Facultad de Matemática y Computación

Universidad de La Habana

Tesis de diploma de la especialidad

Ciencia de la Computación

Un generador de problemas prueba para evaluar la calidad de la solución de los algoritmos de problemas de optimización de dos niveles

Autor: Francisco Vicente Suárez Bellón

Tutora: Dr. C. Gemayqzel Bouza Allende

La Habana, 4 de febrero de 2025

RESUMEN

El problema de optimización binivel se define como minimizar una función sobre un conjunto determinado por los puntos óptimos de un modelo de programación matemática. La optimización en el nivel inferior depende de las decisiones tomadas en el nivel superior, creando así una relación de interdependencia entre ambos niveles.

Para abordar este problema, se considera la creación de un problema relacionado en el cual el nivel inferior se sustituye por las condiciones necesarias de optimalidad, siendo este un problema con restricciones de complementariedad.

En este trabajo se propone una forma de generar problemas de dos niveles cuyo problema con restricciones de complementariedad (MPEC) tiene un punto estacionario perteneciente a una de las siguientes clases: fuertemente estacionario, M-estacionario o C-estacionario, dependiendo de los multiplicadores.

Palabras clave: Optimización binivel, MPECs, Condiciones necesarias de optimalidad, Punto estacionario.

ABSTRACT

The bilevel optimization problem is defined as minimizing a function on a set determined by the optimal points of a mathematical programming model. The optimization at the lower level depends on the decisions made at the upper level, thus creating an interdependent relationship between both levels.

To address this problem, the lower-level problem is replaced by the necessary optimality conditions and the resulting (relaxed) optimization problem with complementarity constraints is solved.

This work proposes a way to generate two-level problems whose relaxed problem has a stationary point belonging to one of the following classes: strongly stationary, M-stationary, or C-stationary, depending on the multipliers.

Keywords: Bi-level optimization, MPECs, Necessary optimality conditions, Stationary point.

AGRADECIMIENTOS

El camino recorrido hasta este momento ha sido posible gracias al apoyo y dedicación de personas extraordinarias que han estado presentes en cada paso de mi formación académica. Deseo expresar mi más sincero agradecimiento a todos los profesores que han contribuido a mi desarrollo profesional durante mi trayectoria universitaria.

Mi más profunda gratitud a la profesora Dra. C. Gemayqzel, mi tutora, por su paciencia, dedicación y capacidad para transmitir conocimientos en el fascinante mundo de la optimización binivel. Su apoyo y comprensión han sido fundamentales para la culminación exitosa de este trabajo.

A mi familia, quiero agradecerles especialmente por su infinita paciencia y dedicación en los momentos más difíciles de este proceso. Su apoyo constante y su aliento para no abandonar nunca me han motivado a seguir adelante y alcanzar mis metas.

A mis amigos y compañeros de estudio, gracias por su tiempo, dedicación y palabras de aliento, que enriquecieron esta experiencia académica.

Un agradecimiento especial a mis mascotas, por su compañía y alegría en los momentos desafiantes, y a ChatGPT, cuya asistencia fue invaluable en la redacción y organización de mis ideas.

Este trabajo no habría sido posible sin el soporte y la guía de cada uno de ustedes. Gracias por ser parte fundamental de este capítulo de mi vida.

Índice general

1. INTRODUCCIÓN

La optimización de dos niveles, un área fundamental en la investigación operativa y la teoría de juegos, presenta desafíos significativos debido a su complejidad inherente. Este tipo de problemas se caracterizan por la interacción entre un líder y un seguidor, donde las decisiones del líder afectan las respuestas del seguidor. Uno de los aspectos más críticos de esta problemática es garantizar la existencia de soluciones óptimas, lo cual se ve complicado por la naturaleza no convexa del problema, incluso cuando las funciones y los conjuntos factibles son convexos. A menudo, los algoritmos utilizados en este contexto solo logran identificar puntos estacionarios o críticos, que no necesariamente representan soluciones locales o globales, ver Dempe y Zemkoho, 2020a.

El modelo de dos niveles es:

$$\begin{array}{ll} \min\limits_{x} & F(x,y) \\ s.a & \begin{cases} x \in T \\ y \in S(x) = \arg\min\limits_{y}\{f(x,y) \quad s.a \quad y \in H\} \\ x, y \in M^0 \end{cases} \end{array} \tag{1.0.0}$$

Problema de Optimización Binivel

donde:

$$x \in R^n, \qquad y \in R^m$$

$$F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \qquad T \subseteq \mathbb{R}^n,$$
$$f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \qquad H \subseteq \mathbb{R}^m,$$

$$M^0 \subseteq \mathbb{R}^{n+m}$$

En otras palabras, el problema de optimización binivel se centra en que el líder (nivel superior) debe tomar decisiones (x) que optimicen su objetivo $F(x, y)$, anticipando que el seguidor (nivel inferior) responderá de manera óptima con respecto a su propio objetivo $f(x, y)$, dado el valor de x elegido por el líder. Esta interacción jerárquica entre ambos niveles añade una gran complejidad al problema en comparación con los problemas de optimización de un solo nivel.

Los problemas de optimización de dos niveles son muy utilizados para modelar y analizar mercados eléctricos complejos, ofreciendo una perspectiva única sobre las interacciones estratégicas entre diversos agentes económicos. En el trabajo de Aussel et al., 2016 se desarrolló un modelo innovador que aborda los mercados de electricidad desregulados. Su enfoque se distingue por incorporar restricciones de producción y pérdidas térmicas, lo que permite una modelización más precisa y realista. Mediante el uso de modelos binivel, los investigadores pueden explorar escenarios más complejos y representativos del funcionamiento real de los mercados energéticos. Otro ejemplo en esta línea se encuentra en Aussel et al., 2017, se profundiza en el análisis de mercados de electricidad de pago por oferta, explorando cómo un productor puede ajustar su estrategia considerando las acciones de sus competidores. El estudio destaca la aplicación de conceptos de **equilibrio de Nash** y técnicas de mejor respuesta, proporcionando una metodología sofisticada para optimizar la participación de un productor en el mercado.

Los modelos binivel también tiene aplicaciones fundamentales en la selección de hiperparámetros en aprendizaje automático, como lo demuestra el trabajo de Dempe y Zemkoho, 2020b. El capítulo 6 del libro aborda la optimización de hiperparámetros en problemas de clasificación y regresión, presentando algoritmos innovadores para manejar funciones objetivo no suaves y no convexas. La razón del uso de esta radica en su capacidad para minimizar errores en modelos complejos, mejorando así la precisión general del aprendizaje automático. Además, se implementan algoritmos especializados para abordar problemas no convexos.

La optimización de dos niveles es una herramienta clave en el diseño y operación de redes industriales sostenibles. Ejemplos notables se incluyen en los estudios de Ramos et al., 2016 donde se optimiza el uso del agua a partir de modelar la situación mediante juegos de múltiples líderes-seguidores, priorizando objetivos ambientales y económicos. Los resultados mostraron que las empresas participantes lograron beneficios significativos con las formulaciones **KKT** (Karush-Kuhn-Tucker) del modelo **MLFG** (Multi-Leader-Follower Game) utilizado. Además, en Ramos et al., 2016 se destaca la influencia de la estructura del juego en la configuración óptima, sugiriendo la necesidad de un diseño óptimo para cada planta dentro del EIP. Los enfoques **SLMFG** (Single-Leader-Multifollower Game) y **MLSFG** (Multi-Leader-Single-Follower Game) presentan variaciones en el rol de los participantes: en SLMFG, las empresas son seguidoras y la autoridad es líder, mientras que en MLSFG, ocurre lo contrario. Se resalta que el enfoque MLFG logra equilibrar objetivos económicos y ambientales, generando ahorros significativos mediante la reutilización de recursos. Los resultados indican una reducción en el consumo de agua fresca gracias a las estrategias implementadas, utilizando herramientas como GAMS para modelar los problemas de optimización, ver Ramos et al., 2016. Además, en Ramos et al., 2018 los autores introducen el concepto de autoridad ambiental en el diseño de redes de servicios públicos, utilizando juegos

de múltiples líderes-seguidores y reformulaciones KKT. En el ámbito del despacho energético bajo restricciones de carbono, en Gu et al., 2020 se modela incentivos de precios de energía en un parque industrial, demostrando que un enfoque binivel puede simultáneamente mejorar el impacto ambiental y los beneficios económicos, utilizando un procedimiento iterativo primal-dual.

Además estudios como los de Bhavsar y Verma, 2021 investigan sobre la aplicación de una política de subsidios para gestionar el riesgo de materiales peligrosos en una red de transporte ferroviario. En este modelo, el gobierno actúa como líder, ofreciendo subsidios para incentivar al operador ferroviario (el seguidor) a usar rutas alternativas que eviten los enlaces de alto riesgo en la red, utilizando el enfoque **SLSF** (Single-Leader-Single-Follower). Los autores utilizan una reformulación de KKT para resolver el problema y aplican su método a un caso real en los Estados Unidos. Se demuestra que incluso subsidios modestos pueden resultar en una reducción significativa del riesgo.

