Uso de redes neuronales artificiales para problemas de programación entera

Using artificial neural networks for integer programming problems

Julián Silva Rodríguez, Andrés Fernández Rosas

Palabras clave: investigación de operaciones, programación lineal entera, redes neuronales artificiales

Key words: operations research, integer linear programming, artificial neural networks

RESUMEN

En la investigación de operaciones, la resolución eficiente de modelos de programación entera representa un desafío computacional complejo debido a su naturaleza combinatoria. Ante las limitaciones de los métodos exactos, las redes neuronales artificiales emergen como un paradigma disruptivo capaz de optimizar decisiones algorítmicas mediante el aprendizaje de patrones estructurales. Este artículo presenta una propuesta metodológica para integrar redes neuronales artificiales en la solución de problemas de programación entera, con énfasis en programación lineal entera mixta. Se realizó una revisión de literatura (2016–2025) en Scopus y ScienceDirect para identificar enfoques que combinan aprendizaje automático con solucionadores exactos (branch-and-bound y branch-and-cut). La evidencia reciente reporta mejoras frente a heurísticas tradicionales en componentes críticos como ramificación, heurísticas primales y selección de nodo/variable. Con base en esta síntesis, se propone un esquema de representación del programación lineal entera mixta como grafo bipartito y un protocolo de evaluación centrado en tiempo de solución, tamaño del árbol y brecha primal–dual, demostrando ventajas comparativas en familias de instancias homogéneas.

ABSTRACT

In operations research, the efficient resolution of integer programming models represents a complex computational challenge due to its combinatorial nature. Given the limitations of exact methods, artificial neural networks emerge as a disruptive paradigm capable of optimizing algorithmic decisions through the learning of structural patterns. This article presents a methodological proposal to integrate artificial neural networks in the solution of integer programming problems, with an emphasis on mixed-integer linear programming. A literature review (2016–2025) was conducted in Scopus and ScienceDirect to identify approaches that combine machine learning with exact solvers (branch-and-bound and branch-and-cut). Recent evidence reports improvements over traditional heuristics in critical components such as branching, primal heuristics, and node/variable selection. Based on this synthesis, a representation scheme of mixed-integer linear programming as a bipartite graph and an evaluation protocol centered on solution time, tree size, and primal–dual gap are proposed, demonstrating comparative advantages in families of homogeneous instances.

INTRODUCIÓN

En la actualidad, las organizaciones operan en entornos de incertidumbre sistémica, caracterizados por una volatilidad donde las variables que inciden en el desempeño institucional resultan cada vez más difíciles de predecir y controlar (World Economic Forum [WEF], 2024). En este contexto, la dinámica competitiva se ve impulsada por fluctuaciones drásticas en la demanda y una evolución acelerada de las necesidades del mercado (Niebel y Freivalds, 2014), factores que se ven potenciados por la disrupción tecnológica y la interconectividad global. En la práctica, estas presiones incrementan la urgencia de implementar modelos de decisión robustos y reproducibles, especialmente en escenarios que requieren fundamentar decisiones discretas —de naturaleza binaria o entera— a nivel táctico y estratégico (Scavuzzo et al., 2024).

Debido a lo anterior, las organizaciones han fortalecido estrategias de mejora basadas en el análisis de procesos operativos y administrativos, con el fin de aumentar competitividad y rentabilidad (Sánchez et al., 2019). En problemas de decisión complejos, dichas estrategias suelen traducirse en modelos de optimización que integran restricciones reales —capacidad, presupuesto, normatividad— y decisiones discretas (Zhang et al., 2023). Una herramienta central para soportar decisiones óptimas es la Investigación de Operaciones (IO). Por medio de modelos como programación lineal (PL), programación entera (PE) y programación lineal entera mixta (PLEM, —*MILP*, *Mixed-Integer Linear Programming*—, se pueden obtener soluciones óptimas o certificadas para procesos de producción, logística, finanzas y planificación, entre otros (Ortiz y Caro, 2017; Scavuzzo et al., 2024).

La PE es particularmente útil cuando las variables representan decisiones discretas (seleccionar/no seleccionar, abrir/cerrar, asignar/no asignar). En términos computacionales, la PE y la PLEM son NP-difíciles —*non-deterministic polynomial-time hard*, comúnmente abreviado como *NP-hard*— y típicamente se resuelven con algoritmos de ramificación y acotamiento —*branch-and-bound*— y ramificación y corte —*branch-and-cut*—, los cuales suelen apoyarse en heurísticas para agilizar la búsqueda de soluciones en tiempos razonables (Deza y Khalil, 2023; Scavuzzo et al., 2024). Esto abre un espacio natural para que el aprendizaje automático construya heurísticas basadas en datos, complementando —no reemplazando— los métodos exactos (Bengio et al., 2021).

