango de aplicaciones prácticas.

- **Caracterización teórica.** Se ha demostrado la convergencia del algoritmo bajo hipótesis de convexidad en el problema del nivel inferior y suavidad de Lipschitz en ambos niveles. Esta caracterización se recoge en el teorema 2, mediante el que se asegura una solución ϵ-aproximada del problema BLO original. Además, se han establecido garantías de convergencia en tiempo finito –véase el teorema 3–, lo que refuerza la robustez del algoritmo y lo hace especialmente adecuado para escenarios con capacidad computacional limitada.

- **Aplicaciones prácticas: impacto y resultados.** El algoritmo BLOCC se ha aplicado exitosamente en dos problemas reales de ámbitos distintos: la optimización de hiperparámetros y el

diseño de redes de transporte. Por un lado, en el ámbito de la optimización de hiperparámetros, el algoritmo se ha aplicado exitosamente en la selección de hiperparámetros para SVMs. En cuanto al diseño de redes de transporte, el algoritmo BLOCC se ha utilizado para abordar el diseño de una red de transporte mediante la resolución de un problema BLO donde, en el nivel superior, se optimizan los beneficios del operador, mientras que en el nivel inferior los usuarios toman decisiones de acuerdo a sus preferencias de viaje. Esta aplicación es particularmente relevante debido a la complejidad y el tamaño del problema que se resuelve. Además, el algoritmo BLOCC ha logrado mejorar significativamente los resultados obtenidos aplicando otros algoritmos de la literatura. El algoritmo BLOCC ha demostrado ser una herramienta eficiente para resolver problemas BLO con CCs. Los experimentos realizados no solo han validado el algoritmo en términos de eficiencia computacional, sino que también han demostrado su robustez y capacidad para proporcionar soluciones óptimas.

- **Desarrollo académico y profesional.** Finalmente, y desviándome un poco del tema principal, quiero resaltar el impacto significativo que los cuatro meses dedicados a este TFG han tenido en mi desarrollo académico, profesional y personal. Durante este tiempo, he podido integrar de manera natural los conocimientos adquiridos en diversas asignaturas a lo largo de la carrera, tales como Procesamiento Digital de la Información, Álgebra, Cálculo y Programación, aplicando de forma práctica muchas de las herramientas y tecnologías aprendidas en ellas. Además, el TFG me ha permitido adquirir tanto nuevos conocimientos y habilidades, muchas de ellas relacionadas con el ámbito de la investigación. He tenido la oportunidad de colaborar con grupos internacionales de prestigio y desarrollar documentos científico-técnicos, mejorando durante el proceso tanto mi capacidad para formalizar y trasmitir mi ideas a través del lenguaje matemático como mis habilidades de trabajo en grupo. En lo que a adquisición de conocimiento se refiere, he profundizado en áreas como la optimización, la algoritmia, el aprendizaje automático y las redes de transporte, explorando técnicas avanzadas y algoritmos complejos aplicados a problemas reales. Este proceso ha reforzado mi formación y me ha hecho más consciente de las múltiples aplicaciones de los conocimientos adquiridos durante la carrera. Desde un punto de vista más transversal, además de desarrollar habilidades comunicativas y colaborativas en grupos internacionales, también he aprendido a utilizar nuevas herramientas de desarrollo software, abarcando desde el uso de servidores de simulación hasta la computación distribuida. En conjunto, creo que todas estas experiencias han contribuido significativamente a mi crecimiento académico y profesional y las considero como un resultado más de este TFG.

5.2. Líneas de trabajo futuro

Esta sección pretende identificar, de forma no exhaustiva, algunas de las potenciales líneas de trabajo futuro que han surgido tras la realización de este TFG. Si bien el interés, plazo de ejecución o nivel de detalle de las líneas identificadas no es homogéneo, se intenta proporcionar un listado representativo.