El estudio de los problemas de dos niveles es de interés de la comunidad científica no solo porque modelan las situaciones antes mencionadas, sino porque es complejo obtener propiedades de sus soluciones así como su cálculo numérico. El concepto mismo de solución del problema binivel es complejo. La decisión del líder es solo respecto a un grupo de variables, mientras que las otras influyen en la función objetivo, pero no son decisión de él. Si para un mismo valor de las variables del líder el problema del seguidor tiene diferentes soluciones óptimas, el valor de la función objetivo del líder no estará determinado, sino que depende de cuál de los óptimos escogió el otro agente.

Para obtener las condiciones de optimalidad y los algoritmos de solución de los modelos binivel se reportan dos enfoques fundamentales. En Dempe y Zemkoho, 2020a se usa la función valor extremal. Esto significa que para todo valor de x, se considera la minimización de la función objetivo del líder en el conjunto dado por

las restricciones de ambos individuos y la condición de que la función objetivo del seguidor es menor o igual que el valor más pequeño que alcanza en el conjunto de soluciones factibles del seguidor.

Otro enfoque clásico consiste en sustituir el problema del nivel inferior por la condición de KKT, lo que permite transformar problemas de optimización binivel en programas matemáticos con restricciones de equilibrio (MPEC), facilitando así su resolución, ver Caselli et al., 2024; Dempe y Freiberg, 2003; Dempe y Zemkoho, 2020a. Conocido como *enfoque KKT* en la literatura, es una de las formas más utilizada para la resolución de problemas de dos niveles en la actualidad, ver Bouza-Allende et al., 2021.

Dado que los problemas de optimización de ese tipo son inherentemente difíciles de resolver debido a su naturaleza **NP-hard**, ver Bard, 1991; Jeroslow, 1985 o incluso $\Sigma P2-hard$, ver Cerulli, 2021; Dempe y Zemkoho, 2020a, se han desarrollado diversos enfoques para abordar su complejidad computacional. Sin embargo, estos enfoques suelen ser computacionalmente intensivos para problemas de gran escala, ver Cerulli, 2021. En paralelo, los algoritmos metaheurísticos, como los evolutivos, han ganado relevancia al proporcionar aproximaciones eficientes en casos no lineales o no convexos, donde las soluciones exactas son inalcanzables en tiempos razonables, ver Sinha et al., 2017. Otro enfoque destacado es el uso de métodos de descomposición, los cuales dividen el problema en subproblemas más manejables que pueden resolverse iterativamente, ver Floudas y Pardalos, 1990.

Estas técnicas son particularmente útiles en aplicaciones prácticas, como los mercados de energía o los modelos de sostenibilidad, ver Siddiqui y Gabriel, 2012. A pesar de estos avances, existen desafíos abiertos. La escalabilidad sigue siendo un problema crítico, ya que el crecimiento exponencial de las opciones en problemas de gran tamaño limita la aplicabilidad de los métodos exactos, ver Dempe y Zemkoho, 2020a. Asimismo, los problemas no convexos carecen de garantías

de convergencia hacia el óptimo global, lo que los hace especialmente difíciles de abordar. Finalmente, la incorporación de incertidumbre en los modelos agrega una capa adicional de complejidad, lo que demanda nuevos enfoques híbridos que combinen algoritmos exactos y heurísticos para mejorar la eficiencia computacional sin sacrificar la calidad de las soluciones, ver Cerulli, 2021; Sinha et al., 2017. Estos avances y desafíos reflejan la importancia de diseñar algoritmos personalizados que aprovechen las estructuras particulares de cada problema binivel. Las aplicaciones industriales, como el diseño de redes ecoindustriales y la gestión de mercados energéticos, destacan la necesidad de enfoques que equilibren precisión y tiempo de cálculo, haciendo de la optimización de dos niveles un área de investigación activa con un impacto significativo en la práctica.

Los algoritmos que se basan en las condiciones KKT incluyen una variedad de métodos, como técnicas de branch-and-bound y métodos de suavizado, ver Dempe y Zemkoho, 2020a. Además, se menciona que los algoritmos SQP (Sequential Quadratic Programming) también se fundamentan en las condiciones KKT para resolver problemas suaves con restricciones. Este hecho demuestra su versatilidad diversas áreas de la optimización.

Los problemas de optimización de dos niveles y la formulación KKT son inherentemente no convexos, lo que implica que los métodos de optimización convexa no son directamente aplicables. Esta no convexidad puede llevar a soluciones subóptimas y a dificultades para encontrar una solución global. Aunque en muchos casos las funciones involucradas en la definición del modelo sean convexas, la estructura general del problema sigue siendo no convexa.

En resumen, obtener garantías sobre soluciones en problemas de optimización de dos niveles es un desafío complejo debido a que por su no convexidad inherente, se presentan dificultades para escapar de óptimos locales, la posible falta de unicidad y la inestabilidad en las soluciones. Los algoritmos frecuentemente hallan

puntos estacionarios, que no siempre corresponden a las soluciones óptimas deseadas. Por lo tanto, es crucial desarrollar métodos especializados que aborden estos problemas y permitan encontrar soluciones globales o aproximaciones adecuadas, ver Dempe y Zemkoho, 2020a.

En este contexto, se ha estudiado la estructura genérica de los problemas de complementariedad (MPCC) que surgen del enfoque KKT y Fritz-John (FJ) aplicado a un problema de dos niveles. Se ha demostrado que, para una clase amplia de estos, la condición de que independencia lineal (MPCC-LICQ) se cumple en todos los puntos factibles. Sin embargo, las condiciones de complementariedad estricta (MPCC-SC) y las condiciones de segundo orden (MPCC-SOC) pueden fallar en puntos críticos (estacionarios), incluso en situaciones genéricas, ver Bouza-Allende y Still, 2012. Esta situación complica aún más la obtención de garantías sobre la solución.

Es importante señalar que existen casos singulares donde los puntos estacionarios pueden ser problemáticos, especialmente cuando el multiplicador (α) asociado a la condición de KKT del problema de nivel inferior es igual a cero. En tales circunstancias, la condición MPCC-SC puede no cumplirse, lo que podría llevar a que el método KKT no funcione adecuadamente, ver Bouza-Allende y Still, 2012.

Basándose en la caracterización de los diferentes tipos de puntos estacionarios establecida por Flegel y Kanzow, 2003, esta tesis propone desarrollar un generador de problemas que, dado un punto y las funciones F, g_s, f y v_s que definen un problema de dos niveles, agregarles funciones polinomiales de primer o segundo grado de forma tal que el punto inicial dado sea un punto crítico del problema creado. Este generador facilitará el estudio del comportamiento de algoritmos conocidos en problemas con ahora al menos un punto estacionario conocido. El usuario además podrá decidir si quiere un punto crítico con multiplicadores arbitrarios o si $\alpha = \vec{0}$, lográndose estudiar las clases de puntos críticos que aparecen en el caso genérico,

o sea en un clase amplia y significativa de los problemas generados.

La tesis está compuesta de 3 capítulos luego del capítulo de introducción, en un segundo capítulo se mostrará la notación que se empleará, se define el problema de dos niveles con un líder y un seguidor, y se explica la teoría matemática para su transformación en un problema MPEC, así como los algoritmos de Julia que se utilizarán en ella. En el tercer capítulo se explicará la implementación algorítmica propuesta anteriormente y su correcta utilización. En el cuarto capítulo se analizarán los resultados obtenidos por el algoritmo propuesto y su comparación con algoritmos implementados en el entorno Julia. Finalmente, se presentarán las conclusiones y recomendaciones del trabajo realizado.

2. PRELIMINARES

La optimización binivel es un problema de optimización donde un subconjunto de variables debe ser la solución óptima de otro problema de optimización, parametrizado por las variables restantes. Este problema tiene dos niveles jerárquicos de decisión: el nivel superior (líder) y el nivel inferior (seguidor). Este tiene dos características principales: primero, el problema del nivel inferior actúa como una restricción para el nivel superior; segundo, la solución del nivel inferior depende de las variables del nivel superior, creando una interdependencia entre ambos niveles. Por ello, el líder debe anticipar la respuesta óptima del seguidor al tomar decisiones. En términos abstractos, la optimización binivel busca minimizar una función objetivo de nivel superior, $F(x, y)$, donde x son las variables de decisión del líder y y son las variables del seguidor.

En este capítulo se abordará los conocimientos necesarios para el desarrollo de esta tesis. Consta de secciones sobre los conceptos fundamentales de la Optimización Binivel del Caso Optimista, así como su reformulación KKT, además de nociones clave sobre los Problemas Matemáticos con Restricciones de Equilibrio (MPEC), y por último la modelación de los problemas binivel en el lenguaje de programación Julia y métodos seleccionados implementados en este lenguaje para su resolución.