La PE optimiza una función objetivo bajo restricciones lineales, exigiendo que algunas o todas las variables tomen valores enteros. Su relevancia radica en que muchas decisiones reales son discretas —por ejemplo, selección de proyectos, localización, asignación, rutas—, y la formulación entera permite modelarlas de forma transparente y verificable (Zhang et al., 2023; Scavuzzo et al., 2024).

Por otra parte, las redes neuronales artificiales (RNA) son modelos de

aprendizaje que aproximan funciones complejas a partir de datos. En optimización combinatoria, las RNA se han usado para aprender componentes decisionales de los solucionadores —ramificación, selección de cortes, heurísticas primales—, explotando regularidades en distribuciones de instancias y reduciendo tiempos de cómputo en escenarios repetitivos (Bengio et al., 2021; Scavuzzo et al., 2024).

De acuerdo con lo anterior, este articulo tiene como objetivo proponer, con base en evidencia reciente, un marco metodológico para representar y resolver problemas de PE mediante RNA integradas a un solucionador exacto, y establecer criterios comparativos —tiempo, tamaño del árbol, brecha primal–dual— frente a métodos tradicionales y enfoques alternativos.

En consecuencia, el presente artículo consolida la literatura reciente sobre aprendizaje para programación entera y propone una aproximación conceptual para representar modelos de PE mediante una arquitectura de RNA, orientada a mejorar decisiones internas del solucionador —por ejemplo, ramificación y construcción de soluciones primales—y no únicamente a "predecir" una solución final. El aporte se complementa con un esquema de validación comparativa frente a heurísticas clásicas y enfoques de referencia reportados en la literatura (Gasse et al., 2019; Nair et al., 2021; Du et al., 2025).

Marco Teórico

Investigación de operaciones

El uso de la Investigación de operaciones para la toma de decisiones en las empresas, ha venido aumentando con el tiempo, apoyando a las organizaciones en la búsqueda de soluciones óptimas para los problemas organizacionales, teniendo en cuenta con los recursos y capacidad con la que cuentan. En consecuencia, la investigación de operaciones ofrece la opción de que por medio de modelos matemáticos se represente el comportamiento de un sistema y se logren tomar decisiones (Winston y Goldberg, 2005).