Contribuciones teóricas

- Diseño de otros tipos de algoritmos para BLO con CCs. Actualmente, el algoritmo BLOCC desarrollado en este TFG se basa en ciertas suposiciones como la convexidad y suavidad de Lipschitz de las funciones involucradas. Sería valioso explorar algoritmos que puedan manejar funciones que no cumplan estas condiciones. Para abordar escenarios de optimización con este tipo de funciones se podrían investigar técnicas de optimización no suave, como métodos de subgradiente o métodos proximales, que permiten trabajar con funciones no diferenciables.
- Obtención de nuevos resultados teóricos. Sería beneficioso obtener resultados de convergencia más fuertes cuando el problema tiene más estructura, como en el caso de CCs lineales o escenarios en

los que las variables del nivel superior x no aparecen en el objetivo del nivel inferior. En estos contextos, las propiedades adicionales de las funciones y restricciones podrían ser explotadas para reducir el problema a una estructura más sencilla que dé lugar a mejoras en las tasas de convergencia del algoritmo.

Contribuciones de aplicación

- Como se ha analizado en esta memoria de TFG, la optimización de hiperparámetros es una aplicación relevante de BLO en aprendizaje automático. Otra área dentro del aprendizaje automático donde BLOCC puede generar resultados interesantes en en problemas de *metalearning* con restricciones. Esto es particularmente relevante, por ejemplo, en entornos distribuidos donde algunas de las restricciones están asociadas a limitaciones físicas de la red de agentes o del canal de comunicación. En este contexto, los algoritmos BLOCC podrían ser utilizados para optimizar configuraciones de modelos que deben cumplir con restricciones específicas en diferentes nodos de una red distribuida, mejorando así la eficiencia y rendimiento global del sistema.
- Resulta igualmente interesante aplicar BLO a ámbitos donde hasta ahora casi no hay resultados. Uno de ellos es el aprendizaje automático sobre grafos. En problemas de este ámbito es común aprender primero el grafo y luego utilizarlo para resolver un problema de inferencia (clasificación o regresión) específico. Una formulación BLO en la que en el nivel superior se estima el grafo y en el nivel inferior se optimizan los parámetros de la arquitectura relacionada con el problema de clasificación o regresión resulta, por lo tanto, de interés. La existencia de restricciones es común tanto en métodos de aprendizaje de grafos (donde se establecen relaciones entre los ejes y determinadas métricas de correlación y asociación) como en las propias arquitecturas de clasificación o regresión, motivando el uso del algoritmos BLOCC. La contribución en esta línea sería doble. En primera instancia, se formalizaría el problema en el contexto de una BLO con restricciones y se aplicaría el algoritmo BLOCC para aprender el grafo óptimo, así como los parámetros de la arquitectura. En segunda instancia, podrían diseñarse variaciones del algoritmo BLOCC que tuvieran en cuenta las características específicas del problema, especialmente el hecho de que, pese a representarse habitualmente por matrices, los grafos tienen una naturaleza discreta. Esta formulación BLO proporcionaría grafos que estén mejor adaptados a las tareas específicas para las que serán utilizados, mejorando así la eficacia y eficiencia de los algoritmos de aprendizaje automático sobre grafos.
- En el ámbito de las redes de transporte, resulta interesante incorporar a la formulación la congestión, que es un factor adicional cuyo impacto en la decisión de viaje de los usuarios ha sido puesto de manifiesto en la literatura. La formulación BLO permite modelar este fenómeno de forma natural. En el problema del nivel superior, el operador construye la red, mientras que en el nivel inferior, los usuarios deciden sus rutas de viaje. Así, es posible introducir términos que representen el tiempo de viaje por un enlace en función de la congestión. La congestión se mide mediante la tasa de ocupación del enlace. Cuando la ocupación alcanza la capacidad diseñada, el tiempo de viaje asociado al enlace aumenta exponencialmente. Este modelo de congestión carece de convexidad conjunta en las variables de los niveles superior x e inferior y, por lo que problemas que incluyen este modelo se han resuelto tradicionalmente mediante algoritmos heurísticos sin fuertes garantías de optimalidad. Estos trabajos previos se enfocan en diseñar la red de transporte y modelar el comportamiento de los usuarios simultáneamente. Sin embargo, este modelo de congestión es convexo en las variables del nivel inferior y para una red de transporte dada (valores fijos de x). Explotando la estructura del algoritmo BLOCC, es posible resolver problemas de diseño de redes de transporte que incluyan este modelo de congestión sin recurrir a técnicas heurísticas.
- En el contexto del grado en el que se realiza este TFG, también resulta interesante buscar aplicaciones de BLO en el ámbito de la ingeniería de telecomunicaciones más allá del aprendizaje