2.1. Optimización Binivel Caso Optimista

En los problemas de optimización binivel se consideran dos enfoques principales: el optimista y el pesimista.

$$\begin{aligned} &\min_{x} \ F(x,y) \\ &\text{sujeto a} \\ &G(x) \leq 0, \\ &y \in \underset{y}{\operatorname{argmin}} \{f(x,y) \mid v(x,y) \leq 0\} \end{aligned} \tag{2.1.1}$$

En el enfoque optimista, se asume que el seguidor, que actúa en el nivel inferior, elegirá la solución más favorable para el líder, quien toma decisiones en el nivel superior. Este es considerado más tratable y, en ciertas situaciones favorables, puede simplificarse a un problema convexo. Además, en el contexto de múltiples objetivos, el enfoque optimista permite alcanzar el mejor frente de Pareto posible, ver Dempe y Zemkoho, 2020a.

$$\begin{aligned} &\min_{x} \ \inf_{y \in S(x)} F(x,y) \\ &\text{con } S(x) = \underset{y}{\operatorname{argmin}} \{f(x,y) \mid v(x,y) \leq 0\} \end{aligned} \tag{2.1.2}$$

Fig. 2.1. Problema de optimización bajo el enfoque optimista.

Por otro lado, el enfoque pesimista asume que el seguidor seleccionará la opción menos favorable para el líder entre las soluciones óptimas disponibles, el cual es más complejo de resolver y puede incluso no tener solución. A menudo, se requieren reformulaciones para abordar estos problemas, lo que lo convierte en un reto teórico y computacional significativo en situaciones de múltiples objetivos, conduce al peor

frente de Pareto posible, ver Sinha et al., 2017.

$$\begin{aligned} &\min_{x} \sup_{y \in S(x)} F(x,y) \\ &\text{con } S(x) = \operatorname*{argmin}_{y} \{ f(x,y) \mid v(x,y) \leq 0 \} \end{aligned} \tag{2.1.3}$$

Fig. 2.2. Problema de optimización bajo el enfoque pesimista.

Es relevante destacar que la mayoría de la literatura se centra en el enfoque optimista debido a su mayor facilidad de tratamiento. Sin embargo, el otro también tiene su utilidad, especialmente en la modelación de situaciones donde se considera la aversión al riesgo, ver Dempe y Zemkoho, 2020a. En este contexto, los términos "líder" y "seguidor" se utilizan para describir los roles en el modelo a optimizar; el líder toma decisiones considerando las posibles reacciones del seguidor, quien a su vez reacciona seleccionando su mejor opción, ver Sinha et al., 2017.

Dado que en la tesis trataremos sobre problemas binivel de enfoque optimista mostraremos algunos resultados referidos al modelo resultante.

Optimización Binivel Optimista. Un problema de optimización binivel optimista, se formula como:

$$\begin{aligned} \min_{x,y} \quad & F(x,y) \\ \text{s.t.} \quad & g_i(x,y) \leq 0, i = 1 \dots q, \\ & h_i(x,y) = 0, i = 1 \dots r, \\ & y \in S(x), \end{aligned} \tag{2.1.4}$$

donde $S(x)$ es el conjunto de soluciones óptimas del problema parametrizado por x

$$\begin{aligned} \min_{y \in Y(x)} \quad & f(x,y) \\ \text{s.t.} \quad & v_j(x,y) \leq 0, j = 1 \dots s, \\ & u_j(x,y) = 0, j = 1 \dots t, \end{aligned} \tag{2.1.5}$$

donde

$$x \in R^n, \qquad y \in R^m$$

$$F(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \qquad g_i(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \quad h_i(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R},$$
$$f(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \qquad v_j(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \quad u_j(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R},$$

Esta minimización está sujeta a dos tipos de restricciones:

- **Restricciones explícitas:**
  - Desigualdades del líder: $g_i(x, y) \leq 0 \quad \forall i$
  - Igualdades del líder: $h_i(x, y) = 0 \quad \forall i$
- **Restricciones implícitas:**
  - Impuestas por el seguidor: $y \in \arg\min\{f(x, y) : v_j(x, y) \leq 0,\ u_j(x, y) = 0\}$,
    donde $f(x, y)$ es la función objetivo del seguidor
- También se puede asumir que ambas variables tienen restricciones conjuntas, las restricciones $g_i(x, y), \quad h_i(x, y), \quad v_j(x, y), \quad u_j(x, y)$ que dependen de las variables x y y.

En el caso del enfoque optimista si para el nivel inferior existen más de un punto que resuelve el problema del nivel inferior y tomará la que más beneficie al nivel superior. Por simplicidad solo se consideran restricciones de desigualdad.
Los problemas de dos niveles pueden ser reformulados en un problema de un solo nivel al reemplazar el problema del nivel inferior por las condiciones KKT de este en las restricciones del primer nivel.

$$\begin{aligned} \min_{x,y,\lambda_j} \quad & F(x,y) \\ s.a \begin{cases} & g_i(x,y) \leq 0, i = 1 \dots q, \\ & \nabla_y f(x,y) + \sum_{j=1}^{s} \nabla_y v_j(x,y)\lambda_j = 0, \\ & v_j(x,y) \leq 0, j = 1 \dots s, \\ & v_j(x,y)\lambda_j = 0, j = 1 \dots s, \\ & \lambda_j \geq 0, j = 1 \dots s \end{cases} \end{aligned} \tag{2.1.5}$$

MPEC resultante

Los tres últimos grupos de restricciones expresan que v y λ están restringidas en signo y que al menos una es 0. Estas condiciones son conocidas como **restricciones de complementariedad**. Estos modelos corresponden a la clase de problemas de programación matemática con restricciones de complementariedad (MPEC). A continuación, presentamos los resultados de esta área necesarios para el desarrollo de esta tesis.

2.2. Sobre los Programas Matemáticos con Restricciones de Equilibrio (MPEC)

Un **Programa Matemático con Restricciones de Equilibrio (MPEC)** es un tipo de programa no lineal que incluye restricciones de equilibrio, específicamente, restricciones de complementariedad.

$$\begin{aligned} \text{mín} \quad & f(z) \\ \text{s.t.} \quad & g(z) \leq 0, \quad h(z) = 0 \\ & G(z) \geq 0, \quad H(z) \geq 0, \quad G(z)^T H(z) = 0 \end{aligned} \tag{2.2.0}$$

Definición de MPEC

donde $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$, $G : \mathbb{R}^n \to \mathbb{R}^\ell$, y $H : \mathbb{R}^n \to \mathbb{R}^\ell$ son funciones continuamente diferenciables. Debido al término de complementariedad en las restricciones, los programas de este tipo son algunas veces referidos como programas matemáticos

Estas restricciones de complementariedad se expresan como:

$$G(z) \geq 0, \quad H(z) \geq 0, \quad \text{y} \quad G(z)^T H(z) = 0 \tag{2.2.1}$$

Los MPEC incluyen restricciones donde el producto de dos funciones (G y H) debe ser cero, y ambas funciones deben ser no negativas. Esto se conoce como restricción de complementariedad. Debido a las restricciones de complementariedad, los MPEC no cumplen con las condiciones de regularidad estándar, lo que hace que las condiciones de KKT no sean directamente aplicables como condiciones de optimalidad de primer orden. Los MPEC son utilizados para modelar problemas donde existen restricciones de equilibrio, como problemas de ingeniería y economía, ver Dempe y Zemkoho, 2020a; Flegel y Kanzow, 2003.

A continuación como en Flegel y Kanzow, 2003, se introduce las siguientes definiciones.

Definición 1 (Conjunto de Índices): Dado un vector factible z^* del MPEC (2.2.0), definimos los siguientes *conjuntos de índices*:

$$\begin{aligned} \alpha := \alpha(z^*) &:= \{i | G_i(z^*) = 0, H_i(z^*) > 0\} \\ \beta := \beta(z^*) &:= \{i | G_i(z^*) = 0, H_i(z^*) = 0\} \\ \gamma := \gamma(z^*) &:= \{i | G_i(z^*) > 0, H_i(z^*) = 0\} \end{aligned} \tag{2.2.2}$$

Para definir calificaciones de restricción alteradas, introducimos el siguiente problema, dependiente de z^*:

Definición 2 (Programa No Lineal Ajustado (TNLP)): Un *Programa No Lineal*

Ajustado $TNLP := TNLP(z^*)$ es:

$$\begin{aligned} \text{mín} \quad & f(z) \\ \text{s.t.} \quad & g(z) \leq 0 \quad h(z) = 0 \\ & G_{\alpha\cup\beta}(z) = 0 \quad G_{\gamma}(z) \geq 0 \\ & H_{\alpha}(z) \geq 0 \quad H_{\gamma\cup\beta}(z) = 0 \\ & \text{Problema No Lineal Ajustado (TNLP)} \end{aligned} \tag{2.2.3}$$

El TNLP (2.2.3) puede ahora ser usado para definir variantes MPEC adecuadas de las calificaciones de restricción estándar de independencia lineal, Mangasarian-Fromovitz y Mangasarian-Fromovitz estricta (LICQ, MFCQ, y SMFCQ en su forma abreviada).