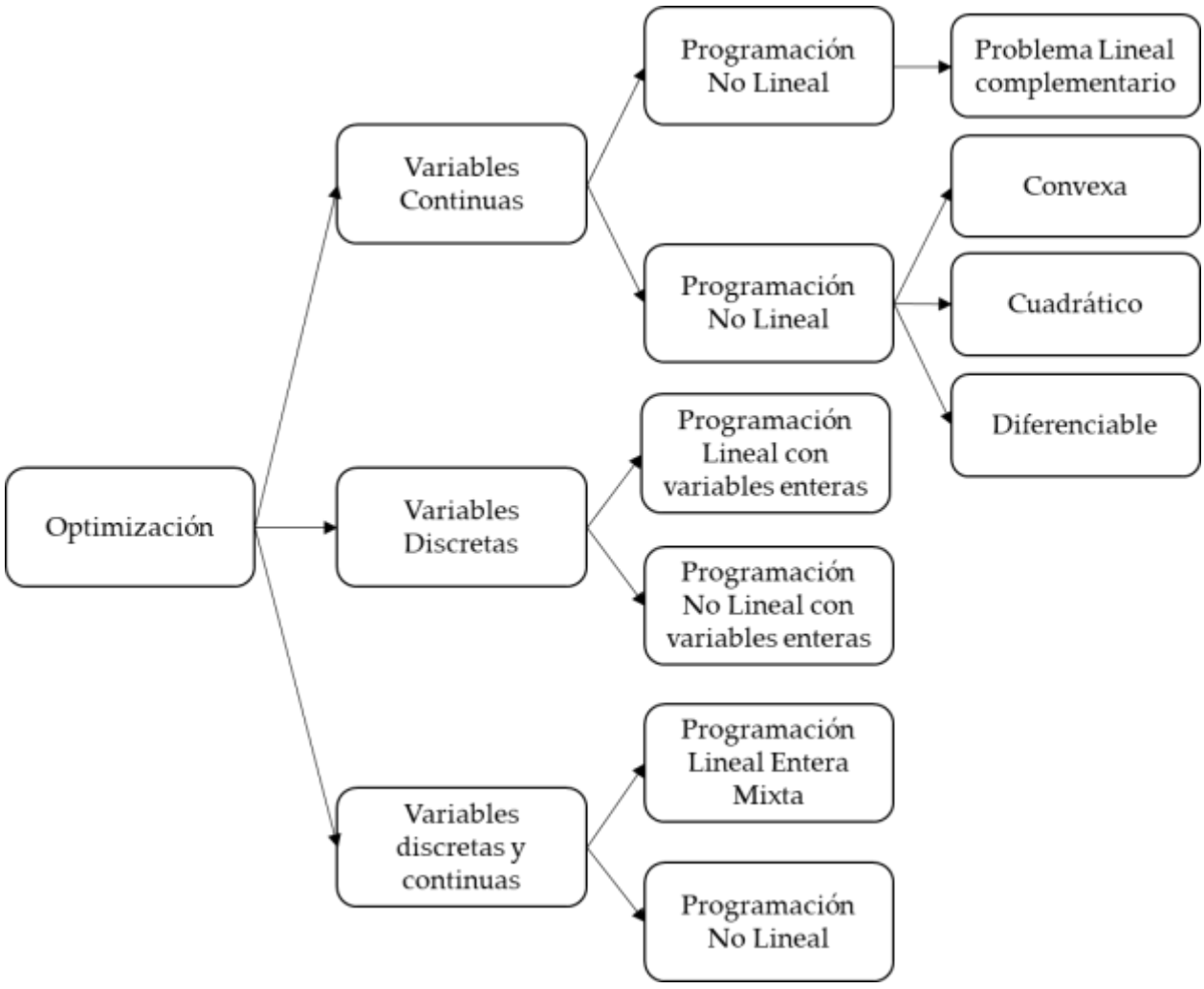
Varios autores han centrado sus esfuerzos en lograr definiciones de la investigación de operaciones y como aporta a la toma de decisiones de las organizaciones. Autores como Taha (2024), Prawda (2004), Hillier y Lieberman (2020) coinciden en afirmar que la Investigación de Operaciones es una disciplina o una rama de las matemáticas que por medio del uso de métodos y herramientas analíticas y cuantitativos, realiza investigación aplicando el método científico a los diferentes problemas o escenarios de las organizaciones, con el objetivo de lograr soluciones optimas, teniendo en cuenta las limitaciones que se pueden tener en los recursos que intervienen en el funcionamiento normal de una organización.

La investigación de operaciones utiliza modelos de optimización, los cuales se pueden clasificar inicialmente en los métodos clásicos, los cuales son aquellos

que buscar encontrar soluciones optimas donde podemos entrar herramientas como programación lineal, programación entera, programación entera mixta, programación estocástica y programación dinámica. En segunda instancia podemos encontrar los métodos heurísticos y metaheurísticos, cuyo objetivo son encontrar soluciones factibles o globales, dentro de estos métodos se pueden encontrar herramientas como agente viajero, problema del cartero chino, búsqueda tabú, colonia de hormigas, entre otros (Linares et al., 2001).

Según Pérez Peña (2019), los modelos de optimización se pueden clasificar también de acuerdo con el tipo de variables que se manejen, lo cual podría ser variables continuas, variables discretas y modelos que funcionan para los dos tipos de variables. En la figura 1 se puede observar dicha clasificación.

Figura 1. *Clasificación modelos de optimización*



Nota. Adaptado de *Introducción a los modelos de optimización*, por R. Pérez Peña, 2019, UniPiloto.

Dentro de las herramientas de la investigación de operaciones se encuentra la programación entera, que permite formular decisiones discretas mediante variables enteras y restricciones lineales. En su forma más usada, la PLEM combina variables continuas e enteras, lo que amplía su aplicabilidad industrial —por ejemplo, planeación, asignación, ruteo, scheduling—. Debido a su complejidad (NP-dificultad), los solucionadores modernos se basan en *branch-and-bound/branch-and-cut* y en múltiples heurísticas cuyo diseño es, en buena medida, empírico (Zhang et al., 2023; Scavuzzo et al., 2024).