automático. Un ejemplo particularmente relevante sería el diseño de redes de comunicaciones inalámbricas. En este caso, un problema particularmente relevante sería una BLO en la que i) en el nivel superior se decide la posición de las estaciones base y las potencias y frecuencias asignadasa a cada una de ellas y ii) en el nivel inferior se asignan los recursos disponibles a los usuarios servidos por cada estación base. Otra área relevante para la aplicación de BLO en comunicaciones es el uso de radios cognitivas que permiten una gestión dinámica del espectro de frecuencias. En un escenario con radios cognitivas, se distinguen dos tipos de usuarios: primarios y secundarios. Los usuarios primarios tienen derechos exclusivos sobre ciertas bandas de frecuencia, mientras que los usuarios secundarios pueden acceder a estas bandas de manera oportunista, siempre y cuando no interfieran con los usuarios primarios.

En este contexto, el problema BLO podría formularse de la siguiente manera: en el nivel superior, se optimiza la asignación de frecuencias y la gestión del espectro para los usuarios primarios, asegurando su calidad de servicio. En el nivel inferior, los usuarios secundarios buscan oportunidades para utilizar el espectro de manera eficiente, sin interferir con los usuarios primarios. Este enfoque binivel permite una coexistencia armoniosa y eficiente entre ambos tipos de usuarios, optimizando el uso del espectro de frecuencias y mejorando la capacidad total de la red de comunicaciones.

La implementación de modelos BLO en redes de comunicaciones sería un desafío particularmente significativo debido a la necesidad de considerar las variaciones aleatorias, tanto en temporales como espaciales, de los canales de comunicaciones, así como las restricciones impuestas por la necesidad de proteger la transmisión de los usuarios primarios en caso de las radios cognitivas. Sin embargo, la aplicación de técnicas de BLO en este ámbito tiene el potencial de mejorar sustancialmente el rendimiento y la eficiencia de las redes de comunicaciones modernas.

Otras líneas de interés podrían enfocarse en abordar problemas de optimización con múltiples niveles, considerar versiones estocásticas o diversos horizontes temporales. Además, se podrían estudiar problemas de BLO en el contexto del control y el aprendizaje por refuerzo. Asimismo, sería beneficioso identificar otros problemas en campos como la ingeniería, la estadística o incluso las ciencias sociales, que puedan beneficiarse de la estructura de las formulaciones BLO.

Bibliografía

[1] H. v. Stackelberg, "Theory of the market economy," 1952.

[2] J. Bracken and J. T. McGill, "Mathematical Programs with Optimization Problems in the Constraints," *Operations Research*, vol. 21, pp. 37–44, February 1973.

[3] D. Aksen and N. Aras, "A bilevel fixed charge location model for facilities under imminent attack," *Computers & OR*, vol. 39, pp. 1364–1381, 07 2012.

[4] B. An, F. Ordóñez, M. Tambe, E. Shieh, R. Yang, C. Baldwin, J. DiRenzo, K. Moretti, B. Maule, and G. Meyer, "A deployed quantal response-based patrol planning system for the u.s. coast guard," *Interfaces*, vol. 43, no. 5, pp. 400–420, 2013.

[5] G. Brown, M. Carlyle, J. Salmerón, and R. Wood, "Defending critical infrastructure," *Interfaces*, vol. 36, pp. 530–544, 12 2006.

[6] M. Bostian, G. Whittaker, A. Sinha, and B. Barnhart, "Incorporating data envelopment analysis solution methods into bilevel multi-objective optimization," pp. 1667–1674, 05 2015.

[7] M. A. Amouzegar and K. Moshirvaziri, "Determining optimal pollution control policies: An application of bilevel programming," *Eur. J. Oper. Res.*, vol. 119, pp. 100–120, 1999.

[8] G. Whittaker, R. Färe, S. Grosskopf, B. Barnhart, M. Bostian, G. Mueller-Warrant, and S. Griffith, "Spatial targeting of agri-environmental policy using bilevel evolutionary optimization," *Omega*, vol. 66, pp. 15–27, 2017.

[9] M. P. Bendsøe, "Optimization of structural topology, shape, and material," 1995.

[10] S. Christiansen, M. Patriksson, and L. Wynter, "Stochastic bilevel programming in structural optimization," *Structural and multidisciplinary optimization*, vol. 21, pp. 361–371, 2001.