Definición 3 (MPEC-LICQ): El MPEC (2.2.0) se dice que satisface la *MPEC-LICQ* (MPEC-MFCQ, MPEC-SMFCQ) en un vector factible z^* si el correspondiente TNLP(z^*) satisface la LICQ (MFCQ, SMFCQ) en ese vector z^*.

En este punto, es importante notar que bajo MPEC-MFCQ, un mínimo local z^* del MPEC (2.2.0) implica la existencia de un multiplicador de Lagrange λ^* tal que (z^*, λ^*) satisface las condiciones KKT para el programa (2.2.3). Por lo tanto, si asumimos que MPEC-MFCQ se cumple para un mínimo local z^* del MPEC (2.2.0), podemos usar cualquier multiplicador de Lagrange λ^* (que ahora sabemos que existe) para definir el MPEC-SMFCQ, es decir, tomando (z^*, λ^*).
En el contexto de los MPEC en Flegel y Kanzow, 2003 se exponen varios tipos de puntos estacionarios que son cruciales para analizar la optimalidad, los cuales son los siguientes:

Definición 4 (Punto Factible): Un *punto factible* z del MPEC se llama débilmente estacionario si existe un multiplicador de Lagrange $\lambda = (\lambda^g, \lambda^h, \lambda^G, \lambda^H)$ tal que se cumplen las siguientes condiciones:

$$\begin{aligned}
&\nabla f(z) + \sum_{i=1}^{m} \lambda_i^g \nabla g_i(z) + \sum_{i=1}^{p} \lambda_i^h \nabla h_i(z) - \sum_{i=1}^{\ell} [\lambda_i^G \nabla G_i(z) + \lambda_i^H \nabla H_i(z)] = \vec{0} \\
&\lambda_\alpha^G \text{ libre}, \quad \lambda_\beta^G \text{ libre}, \quad \lambda_\gamma^G = 0, \\
&\lambda_\gamma^H \text{ libre}, \quad \lambda_\beta^H \text{ libre}, \quad \lambda_\alpha^H = 0, \\
&g(z) \leq 0, \quad \lambda^g \geq 0, \quad (\lambda^g)^T g(z) = 0.
\end{aligned} \tag{2.2.4}$$

Este concepto de estacionariedad, sin embargo, es una condición relativamente débil. Existen conceptos más fuertes de estacionariedad que se derivan y estudian en otros lugares. En particular, un punto factible z con el correspondiente multiplicador de Lagrange $\lambda = (\lambda^g, \lambda^h, \lambda^G, \lambda^H)$ se llama:

- **Definición** 5 (Punto C-estacionario): El punto z es: *C-estacionario* si, para cada $i \in \beta$, $\lambda_i^G \lambda_i^H \geq 0$ se cumple.
- **Definición** 6 (Punto M-estacionario): El punto z es: *M-estacionario* si, para cada $i \in \beta$, o bien $\lambda_i^G, \lambda_i^H > 0 \vee \lambda_i^G \lambda_i^H = 0$.
- **Definición** 7 (Punto Fuertemente estacionario): El punto z es: *fuertemente estacionario* o *estacionario primal-dual* si, para cada $i \in \beta$, $\lambda_i^G, \lambda_i^H \geq 0$.

Teorema 1: Sea $z^* \in \mathbb{R}^n$ un mínimo local del MPEC (2.2.0). Si MPEC-LICQ se cumple en z^*, entonces existe un único multiplicador de Lagrange λ^* tal que (z^*, λ^*) es *fuertemente estacionario*.

2.3. Modelación en Julia

BilevelJuMP.jl es un paquete de Julia diseñado para modelar y resolver problemas de **optimización binivel**, también conocidos como problemas de optimización de

dos niveles o jerárquica, para más detalles ver Garcia et al., 2022. Estos problemas se caracterizan por tener dos niveles de decisión: un nivel superior y un nivel inferior, donde las decisiones del nivel superior influyen en las decisiones del nivel inferior, y viceversa Garcia et al., 2022. Este paquete permite abordar una amplia variedad de tipos de problemas, incluyendo:

- **Problemas de optimización binivel generales:** BilevelJuMP.jl facilita la modelación de problemas que pueden representarse en la sintaxis de JuMP, ver documentación en Lubin et al., 2023, incluyendo restricciones lineales y no lineales, variables continuas y enteras, y diferentes tipos de objetivos.

- **Problemas con restricciones cónicas en el nivel inferior:** Maneja problemas donde el nivel inferior tiene una estructura de **programación cónica**, definiendo restricciones a través de conos convexos. Esto es útil para aplicar condiciones KKT en la reformulación del problema como MPEC.

- **Problemas con restricciones variadas en el nivel superior:** Permite gestionar una variedad de restricciones compatibles con JuMP, tales como restricciones cónicas, cuadráticas, no lineales y enteras.

- **Problemas con restricciones de equilibrio:** Facilita la transformación de problemas binivel en problemas de programación matemática con restricciones de equilibrio (MPEC) y ofrece métodos para abordar las restricciones de complementariedad que surgen en estos casos.

- **Problemas con variables duales del nivel inferior en el nivel superior:** Permite la utilización de variables duales del nivel inferior explícitamente como variables en el nivel superior, lo cual es crucial para modelar problemas como la fijación de precios en mercados de energía.

- **Problemas con diferentes tipos de reformulaciones:** Los usuarios pueden experimentar con diversas reformulaciones para las restricciones de complementariedad en los problemas MPEC, incluyendo SOS1, restricciones de indicador, Fortuny-Amat y McCarl (Big-M), entre otros.
- **Problemas que requieren solvers MIP y NLP:** BilevelJuMP.jl puede utilizar tanto solucionadores de **programación lineal mixta entera (MIP)** como solucionadores de **programación no lineal (NLP)**, dependiendo de las características del problema y la reformulación elegida.

A pesar de sus capacidades, BilevelJuMP.jl presenta algunas limitaciones, entre las cuales se encuentran:

- Enfrentar dificultades en problemas altamente no lineales o con estructuras de optimización complejas que no se puedan representar adecuadamente en la sintaxis de JuMP.
- Existencia de ciertas restricciones que podrían no ser compatibles o que requieren transformaciones adicionales que podrían complicar el modelo.
- El problema es de gran escala afectando el rendimiento del solver, el cual puede ser un factor crítico.
- La formulación y resolución de problemas muy específicos o especializados podrían no estar completamente optimizadas en el paquete.

2.4. Métodos de Reformulación para Optimización Binivel

La optimización binivel presenta desafíos particulares debido a su naturaleza jerárquica y las condiciones de complementariedad resultantes. A continuación, se presentan los principales métodos de reformulación implementados en la literatura, basados en la transformación del MPEC (2.1.5).

2.4.1. Método Big-M

El método Big-M (Fortuny-Amat y McCarl) es una técnica fundamental para reformular problemas de optimización binivel en problemas MPEC. Este método aborda específicamente las condiciones de complementariedad que surgen en estas reformulaciones, transformando el problema original en un problema de programación lineal mixta entera (MILP).

La reformulación mediante Big-M introduce un parámetro M suficientemente grande y variables binarias para transformar las condiciones de complementariedad no lineales en restricciones lineales. Para una condición de complementariedad $v_j(x, y)\lambda_j = 0$ en (2.1.5), el método introduce una variable binaria $f_j \in \{0, 1\}$ y cotas superiores M_p, M_d bajo las siguientes restricciones:

$$\begin{aligned} v_j(x, y) &\geq -M_p(1 - f_j) \\ \lambda_j &\leq M_d f_j \\ f_j &\in \{0, 1\} \end{aligned} \tag{2.4.1}$$

donde M_p y M_d son valores grandes para las variables primales y duales, respectivamente. La efectividad del método depende crucialmente de la selección apropiada de estos valores, que deben ser suficientemente grandes para no excluir la solución óptima, pero no excesivamente grandes para evitar inestabilidades numéricas Garcia et al., 2022.

2.4.2. Método SOS1

El método de Conjuntos Ordenados Especiales tipo 1 (SOS1) evita el uso de parámetros Big-M mediante restricciones de tipo conjunto. Para cada par complementario (v_j, λ_j):

$$[v_j(x, y); \lambda_j] \in \text{SOS1} \tag{2.4.2}$$

Esta restricción fuerza que al menos una variable en el par sea cero, preservando la no linealidad original sin necesidad de cotas. Es particularmente eficaz en problemas con estructura cónica Garcia et al., 2022.