A modo de ejemplo, una de las posibles aplicaciones de la PE es la decisión de si invertir o no en un proyecto, donde las decisiones se representan por medio de variables binarias. En la figura 2, se observa un ejemplo de tomar este tipo de decisiones donde la función objetivo está dada en la maximización de las ganancias y las restricciones se plantean en torno a los fondos posibles.

Figura 2. *Ejemplo modelo Programación Entera*

$$x_j = \begin{cases} 1, \text{ si se selecciona el proyecto } j \\ 0, \text{ si no se selecciona el proyecto } j \end{cases}$$

El modelo de PLE es

$$\text{Maximizar } z = 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5$$

Sujeto a

$$\begin{aligned} 5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 &\leq 25 \\ x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 &\leq 25 \\ 8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 &\leq 25 \\ x_1, x_2, x_3, x_4, x_5 &= (0,1) \end{aligned}$$

Nota. Tomado de *Investigación de operaciones* (11.ª ed.), por H. A. Taha, 2024, Pearson.

Como se mencionaba anteriormente, la programación entera se puede dividir en programación entera pura y programación entera mixta. Algunas aplicaciones de la programación entera pura son:

Problema de asignación: en este problema se busca asignar una cantidad de recursos limitados para realizar diferentes funciones, lo cual considera que dichos recursos no son fraccionables. Los ejemplos más típicos en este tipo de problema es la asignación de personal a cierta cantidad de cargos, asignación de profesionales a una cantidad de proyectos o tareas, entre otros (González Ariza y García Llinás, 2015)

Problema de la mochila: este problema también conocido como Knapsack Problem, hace referencia a un problema cotidiano donde un viajero o un excursionista debe preparar su maleta o su mochila, donde tiene unas posibilidades de elementos a cargar, pero hay que tener en cuenta que la maleta tiene una capacidad de carga. El objetivo final del problema es escoger los elementos a cargar que maximice la utilización de la capacidad de carga y el mayor número de elementos posibles (Silva y Torres, 2014).

Problema de Planificación Forestal: este problema esta directamente relacionado no con decisiones de si o no, si no hace referencia a la planeación de talas de árboles y planificación de números de trabajadores que se necesitan. También incluye variables referentes a inventarios, compras y contratación y despido de trabajadores. Todas las variables inmersas en este problema son enteras (Vargas Suárez, et al., 2004).

Igualmente existen diversas aplicaciones en la programación entera mixta, las cuales pueden ser:

Problema de localización y transporte: en los problemas clásicos de transporte donde las variables hacen referencia a cantidades a transportar las cuales pueden ser discretas o continuas, donde puede haber una variación frente la incursión de variables binarias con el fin de decidir la apertura de plantas de producción, centros de distribución o cualquier localización que necesite una empresa, donde las decisiones son de si abrir o no una localización (Salas, 2009)

Problema de incursión de costos fijos: esta aplicación se da en escenarios donde se asumen costos fijos por utilización de servicios como puede ser luz, agua, gas, teléfono, etc, donde el consumidor debe asumir un costo fijo independiente de la cantidad que consuma y por ende el modelo debe sumar este costo al tomar la decisión de si toma el servicio o no (Hillier y Lieberman, 2020).

Redes Neuronales Artificiales

La teoría de las redes neuronales tiene como antecedente el estudio del sistema nervioso y, en su versión moderna, se consolida como un enfoque de aprendizaje profundo para aproximación de funciones y toma de decisiones a partir de datos. En optimización, el interés no es "imitar el cerebro", sino aprender reglas de decisión que sustituyan o complementen heurísticas manuales (Bengio et al., 2021).