[11] J. Herskovits, A. Leontiev, G. Dias, and G. Santos, "Contact shape optimization: A bilevel programming approach," *Structural and Multidisciplinary Optimization*, vol. 20, pp. 214–221, 11 2000.

[12] P. W. Weixi Chen and H. Dong, "Surrogate-based bilevel shape optimization for blended-wing–body underwater gliders," *Engineering Optimization*, vol. 55, no. 6, pp. 998–1019, 2023.

[13] E. Aiyoshi and K. Shimizu, "Hierarchical decentralized systems and its new solution by a barrier method.," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-11, no. 6, pp. 444–449, 1981.

[14] V. Suryan, A. Sinha, P. Malo, and K. Deb, "Handling inverse optimal control problems using evolutionary bilevel optimization," *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1893–1900, 2016.

[15] S.-W. Chiou, "Bilevel programming for the continuous transport network design problem," *Transportation Research Part B: Methodological*, vol. 39, no. 4, pp. 361–383, 2005.

[16] B. Yu, L. Kong, Y. Sun, B. Yao, and Z. Gao, "A bi-level programming for bus lane network design," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 310–327, 2015.

[17] A. H. Barahimi, A. Eydi, and A. Aghaie, "Multi-modal urban transit network design considering reliability: multi-objective bi-level optimization," *Reliability Engineering & System Safety*, vol. 216, p. 107922, 2021.

[18] J. Ye, Y. Jiang, J. Chen, Z. Liu, and R. Guo, "Joint optimisation of transfer location and capacity for a capacitated multimodal transport network with elastic demand: a bi-level programming model and paradoxes," *Transportation Research Part E: Logistics and Transportation Review*, vol. 156, p. 102540, 2021.

[19] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard, "A bilevel model for toll optimization on a multicommodity transportation network," *Transportation Science*, vol. 35, no. 4, pp. 345–358, 2001.

[20] J. Y. Wang, M. Ehrgott, K. N. Dirks, and A. Gupta, "A bilevel multi-objective road pricing model for economic, environmental and health sustainability," *Transportation Research Procedia*, vol. 3, pp. 393–402, 2014. 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain.

[21] M. Florian and Y. Chen, "A coordinate descent method for the bi-level o-d matrix adjustment problem," *International Transactions in Operational Research*, vol. 2, pp. 165 – 179, 08 2006.

[22] A. K. Shubham N. Patil, Krishna N.S. Behara and A. Bhaskar, "Methods to enhance the quality of bi-level origin–destination matrix adjustment process," *Transportation Letters*, vol. 15, no. 2, pp. 77–86, 2023.

[23] K. Bennett, G. Kunapuli, J. Hu, and J.-S. Pang, "Bilevel optimization and machine learning," pp. 25–47, 06 2008.

[24] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *International Conference on Machine Learning*, pp. 1165–1173, 2017.

[25] N. Couellan and W. Wang, "Bi-level stochastic gradient for large scale support vector machine," *Neurocomputing*, vol. 153, pp. 300–308, 2015.

[26] Q. Li, Z. Li, and A. Zemkoho, "Bilevel hyperparameter optimization for support vector classification: theoretical analysis and a solution method," *Mathematical Methods of Operations Research*, vol. 96, no. 3, pp. 315–350, 2022.

[27] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 115–123, PMLR, 17–19 Jun 2013.

[28] Z. Yang, Y. Chen, M. Hong, and Z. Wang, "On the global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost," 2019.

[29] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, "A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic," 2022.

[30] L. Jiang, Q. Xiao, V. M. Tenorio, F. Real-Rojas, A. G. Marques, and T. Chen, "A primal-dual-assisted penalty approach to bilevel optimization with coupled constraints," 2024. Enviado a Conference on Neural Information Processing Systems (NeurIPS), Dec. 2024.

[31] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[32] D. Bertsekas, *Nonlinear Programming*. Athena scientific optimization and computation series, Athena Scientific, 2016.

[33] A. Nemirovski, "Convex analysis, nonlinear programming theory, nonlinear programming algorithms," *Lecture notes. Georgia Institute of Technology*, 2023.

[34] D. Bertsekas, *Convex Optimization Algorithms*. Athena Scientific, 2015.

[35] R. M. Freund, "Penalty and barrier methods for constrained optimization," *Lecture notes. Massachusetts Institute of Technology*, 2004.