2.4.3. Método ProductMode

El método ProductMode representa un enfoque directo para manejar las condiciones de complementariedad en su forma de producto original. Este método es particularmente útil cuando se trabaja con solucionadores de programación no lineal (NLP), aunque no garantiza la optimalidad global.
La implementación del ProductMode mantiene la restricción de complementariedad en su forma original:

$$v_j(x, y) \cdot \lambda_j \leq t \tag{2.4.3}$$

donde $t > 0$ es un parámetro de regularización pequeño. Esta formulación, aunque no satisface las condiciones de calificación de restricciones estándar, es útil para obtener soluciones iniciales y puede ser especialmente efectiva cuando se combina con solucionadores NLP, ver Garcia et al., 2022.

3. IMPLEMENTACIÓN

En este capítulo se mostrará la forma y los pasos para generar el problema deseado en el contexto de un generador de puntos estacionarios para problemas binivel. Este capítulo aborda el diseño y la implementación del generador, explicando detalladamente las metodologías empleadas, los criterios necesarios para asegurar la validez de los puntos obtenidos y los algoritmos utilizados en cada etapa del proceso.

3.1. Nomenclatura a utilizar:

$F(x, y)$	Función del líder.
$g_i(x, y)$	Restricciones de desigualdad del líder $i = 1 \dots q$.
$h_i(x, y)$	Restricciones de igualdad del líder $i = 1 \dots r$.
$f(x, y)$	Función del seguidor.
$v_j(x, y)$	Restricciones de desigualdad del seguidor $j = 1 \dots s$.
$u_j(x, y)$	restricciones de igualdad del seguidor $j = 1 \dots t$.
$v_j^\star$	Restricción de desigualdad del seguidor activa después de modificarse el problema $j = 1 \dots s$.
z_{est}	Punto que se desea que sea estacionario.
J_0^g	Índices activos del líder.
J_0^v	Índices activos del seguidor.
α	Vector de entrada de dimensión igual cantidad variables seguidor.
μ_i	Multiplicador asociado al $g_i(x, y)$ en el líder.
β_j	Multiplicador asociado al $v_j^\star$ en el líder.
λ_j	Multiplicador asociado al $v_j^\star$ en el seguidor.
γ_j	Valor asociado a cada $v_j^\star$ con $\lambda_j = 0$.

Tab. 3.1. Nomenclatura a utilizar

3.2. Generación del problema

A continuación se mostrarán los pasos a seguir para generar el problema deseado donde el punto sea estacionario de la clase específica.

3.2.1. Requerimientos de entrada

Para generar un problema es necesario inicialmente introducir un problema de optimización binivel del tipo SLSF, un punto (z_{est}), el cual se desea que sea estacionario y los multiplicadores con signo o valores con ciertas condiciones en dependencia del tipo de estacionariedad requerida.

El problema de entrada tiene que cumplir con ser un programa de optimización binivel como 2.2. Debe introducirse también sus índices activos de las restricciones de desigualdad en cada nivel para z_{est} :

- Nivel Superior:

$$J_0^G = \{i | g_i(x, y) = 0\} \tag{3.2.1}$$

  donde estos índices activos tienen un μ_i relacionados.

- Nivel Inferior: Se tiene en cuenta con respecto a los valores del multiplicador λ_i y $v_j(x, y)$.

  - $$J_1^v = \{j | v_j(x, y) = 0 \wedge \lambda_j = 0\} \tag{3.2.2}$$

  - $$J_2^v = \{j | v_j(x, y) = 0 \wedge \lambda_j > 0\} \tag{3.2.3}$$

  - $$J_3^v = \{j | v_j(x, y) < 0 \wedge \lambda_j = 0\} \tag{3.2.4}$$

  Cada índice activo tiene multiplicadores β_j y en dependencia del caso γ_j asociados.

Para cada $i \in J_0^G$ (3.2.1) debe de asignarse su multiplicador $\mu_i \geq 0$ correspondiente. Además se debe conocer el valor del $\vec{\alpha}$, para el nivel inferior debe tenerse en cuenta si $\alpha = \vec{0}$ en cuyo caso afirmativo γ_j es un valor de entrada

Para obtener las diferentes clases de puntos estacionarios debe de introducirse β_j, γ_j, en caso de ser este último valor de entrada tal que, los multiplicadores β_j y γ_j de forma tal que en los indices activos J_1^v (**??**):

- **Punto C-Estacionario:**

$$\beta_j * \ \gamma_j \ >= 0 \tag{3.2.5}$$

- **Punto M-Estacionario**

$$\begin{aligned} &\beta_j \wedge \gamma_j > 0 \\ &\beta_j \quad \text{Libre} \quad \gamma_j = 0 \\ &\beta_j = 0 \quad \gamma_j \quad \text{Libre} \end{aligned} \tag{3.2.6}$$

- **Punto Fuertemente Estacionario**

$$\beta_j \wedge \gamma_j \geq 0 \tag{3.2.7}$$

En caso que $\vec{\alpha} \neq \vec{0}$ se asume que $\gamma_j = 0$ en las condiciones anteriores.

Modificación del Problema Original

Después de tener los datos de entrada necesarios se procede a la modificación del problema de entrada para que este sea estacionario del tipo requerido, z_{est}.

Primero en caso de que $\alpha \neq 0$ en los $v_j(x,y) \in J_0^v$ se procede a calcular el $\vec{b_j}$ de la siguiente forma:

- $v_j(x,y) \in J_2^v$ **3.2.3**:

$$\vec{b_j} = \frac{\alpha \cdot (-\nabla_y v_j(z_{est})^T \cdot \alpha)}{\|\alpha\|_2} \tag{3.2.8}$$

- $v_j(x,y) \in J_1^v \vee J_3^v$ **(??,3.2.4)**

$$\vec{b_j} = \frac{\alpha \cdot ((-\nabla_y v_j(z_{est})^T \cdot \alpha) + \gamma_j)}{\|\alpha\|_2^2} \tag{3.2.9}$$

Después de realizado el calculo del b_j se procede a modificar su índice activo correspondiente:

$$v_j^\star(z_{est}) = v_j(z_{est}) + (\vec{b_j}^T) \cdot (y_1, y_2, \ldots, y_m) \tag{3.2.10}$$

Luego en todas las restricciones del nivel inferior y superior se evalúa el punto y se verifica si es factible. En caso de no ser factible bajo las nuevas restricciones se añaden constantes c_i en el caso de las restricciones del nivel superior y c_j para las del inferior. Esto conlleva a que sea un punto factible en el conjunto de restricciones de ambos niveles sin perjudicar las propiedades relacionadas con la convexidad al ser una adicción de funciones lineales.

Al hacer factible se procede a realizar el KKT del nivel inferior con las modificaciones anteriores evaluado en z_{est}.

Se plantea las condiciones KKT y se halla $\vec{bf}$:

$$\nabla_y f(x,y) + \sum_{j=0}^{|V \in J_0|} (\lambda_j \nabla_y v_j^\star(x,y)) + \vec{bf} = \vec{0} \tag{3.2.11}$$

KKT del problema del nivel inferior

Finalmente podemos construir la reformulación en MPEC como 2.1.5 calculando el $\vec{BF}$ y obteniendo el problema de forma reformulada.

(3.2.12)

$$\nabla_{xy}F(x,y) + \sum_{i=1}^{|G\in J_o|}(\mu_i \nabla_{xy} g(x,y)) + [\nabla_{x,y}\nabla_y f(x,y) + \sum_{j=1}^{|V\in J_0|} \lambda_j \nabla_{xy}\nabla_y v_j^\star(x,y)]\alpha + \sum_{j=1}^{|V\in J_0|}(\beta_j \nabla_{xy} v_j^\star(x,y)) + \vec{BF} = \vec{0}$$

KKT del MPEC

3.3. Implementación Algorítmica y Guía de Usuario

En la implementación computacional del generador propuesto en la sección anteriores se utilizó el lenguaje de programación **Julia**, ver JuliaLang, 2025, dado a su versatilidad en expresiones y funciones matemáticas implementadas en su paquete base y sus bibliotecas externas como **BilevelJuMP**, ver Garcia et al., 2022, que permite resolver problemas binivel SLSF lineales y cuadráticos sin tener que transformar el problema original y **JuMP** Lubin et al., 2023 el cual, es una interfaz robusta para la optimización general, todo ello con unas excelentes prestaciones de cómputo. Por ello se brindará una guía de usuario para el uso del generador.

La implementación algorítmica se ha llevado a cabo mediante la creación de una biblioteca de Julia llamada **ProblemGenerator**, con una sintaxis *JuMP-like* intuitiva. Primero debe tenerse un problema de optimización Binivel SLSF planteado como el 2.2, el punto que se desea que sea estacionario (z_{est}), los índices activos descritos anteriormente según el caso así como sus multiplicadores correspondientes, que cumplirán las propiedades del tipo de estacionariedad requerida y el $\vec{\alpha}$.