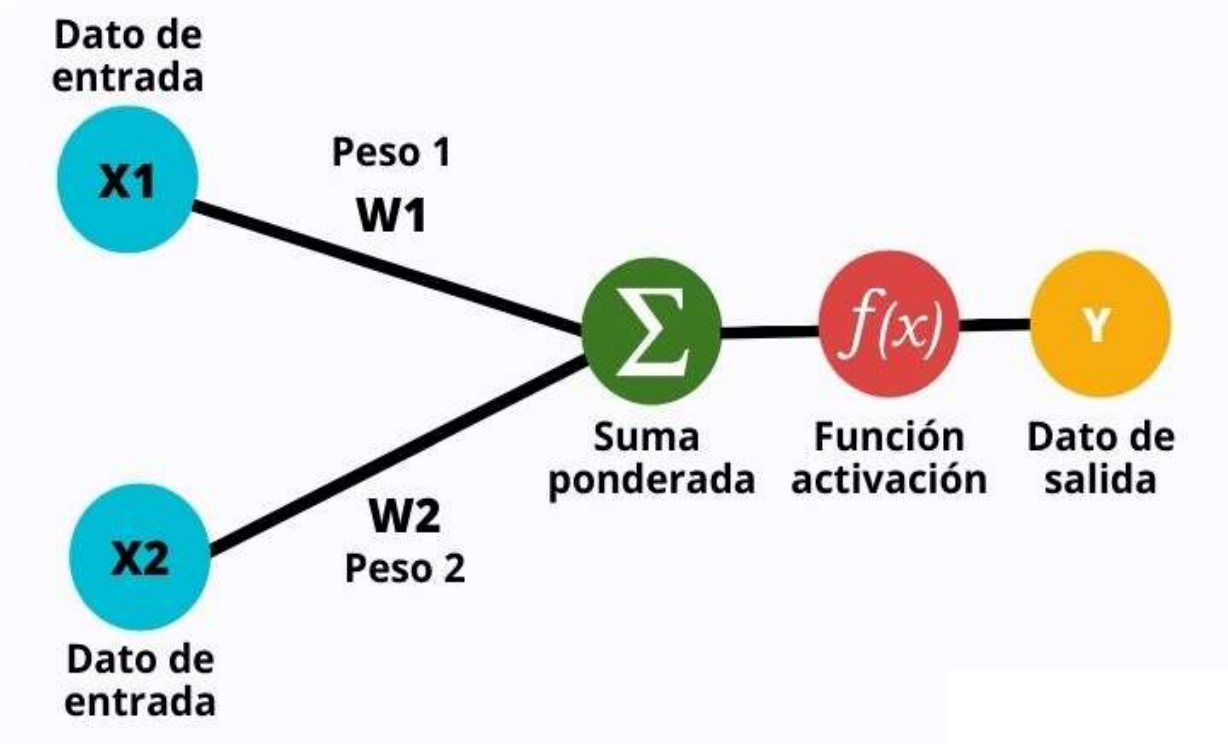
En consecuencia, las RNA se implementan como capas de neuronas artificiales conectadas por pesos, que se ajustan mediante entrenamiento para minimizar una función de pérdida. Para problemas de PE, la representación como grafo (variables–restricciones) ha sido especialmente efectiva porque preserva estructura y permite generalización a instancias de distinto tamaño (Gasse et al., 2019; Du et al., 2025).

Los componentes de procesamiento en las RNA incluyen: entradas —*features*—, pesos, funciones de activación y una regla de actualización —por ejemplo, Adam—. En el contexto de PE, las salidas típicas no son necesariamente una solución completa,

sino decisiones internas del solucionador: elegir la variable de ramificación, ordenar nodos, seleccionar cortes o proponer asignaciones parciales (Deza y Khalil, 2023; Scavuzzo et al., 2024).

El perceptrón es el bloque conceptual básico, pero en aplicaciones modernas se utilizan arquitecturas profundas (PLEM, GNN, *transformers*) según la estructura del dato. Para PE/PLEM, la evidencia reciente favorece modelos basados en grafos por su alineación natural con la matriz de restricciones y su desempeño al aprender políticas de ramificación y otras heurísticas (Gasse et al., 2019; Gupta et al., 2020).

Figura 3. *Estructura de un perceptrón*



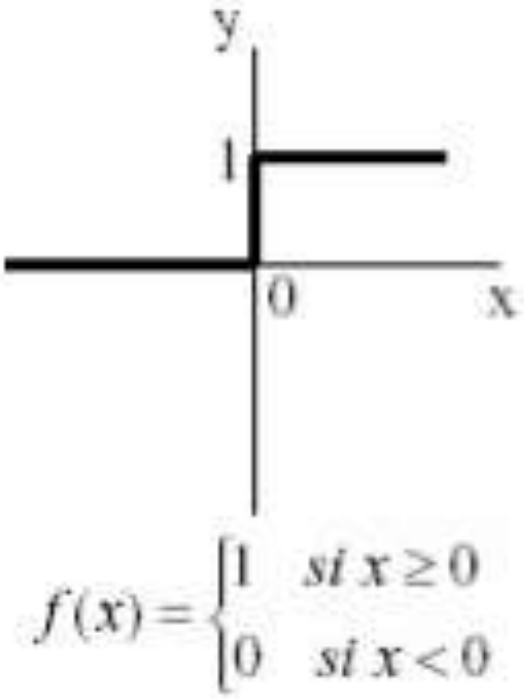
Nota. Tomado de *Las redes neuronales artificiales*, por R. F. López y J. M. F. Fernández, 2008, Netbiblo.