[36] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated, 2015.

[37] X. Deng, "Complexity issues in bilevel linear programming," 1998.

[38] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2018.

[39] J. F. Bard and J. E. Falk, "An explicit solution to the multi-level programming problem," *Computers & Operations Research*, vol. 9, no. 1, pp. 77–100, 1982.

[40] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," vol. 32, pp. 783–792, 1981.

[41] J. Bard and J. Moore, "A branch and bound algorithm for the bilevel programming problem," vol. 11, pp. 281–292, 1990.

[42] F. Pedregosa, "Hyperparameter optimization with approximate gradient," in *International conference on machine learning*, pp. 737–746, 2016.

[43] S. Ghadimi and M. Wang, "Approximation methods for bilevel programming," *arXiv preprint arXiv:1802.02246*, 2018.

[44] C. Kolstad and L. Lasdon, "Derivative evaluation and computational experience with large bilevel mathematical programs," vol. 65, pp. 485–499, 1990.

[45] S. Xu and M. Zhu, "Efficient gradient approximation method for constrained bilevel optimization," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23, AAAI Press, 2023.

[46] E. Aiyoshi and K. Shimizu, "A solution method for the static constrained stackelberg problem via penalty method," *IEEE Transactions on Automatic Control*, vol. 29, no. 12, pp. 1111–1114, 1984.

[47] Y. Ishizuka and E. Aiyoshi, "Double penalty method for bilevel optimization problems," *Annals of Operations Research*, vol. 34, pp. 73–88, Dec. 1992.

[48] A. Sinha, P. Malo, A. Frantsev, and K. Deb, "Finding optimal strategies in a multi-period multi-leader–follower stackelberg game using an evolutionary algorithm," *Computers & Operations Research*, vol. 41, pp. 374–385, 2014.

[49] R. Mathieu, L. Pittard, and G. Anandalingam, "Genetic algorithm based approach to bi-level linear programming," *RAIRO. Recherche Opérationnelle*, vol. 28, 01 1994.

[50] G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *Journal of Mechanical Design - J MECH DESIGN*, vol. 129, 04 2007.

[51] A. Sinha, P. Malo, and K. Deb, "Efficient evolutionary algorithm for single-objective bilevel optimization," *ArXiv*, vol. abs/1303.3901, 2013.

[52] A. Sinha, P. Malo, and K. Deb, "An improved bilevel evolutionary algorithm based on quadratic approximations," in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 1870–1877, IEEE, 2014.

[53] A. Sinha, P. Malo, and K. Deb, "Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping," *Eur. J. Oper. Res.*, vol. 257, pp. 395–411, 2017.

[54] J. J. Ye and D. Zhu, "New necessary optimality conditions for bilevel programs by combining the mpec and value function approaches," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1885–1905, 2010.

[55] H. Von Stackelberg, *Market structure and equilibrium*. Springer Science & Business Media, 2010.

[56] W. Candler and R. Norton, "Multilevel programming," Tech. Rep. 20, 1977.

[57] W. Bialas and M. Karwan, "Multilevel linear programming," Tech. Rep. 78-1, 1978.

[58] J. Bard, "Convex two-level optimization," vol. 40, pp. 15–27, 1988.

[59] T. Edmunds and J. Bard, "Algorithms for nonlinear bilevel mathematical programming," vol. 21, pp. 83–89, 1991.

[60] P. Hansen, B. Jaumard, and G. Savard, "New branch-and-bound rules for linear bilevel programming," vol. 13, pp. 1194–1217, 1992.

[61] J. J. Ye and D. L. Zhu, "Optimality conditions for bilevel programming problems," *Optimization*, vol. 33, no. 1, pp. 9–27, 1995.

[62] S. Dempe, "A necessary and a sufficient optimality condition for bilevel programming problems," *Optimization*, vol. 25, pp. 341–354, 1992.

[63] S. Dempe, "Optimality conditions for bilevel programming problems," in *System Modelling and Optimization* (L. D. Davisson, A. G. J. MacFarlane, H. Kwakernaak, J. L. Massey, Y. Z. Tsypkin, A. J. Viterbi, and P. Kall, eds.), (Berlin, Heidelberg), pp. 17–24, Springer Berlin Heidelberg, 1992.