Para una mayor facilidad de compresión del uso de la biblioteca se utiliza este

problema ejemplo:

$$\begin{aligned} \min_{x_1,x_2} \quad & x_1^2 y_1^2 y_2 + x_2 \\ \text{s.t.} \quad & x_1 + y_2 - y_1 \leq 9, \\ & \min_{y_1,y_2} \quad x_2^2 y_1^2 y_2 + x_1 \\ & \text{s.t.} \quad x_1^2 y_1^2 + x_2 \leq 0. \end{aligned} \tag{3.3.1}$$

Se debe tener instalado en el entorno de desarrollo de Julia los siguientes módulos:

- Symbolics
- LinearAlgebra

Para la utilización de la biblioteca se procede a su importación de la siguiente forma:

```
using ProblemGenerator
```

Ejemplo 3.1. Importar el Módulo

Para comenzar debe llamarse a la función **GeneratorModel** para crear el modelo base inicial.

Para $\vec{\alpha} = \vec{0}$

Para crear un modelo donde $\vec{\alpha} = \vec{0}$:

```
# Crear el modelo base del generador alpha=0
model=GeneratorModel()

```

Ejemplo 3.2. Crear el modelo para $\vec{\alpha} = \vec{0}$

Para $\vec{\alpha} \neq \vec{0}$

Para crear un modelo donde $\vec{\alpha} \neq \vec{0}$, se tiene que pasar como parámetro el valor del α, el cual será un vector de *Number*:

```
# Crear el modelo base del generador alpha!=0
alpha_vec=Vector::{Number}
model=GeneratorModel(alpha_vec)
```

Ejemplo 3.3. Crear el modelo para $\vec{\alpha} \neq \vec{0}$

En caso que se requiera como entrada un **Problem**, se debe declarar de que nivel se trata de evaluar el **model** en las funciones **Upper** para el nivel superior y **Lower** para el inferior.

Para referirse al nivel superior

```
# Referirse nivel superior
Upper(model)
```

Ejemplo 3.4. Referirse al nivel superior

Para referirse al nivel inferior

```
# Referirse nivel inferior
Lower(model)
```

Ejemplo 3.5. Referirse al nivel inferior

Las variables deben de declararse bajo la siguiente macro **@myvariables**, la cual recibe una función **Upper** o **Lower** que recibe el modelo para las variables del nivel superior y las del nivel inferior respectivamente.

Introduccion de las variables del nivel superior

```
# Se declara en el nivel superior las variables x_1, x_2
@myvariables Upper(model) x_1, x_2
```

Ejemplo 3.6. Introducir las variables del nivel superior

Introducción de las variables del nivel inferior

```
# Se declara en el nivel inferior las variables y_1, y_2
@myvariables Lower(model) y_1,y_2
```

Ejemplo 3.7. Introducir las variables del nivel inferior

3.3.1. Declarar Funciones Objetivo

Para declarar las funciones objetivos debe asumirse que en ambos casos es un problema de minimización. Se utilizará la función **SetObjectiveFunction** que recibe un **Problem** y una expresión de **Num**.

Declaración de una función objetivo del nivel superior

Con:

$$\text{mín}(x_1^2 * y_1^2 * y_2) + x_2$$

```
# Declarar la funcion objetivo del nivel superior
# Min de ((x_1^2)*(y_1^2)*(y_2))+x_2
SetObjectiveFunction(Upper(model),((x_1^2)*(y_1^2)*(y_2)
)+x_2)
```

Ejemplo 3.8. Declarar una función objetivo del nivel superior

Declaración de una función objetivo del nivel inferior

Con:

$$\text{mín}(x_2^2 * y_1^2 * y_2) + x_1$$

```
1    # Declarar la funcion objetivo del nivel inferior
2    # Min de ((x_2^2)*(y_1^2)*(y_2))+x_1
3    SetObjectiveFunction(Lower(model),((x_2^2)*(y_1^2)*(y_2)
   )+x_1)
```

Ejemplo 3.9. Declarar una función objetivo del nivel inferior.

3.3.2. Definición del conjunto de Índices activos

Antes de ilustrar como introducir las restricciones se va a explicar que los tipos de índices activos.

- **Ambos Niveles**:
  - **Normal** si no es un índice activo.
- **Nivel Superior**:
  - **J__0__g** Si es un índice como en 3.2.1.
- **Nivel Inferior**:
  - **J__0__LP__v** Si es un índice como en 3.2.3.
  - **J__0__L0__v** Si es un índice como en **??**.
  - **J__Ne__L0__v** Si es un índice como en 3.2.4.

Los **RestrictionSetType** deben expresar en código de esta forma:

```
1    Normal
2    J_0_g
3    J_0_LP_v
4    J_0_L0_v
```

```
5    J_Ne_L0_v
```

Ejemplo 3.10. Definir el conjunto de índice activo

Los **RestrictionSetType** son un *enum* de Julia.

Para declarar las restricciones se brindan dos funciones:

- **SetLeaderRestriction:** Para el nivel superior.
- **SetFollowerRestriction:** Para el nivel inferior.
- **Nivel Superior:**

  Para declarar las restricciones del nivel superior debe por cada restricción llamarse a la función **SetLeaderRestriction** con:

  - El modelo (**model**)
  - La expresión de la restricción, del tipo **Num**
  - El tipo de restricción, del tipo **RestrictionSetType**
  - El valor de μ_i correspondiente, del tipo **Number**

  Ejemplo para introducir la restricción:

  $$x_1 + y_2 - y_1 \leq 9$$

  Índice activo del tipo J_0_g 3.2.1

  $\mu_i = 0{,}3$

```
1    # Ejemplo de restriccion del nivel superior
2    SetLeaderRestriction(model,x_1+y_2-y_1<=9,J_0_g,0.3)
```

Ejemplo 3.11. Introducir restricción del nivel superior

- **Nivel Inferior:**

  Para declarar las restricciones del nivel inferior debe por cada restricción llamarse a la función **SetFollowerRestriction** con:

  - El modelo (**model**)
  - La expresión de la restricción, del tipo **Num**
  - El tipo de restricción, del tipo **RestrictionSetType**
  - El valor de β_j correspondiente, del tipo **Number**
  - El valor de λ_j correspondiente, del tipo **Number**
  - El valor de γ_j correspondiente en caso de ser valor de entrada, del tipo **Number**

  Ejemplo para introducir la restricción:

  - $\lambda_j \neq 0$**:**

    $((x_1^2) * (y_1^2)) + x_2 \leq 0$

    Índice activo del tipo J_Ne_L0_v3.2.4

    $\beta_j = 0{,}1$

    $\lambda_j = 0{,}5$

```
1        # Para caso lamda_j=0
2        SetFollowerRestriction(model,((x_1^2)*(y_1
   ^2))+x_2<=0,J_Ne_L0_v,0.1,0.5,0)
3
```

- $\lambda_j = 0$:

Análogo al caso anterior pero con $\lambda_j = 0$, por lo que γ_j valor de entrada

$$\beta_j = 0{,}1, \quad \lambda_j = 0, \quad \gamma_j = 0{,}4$$

```
        # Para caso lamda_j!=0
        SetFollowerRestriction(model,((x_1^2)*(y_1
    ^2))+x_2<=0,J_Ne_L0_v,0.1,0,0.4)

```

3.3.3. *Introducir el Punto*

Ahora debe de introducirse el valor del punto que debe ser estacionario de la clase seleccionada. Debe de tenerse en cuenta que para todas las variables declaradas en ambos niveles debe de definirse el valor de la componente.

```
    # Introducir el punto (1,1,1,1)
    SetPoint(model,Dict(x_1=>1,x_2=>1,y_1=>1,y_2=>1))
```

Ejemplo 3.12. Introducir el punto $(1, 1, 1, 1)$

3.3.4. *Generar el Problema*

Finalmente al tener todos los datos previos introducidos se llama al **CreateProblem** pasándole como parámetro el *model* y generará dicho problema imprimiendo en consola este.

```
    # Llamar para generar el problema
    problem=CreateProblem(model)
```

Ejemplo 3.13. Generar el problema

3.3.5. *Ejemplo completo*

Se muestra el ejemplo completo para el problema antes propuesto:

```
# Importar dependencias necesarias
using ProblemGenerator
# Crear el modelo base del generador alpha=0
model=GeneratorModel()
#Declaracion de variables
# Se declara en el nivel superior las variables x_1, x_2
@myvariables Upper(model) x_1, x_2
# Se declara en el nivel inferior las variables y_1, y_2
@myvariables Lower(model) y_1,y_2
# Declarar la funcion objetivo del nivel superior
# Min de ((x_1^2)*(y_1^2)*(y_2))+x_2
SetObjectiveFunction(Upper(model),((x_1^2)*(y_1^2)*(y_2))+x_2)
# Ejemplo de restriccion del nivel superior
SetLeaderRestriction(model,x_1+y_2-y_1>9,J_0_g,0.3)
# Declarar la funcion objetivo del nivel inferior
# Min de ((x_2^2)*(y_1^2)*(y_2))+x_1
SetObjectiveFunction(Lower(model),((x_2^2)*(y_1^2)*(y_2))+x_1)
# Para caso lamda_j!=0
SetFollowerRestriction(model,((x_1^2)*(y_1^2))+x_2<=0,J_Ne_L0_v,0.1,0,0.4)
# Introducir el punto (1,1,1,1)
```

```
21    SetPoint(model,Dict(x_1=>1,x_2=>1,y_1=>1,y_2=>1))
22    # Llamar para generar el problema
23    problem=CreateProblem(model)
```

Ejemplo 3.14. Script

3.3.6. Documentación Oficial

Para mayor información visitar la documentación oficial del generador:

https://fvsb.github.io/Tesis/.