En la estructura del perceptrón los X_i representan los valores de entradas o las capas de entrada, las cuales se pueden considerar como características de las clases que se pretender clasificar, W_i son los pesos porcentuales de entradas que son multiplicados posteriormente con los datos de entrada.

Las funciones de activación permiten a la red neuronal controlar el proceso de aprendizaje. Según López y Fernández (2008), dentro de las funciones de activación se pueden encontrar:

Función escalón: con esta función se logra el envío de señales de salida o respuestas binarias de 1 o 0 y el valor que tome dependerá de superar cierto valor parametrizado (ver figura 4)

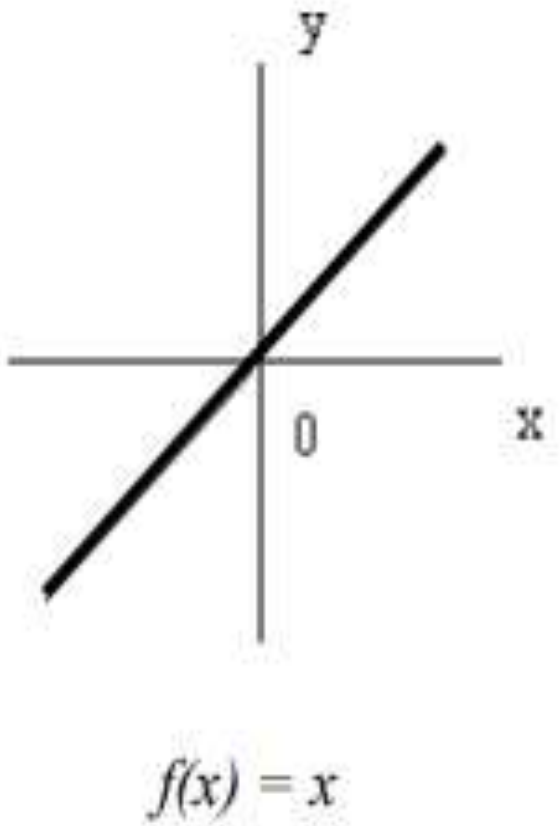
Figura 4. *Función escalón*



Nota. Tomado de *Las redes neuronales artificiales*, por R. F. López y J. M. F. Fernández, 2008, Netbiblo.

Función lineal: esta función en realidad es una identidad de las variables de entrada y equivaldría a la misma variable de salida. (ver figura 5)