[64] A. W. Yezza, "First-order necessary optimality conditions for general bilevel programming problems," *Journal of Optimization Theory and Applications*, vol. 89, pp. 189–219, 1996.

[65] J. V. Outrata, "Necessary optimality conditions for stackelberg problems," *Journal of optimization theory and applications*, vol. 76, pp. 305–320, 1993.

[66] G. Savard and J. Gauvin, "The steepest descent direction for the nonlinear bilevel programming problem," *Oper. Res. Lett.*, vol. 15, pp. 265–272, 1990.

[67] K. Ji, J. Yang, and Y. Liang, "Provably faster algorithms for bilevel optimization and applications to meta-learning," 2020.

[68] T. Chen, Y. Sun, Q. Xiao, and W. Yin, "A single-timescale method for stochastic bilevel optimization," in *International Conference on Artificial Intelligence and Statistics*, (virtual), 2022.

[69] T. Chen, Y. Sun, and W. Yin, "Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems," in *Advances in Neural Information Processing Systems*, (Virtual), 2021.

[70] P. Khanduri, I. Tsaknakis, Y. Zhang, J. Liu, S. Liu, J. Zhang, and M. Hong, "Linearly constrained bilevel optimization: A smoothed implicit gradient approach," in *International Conference on Machine Learning*, pp. 16291–16325, 2023.

[71] J. Kwon, D. Kwon, S. Wright, and R. Nowak, "On penalty methods for nonconvex bilevel optimization and first-order stochastic approximation," 2024.

[72] H. Shen, Q. Xiao, and T. Chen, "On penalty-based bilevel gradient descent method," 2023.

[73] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, "A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic," *SIAM Journal on Optimization*, vol. 33, no. 1, 2023.

[74] P. Khanduri, S. Zeng, M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, "A near-optimal algorithm for stochastic bilevel optimization via double-momentum," in *Advances in Neural Information Processing Systems*, (Virtual), 2021.

[75] H. Shen and T. Chen, "A single-timescale analysis for stochastic approximation with multiple coupled sequences," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17415–17429, 2022.

[76] J. Li, B. Gu, and H. Huang, "A fully single loop algorithm for bilevel optimization without hessian inverse," in *Association for the Advancement of Artificial Intelligence*, (virtual), pp. 7426–7434, 2022.

[77] D. Sow, K. Ji, and Y. Liang, "On the convergence theory for hessian-free bilevel algorithms," vol. 35, pp. 4136–4149, 2022.

[78] H. Yang, L. Luo, C. J. Li, M. Jordan, and M. Fazel, "Accelerating inexact hypergradient descent for bilevel optimization," in *OPT 2023: Optimization for Machine Learning*, 2023.

[79] K. Ji, J. Yang, and Y. Liang, "Bilevel optimization: Convergence analysis and enhanced design," in *International Conference on Machine Learning*, 2021.

[80] J. J. Ye, D. Zhu, and Q. J. Zhu, "Exact penalization and necessary optimality conditions for generalized bilevel programming problems," *SIAM Journal on optimization*, vol. 7, no. 2, pp. 481–507, 1997.

[81] R. Liu, X. Liu, X. Yuan, S. Zeng, and J. Zhang, "A value-function-based interior-point method for non-convex bi-level optimization," in *International conference on machine learning*, pp. 6882–6892, 2021.

[82] A. Mehra and J. Hamm, "Penalty method for inversion-free deep bilevel optimization," in *Asian conference on machine learning*, pp. 347–362, 2021.

[83] B. Liu, M. Ye, S. Wright, P. Stone, and Q. Liu, "Bome! bilevel optimization made easy: A simple first-order approach," *Advances in neural information processing systems*, vol. 35, pp. 17248–17262, 2022.

[84] J. Kwon, D. Kwon, S. Wright, and R. D. Nowak, "A fully first-order method for stochastic bilevel optimization," in *International Conference on Machine Learning*, pp. 18083–18113, 2023.

[85] J. Xu, Y. Tu, and Z. Zeng, "A nonlinear multiobjective bilevel model for minimum cost network flow problem in a large-scale construction project," *Mathematical Problems in Engineering*, vol. 2012, 04 2012.

[86] S. Maldonado-Pinto, M.-S. Casas-Ramírez, and J.-F. Camacho-Vallejo, "Analyzing the performance of a hybrid heuristic for solving a bilevel location problem under different approaches to tackle the lower level," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–10, 06 2016.