4. EXPERIMENTACIÓN

En este capítulo se experimentará tomando una serie de problemas, ver Zhou et al., 2018, que sean: Lineales, Cuadráticos y No Convexos. En ella primeramente utilizaremos las bibliotecas de Julia para obtener puntos que sean mínimos locales, después añadir valores aleatorios a esos puntos y posteriormente generar problemas estacionarios del tipo: fuerte, M y C . Posteriormente estos problemas modificados serán nuevamente ejecutados por los algoritmos tradicionales de Julia para conocer su efectividad con respecto al valor de la función objetivo del nivel superior.
Para ello tomaremos 15 problemas de optimización binivel. Entre ellos habrá 3 clases: Lineales, Cuadráticos y No Convexos, donde habrá 5 problemas de cada clase. Estos han sido extraídos de Floudas, 1999 para las dos primeras clasificaciones y Zhou et al., 2018 para la última.

Los problemas escogidos para la experimentación fueron:

No Convexos	**Lineales**	**Cuadráticos**
MitsosBarton2006Ex312	ex9.1.1	ex9.2.1
MitsosBarton2006Ex313	ex9.1.2	ex9.2.2
MitsosBarton2006Ex314	ex9.1.8	ex9.2.3
MitsosBarton2006Ex323	ex9.1.9	ex9.2.4
MorganPatrone2006a	ex9.1.10	ex9.2.5

Tab. 4.1. Problemas Seleccionados

4.1. Modelación de la experimentación

Se describirá el proceso de generar la experimentación. Todos los valores, en esta parte, han sido redondeados por exceso a dos cifras después de la coma.
Inicialmente se necesita obtener óptimos de los problemas con los paquetes convencionales de Julia cuyos pasos son los siguientes:

- **Problemas Lineales y Cuadráticos :**
  Con dicho problema se introduce los datos en la interfaz de **BilevelJuMP**, ver Garcia et al., 2022, con el cual se utilizan 3 técnicas entre las ofrecidas por esta:

  - **Big-M :** Con el optimizador High-Performance Solver for Linear Programming (HiGHS) y los valores primal big M = 100, dual big M = 100.
  - **SOS1 :** Con el optimizador Solving Constraint Integer Programs (SCIP).
  - **ProductMode :** Con el optimizador Interior Point Optimizer (Ipopt).

  Cada uno de los resultados de evaluar el problema en cada forma anterior se guarda en un formato *.xlsx* donde por cada optimizador se guarda los parámetros:

  - Estatus del Primal, el cual define si es un punto factible o no.
  - Estatus de la Finalización, si terminó porque encontró un óptimo o se estancó en un óptimo local.
  - Valor de la función objetivo del nivel superior.
  - El punto óptimo encontrado, en caso de ser hallado.

Posteriormente se analizan los resultados de dichos métodos, se selecciona el de mejor evaluación de la función objetivo.

- **Problemas No Convexos :**

  Al **BilevelJuMP** no contar con soporte para esta clase de problemas se utiliza **JuMP**, ver Lubin et al., 2023. Por ello se utiliza la reformulación KKT como la de 2.1.5 y se procede a utilizar la interfaz brindada por este. Para el caso de las restricciones de complementariedad se utiliza **Complementarity**, ver Chkwon, 2025, con el optimizador Ipopt y análogo al caso anterior se extraen los mismos datos.

Generación de los problemas

Se toma el problema original de entrada y el punto obtenido en el paso anterior el cual en cada componente se le hace suma un valor aleatorio entre $1e-10$ y 5, siendo este modificado: z_0^*. Se generan los 3 problemas bajo los 3 tipos de estacionariedad descritos en cada uno, además en cada uno se toma la opción de $\vec{\alpha} = \vec{0}$ y $\vec{\alpha} \neq \vec{0}$. Para los $\vec{\alpha} \neq \vec{0}$ se genera un vector aleatorio donde cada componente está entre $1e-10$ y 3. Luego para los conjuntos de índices activos de las $v_j s$ se dividen en 1/2 del tipo J_1^v (**??**) y 1/4 para los dos restantes. Con respecto a la selección de los multiplicadores β_j y γ_j se se generan valores aleatorios entre $1e-10$ y 10 en caso que estos no tengan que ser 0, para los casos en que haya más de una combinación de los multiplicadores con respecto a su igualdad a 0 se toma un valor aleatorio generado por una distribución uniforme discreta. Finalmente cada problema generado es guardado en un archivo *xlsx* con la siguiente designación: *(nombre del problema)_(Tipo de punto estacionario)(generator)_alpha_((non_zero) si* $\alpha \neq 0$ *y (zero) si* $\alpha = 0$*).xlsx*, donde se guarda:

- Las expresiones de las funciones objetivo de ambos niveles y su valor evaluado en el punto.

- Las restricciones de ambos niveles con sus multiplicadores respectivos, el tipo de indice activo y la evaluación de dicha función en el punto.
- El punto z_0^*.
- El vector $\vec{bf}$.
- El vector $\vec{BF}$.
- El multiplicador $\vec{\alpha}$.

Comparación de los algoritmos de Julia

Después de tener generados los problemas se utilizan los mismos métodos de Julia mencionados anteriormente para obtener óptimos de cada problema generado y se elige como representante el que más haya superado su óptimo o en caso de no superar el que mayor distancia tenga con el valor objetivo inicial.

4.1.1. Resultados:

Se presentan los resultados seleccionados bajo los criterios expuestos anteriormente en las siguientes tablas que contienen:

- El nombre del problema original desde el cual fue modificado para que fuese estacionario de la clase deseada.
- El punto al cual se forzó ser estacionario de la clase requerida y no de un subconjunto de esta, por ejemplo si es C-estacionario no es M-estacionario.
- La evaluación de la función objetivo del punto estacionario.
- El punto óptimo hallado por los algoritmos de Julia.
- La evaluación de la función objetivo del óptimo.

- Método seleccionado.

Problemas Lineales

Tipo de punto	Nombre del problema	Punto estacionario	Valor objetivo del punto estacionario	Punto óptimo	Valor del punto óptimo	Método seleccionado
C-Estacionario	ex9.1.8	(2.55,1.25,4.25,2.15)	-1.72	(3.16,0.79,.5.6,0)	-2.72	Big-M
Fuertemente-Estacionario	ex9.1.10	(52.15,20.25,104.6,2.05)	-31.75	(267,90.35,100,0)	-393.65	Big-M
M-Estacionario	ex9.1.8	(2.55,1.25,4.25,2.15)	-1.72	(3.03,2.85,0,0)	-3.20	Big-M
$\alpha = 0$	ex9.1.8	(2.55,1.25,4.25,2.15)	-1.72	(3.34,0.8,3.68,0)	-4.05	Big-M

Tab. 4.2. Problemas Lineales Seleccionados

Problemas Cuadráticos

Tipo de punto	Nombre del problema	Punto estacionario	Valor objetivo del punto estacionario	Punto óptimo	Valor del punto óptimo	Método seleccionado
C-Estacionario	ex9.2.1	(1.85,4.65)	116.01	(4.1,0)	1.64	SOS1
Fuertemente-Estacionario	ex9.2.1	(1.85,4.65)	116.01	(0.15,1.12)	33.92	ProductMode
M-Estacionario	ex9.2.5	(4.75,4.05)	7.27	(2.53,2.83)	0.91	Product Mode
$\alpha = 0$	ex9.2.3	(1.55,2.7,-5.1,-8.65)	-10.25	(0.0,-5.1,-10)	-14.7	Big-M