[87] L. L. Gao, J. Ye, H. Yin, S. Zeng, and J. Zhang, "Value function based difference-of-convex algorithm for bilevel hyperparameter selection problems," in *International Conference on Machine Learning*, (Baltimore, MD), pp. 7164–7182, 2022.

[88] S. Lu, "Slm: A smoothed first-order lagrangian method for structured constrained nonconvex optimization," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[89] Z. Chen, B. Kailkhura, and Y. Zhou, "A fast and convergent proximal algorithm for regularized nonconvex and nonsmooth bi-level optimization," *arXiv preprint arXiv:2203.16615*, 2022.

[90] R. Liu, X. Liu, S. Zeng, J. Zhang, and Y. Zhang, "Value-function-based sequential minimization for bi-level optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[91] Q. Xiao, H. Shen, W. Yin, and T. Chen, "Alternating implicit projected sgd and its efficient variants for equality-constrained bilevel optimization," in *International Conference on Artificial Intelligence and Statistics*, 2023.

[92] W. Yao, C. Yu, S. Zeng, and J. Zhang, "Constrained bi-level optimization: Proximal lagrangian value function approach and hessian-free algorithm," *arXiv preprint arXiv:2401.16164*, 2024.

[93] W. Yao, H. Yin, S. Zeng, and J. Zhang, "Overcoming lower-level constraints in bilevel optimization: A novel approach with regularized gap functions," *arXiv preprint arXiv:2406.01992*, 2024.

[94] A. Sinha, P. Malo, P. Xu, and K. Deb, "A bilevel optimization approach to automated parameter tuning," *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, 07 2014.

[95] Y. P. Wang, Y. C. Jiao, and H. Li, "An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint - handling scheme," *IEEE Trans. on Systems, Man, and Cybernetics-Part C*, 2005.

[96] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of Operations Research*, vol. 153, pp. 235–256, 2007.

[97] J. F. Bard, *Practical bilevel optimization: algorithms and applications*, vol. 30. Springer Science & Business Media, 2013.

[98] G. Wachsmuth, "On licq and the uniqueness of lagrange multipliers," *Operations Research Letters*, vol. 41, no. 1, pp. 78–80, 2013.

[99] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pp. 795–811, 2016.

[100] A. Ruszczyński, *Nonlinear Optimization*. Princeton University Press, 2006.

[101] J. F. Bonnans and A. Shapiro, *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[102] K. Ito and K. Kunisch, *Lagrange Multiplier Approach to Variational Problems and Applications*, vol. 15 of *Advances in Design and Control*. Not specified in the provided information, 2008. Includes bibliographical references and index.

[103] S. Bubeck, *Convex Optimization: Algorithms and Complexity*. Foundations and Trends® in Machine Learning, 2015.

[104] S. Ghadimi, G. Lan, and H. Zhang, "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization," *Mathematical Programming*, vol. 155, no. 1, pp. 267–305, 2016.

[105] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, p. 273–297, Sept. 1995.

[106] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, 05 2001.

[107] D. Dua and C. Graff, "Uci machine learning repository," 2017. Accessed: 2024-05-21.

[108] T. K. Ho and E. M. Kleinberg, "Building projectable classifiers of arbitrary complexity," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 2, pp. 880–885, 1996.

[109] L. Cadarso and A. Marin, "Combining robustness and recovery in rapid transit network design," *Transportmetrica A: Transport Science*, vol. 12, pp. 1–26, 11 2015.

[110] M. E. Ben-Akiva and S. R. Lerman, *Discrete choice analysis: theory and application to travel demand*, vol. 9. MIT press, 1985.

[111] E. Cascetta, *Transportation systems analysis: models and applications*, vol. 29. Springer Science & Business Media, 2009.

[112] N. Oppenheim, "Equilibrium trip distribution/assignment with variable destination costs," *Transportation Research Part B: Methodological*, vol. 27, no. 3, pp. 207–217, 1993.

[113] F. Real Rojas *et al.*, "Diseño de redes de transporte: integrando la competencia modal con técnicas de optimización convexa," 2024.

[114] L. Escudero and S. Muñoz, "An approach for solving a modification of the extended rapid transit network design problem," *Top*, vol. 17, pp. 320–334, 12 2009.