Tab. 4.3. Problemas Cuadráticos Seleccionados

Problemas No Convexos

Tipo de punto	Nombre del problema	Punto estacionario	Valor objetivo del punto estacionario	Punto óptimo	Valor del punto óptimo	Método seleccionado
C-Estacionario	MitsosBarton2006Ex312	(0.8,1.85)	34,94	(0.8,1.8)	34.87	JuMP
Fuertemente-Estacionario	MitsosBarton2006Ex314	(2.1,3.3)	14.31	(2.1,-1.45)	5.52	JuMP
M-Estacionario	MitsosBarton2006Ex312	(0.8,1.85)	34,94	(0.8,-0.89)	6.48	JuMP
$\alpha = 0$	MitsosBarton2006Ex313	(2.3,4.45)	-2.15	(2.3,4.47)	-2.17	JuMP

Tab. 4.4. Problemas No Convexos Seleccionados

Análisis de los resultados

En el caso de los problemas lineales los 3 algoritmos generalmente dieron con puntos con similar valor de la función objetivo aunque se tomó como referencia Big-M dado que fue el más eficiente en tiempo y si redondeamos a 2 números después de la coma son prácticamente todos iguales. En los problemas cuadráticos se encontró

con mayor diferencia entre el valor de la función en el punto estacionario que en el óptimo local alcanzado, en algunos de estos problemas Big-M no pudo hallar un óptimo con calidad posiblemente debido a sus parámetros y continua la tendencia del problema que parte desde un punto fuertemente estacionario de tener el mayor rango de mejora. En los no convexos continua esta hipótesis dado que los puntos M-estacionarios y fuertemente estacionarios son los de mayor requerimientos, fueron los que lograron encontrar mejores óptimos.

5. CONCLUSIONES

En esta tesis se desarrolló un algoritmo para la generación de problemas de optimización binivel con características específicas de estacionariedad en puntos determinados. El mismo tiene la capacidad de modificar problemas originales para garantizar la factibilidad y estacionariedad en puntos dados, aprovechando las capacidades del lenguaje de programación Julia, que destaca por su alto rendimiento computacional y sintaxis adaptada a problemas de optimización.

La experimentación se llevó a cabo sobre tres categorías fundamentales de problemas: lineales, cuadráticos y no convexos. El proceso experimental comenzó con la obtención de puntos mínimos utilizando bibliotecas establecidas como Bilevel-JuMP y JuMP. Posteriormente, se generaron problemas estacionarios mediante la adición de componentes aleatorias a estos puntos para cada clase de problema definida.

Los problemas modificados fueron sometidos nuevamente a algoritmos tradicionales implementados en Julia para evaluar su efectividad frente a los puntos estacionarios generados. Se realizó un análisis comparativo destacando los casos más relevantes de cada categoría de puntos estacionarios, considerando la evaluación de la función objetivo del nivel superior en el punto donde se garantizó la estacionariedad, contrastándola con los resultados obtenidos por las bibliotecas convencionales.

Como resultado de la investigación realizada, se han identificado varias líneas de trabajo futuro que permitirían expandir y mejorar los resultados obtenidos. En primer lugar, se recomienda ampliar el alcance de la experimentación numérica

para incluir una mayor diversidad de problemas de optimización binivel. Esta expansión permitiría validar la robustez y versatilidad del algoritmo propuesto en diferentes contextos y escenarios de aplicación.

En segunda instancia, se sugiere profundizar en la investigación sobre la implementación del algoritmo desarrollado como criterio de parada en nuevos métodos de optimización binivel. Esta línea de investigación podría contribuir significativamente al desarrollo de algoritmos más eficientes y confiables, mejorando la capacidad de detectar y verificar puntos estacionarios durante el proceso de optimización.

Finalmente, se propone el desarrollo de una interfaz gráfica más intuitiva y funcional que facilite la generación automática de puntos según el tipo de estacionariedad requerida. Esta mejora en la usabilidad del software permitiría que usuarios con diferentes niveles de experiencia puedan aprovechar las capacidades del algoritmo de manera más efectiva, ampliando así su aplicabilidad práctica en diversos campos de estudio.

BIBLIOGRAFíA

Aussel, D., Bendotti, P., & Pistek, M. (2017). Nash equilibrium in a pay-as-bid electricity market Part 2 - best response of a producer. *Optimization*, *66*, 1027-1053. https://api.semanticscholar.org/CorpusID:18648572

Aussel, D., Cervinka, M., & Marechal, M. (2016). Deregulated electricity markets with thermal losses and production bounds: models and optimality conditions. *RAIRO Oper. Res.*, *50*, 19-38. https://api.semanticscholar.org/CorpusID:41625879

Bard, J. F. (1991). Some properties of the bilevel programming problem. *Journal of Optimization Theory and Applications*, *68*(2), 371-378. https://doi.org/10.1007/BF00941574

Bhavsar, N., & Verma, M. (2021). A subsidy policy to managing hazmat risk in railroad transportation network. *Eur. J. Oper. Res.*, *300*, 633-646. https://api.semanticscholar.org/CorpusID:238706367

Bouza-Allende, G., Aussel, D., Dempe, S., & Lepaul, S. (2021). Genericity Analysis of Multi-Leader-Disjoint-Followers Game. *SIAM J. Optim.*, *31*, 2055-2079. https://api.semanticscholar.org/CorpusID:238717899

Bouza-Allende, G., & Still, G. (2012). Solving bilevel programs with the KKT-approach. *Mathematical Programming*, *138*, 309-332. https://api.semanticscholar.org/CorpusID:18500519

Caselli, G., Iori, M., & Ljubi'c, I. (2024). Bilevel optimization with sustainability perspective: a survey on applications. https://api.semanticscholar.org/CorpusID:270379993

Cerulli, M. (2021, diciembre). *Bilevel optimization and applications* [Tesis doctoral].

Chkwon. (2025). Complementarity.jl. https://github.com/chkwon/Complementarity.jl

Dempe, S., & Freiberg, T. (2003). Annotated Bibliography on Bilevel Programming and Mathematical Programs with Equilibrium Constraints. *Optimization*, *52*. https://doi.org/10.1080/0233193031000149894

Dempe, S., & Zemkoho, A. (2020a). *Bilevel Optimization: Advances and Next Challenges*.

Dempe, S., & Zemkoho, A. (2020b). *Bilevel Optimization: Advances and Next Challenges*.

Flegel, M. L., & Kanzow, C. (2003). A Fritz John Approach to First Order Optimality Conditions for Mathematical Programs with Equilibrium Constraints. *Optimization*, *52*, 277-286. https://api.semanticscholar.org/CorpusID:26882450

Floudas, C. A. (1999). Handbook of Test Problems in Local and Global Optimization. https://api.semanticscholar.org/CorpusID:117898119

Floudas, C. A., & Pardalos, P. M. (1990). A Collection of Test Problems for Constrained Global Optimization Algorithms. *Lecture Notes in Computer Science*. https://api.semanticscholar.org/CorpusID:139191

Garcia, J. D., Bodin, G., & Street, A. (2022). BilevelJuMP.jl: Modeling and Solving Bilevel Optimization in Julia. *ArXiv*, *abs/2205.02307*. https://api.semanticscholar.org/CorpusID:248524800

Gu, H., Li, Y., Yu, J., Wu, C., Song, T., & Xu, J. (2020). Bi-level optimal low-carbon economic dispatch for an industrial park with consideration of multi-energy price incentives. *Applied Energy*, *262*, 114276. https://api.semanticscholar.org/CorpusID:213998633

Jeroslow, R. G. (1985). The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, *32*, 146-164. https://api.semanticscholar.org/CorpusID:39987722

JuliaLang. (2025). *Julia Documentation*. https://docs.julialang.org/en/v1/

Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., & Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*. https://doi.org/10.1007/s12532-023-00239-3

Ramos, M. A., Boix, M., Aussel, D., Montastruc, L., & Domenech, S. (2016). Water integration in eco-industrial parks using a multi-leader-follower approach. *Comput. Chem. Eng.*, *87*, 190-207. https://api.semanticscholar.org/CorpusID:26463725

Ramos, M. A., Rocafull, M., Boix, M., Aussel, D., Montastruc, L., & Domenech, S. (2018). Utility network optimization in eco-industrial parks by a multi-leader follower game methodology. *Comput. Chem. Eng.*, *112*, 132-153. https://api.semanticscholar.org/CorpusID:4003323

Siddiqui, S., & Gabriel, S. (2012). An SOS1-Based Approach for Solving MPECs with a Natural Gas Market Application. *Networks and Spatial Economics*, *13*. https://doi.org/10.1007/s11067-012-9178-y

Sinha, A., Malo, P., & Deb, K. (2017). A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. *IEEE Transactions on Evolutionary Computation*, *22*, 276-295. https://api.semanticscholar.org/CorpusID:4626744

Zhou, S., Zemkoho, A. B., & Tin, A. (2018). BOLIB: Bilevel Optimization LIBrary of Test Problems. *Bilevel Optimization*. https://api.semanticscholar.org/CorpusID:119636540