Figura 5. *Función lineal*

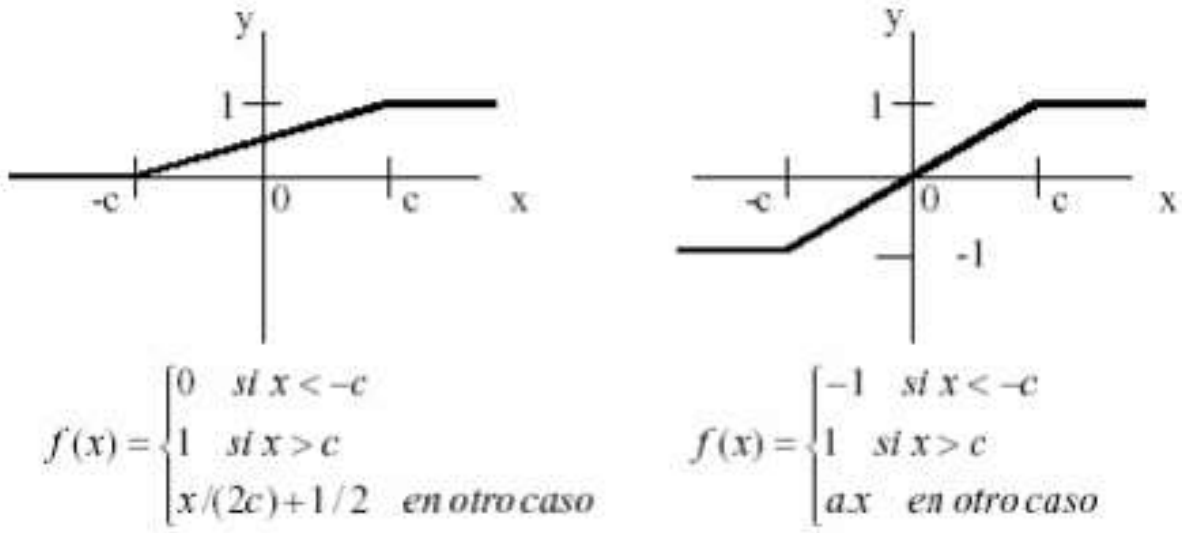


Nota. Tomado de *Las redes neuronales artificiales*, por R. F. López y J. M. F. Fernández, 2008, Netbiblo.

Función mixta: esta función combina las dos anteriores, donde si la activación es menor a un parámetro la salida o resultado es 0 pero si es mayor la salida es 1 y si la activación esta entre los limites de los parámetros va a tomar valores de acuerdo con una función lineal (ver figura 6).

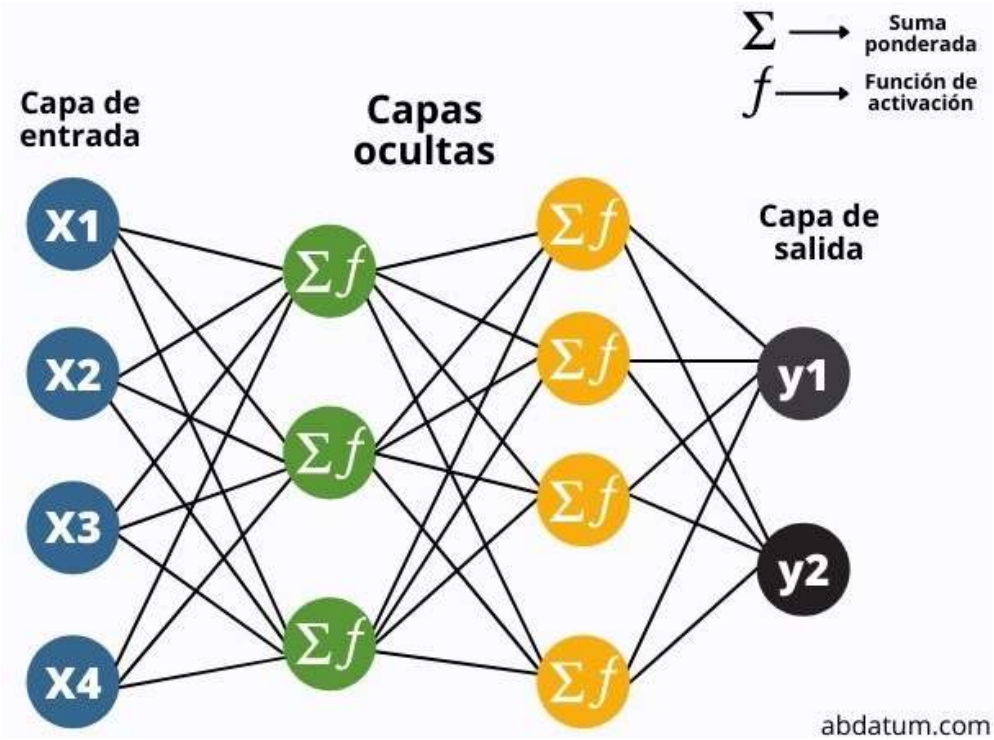
Finalmente se puede decir que una red neuronal es la combinación de varios perceptrones. En la figura 7 se puede observar la estructura de la red neuronal artificial.

Figura 6. *Función mixta*



Nota. Tomado de *Las redes neuronales artificiales*, por R. F. López y J. M. F. Fernández, 2008, Netbiblo.

Figura 7. *Estructura red neuronal artificial*



Nota. Tomado de *Las redes neuronales artificiales*, por R. F. López y J. M. F. Fernández, 2008, Netbiblo.

METODOLOGÍA

La metodología se estructuró en dos frentes: (i) revisión de literatura para identificar enfoques de RNA aplicados a PE/PLEM y (ii) síntesis metodológica para construir la propuesta.

Revisión: se consultaron Scopus y ScienceDirect (periodo 2016–2025) con combinaciones de palabras clave: "*mixed-integer programming*", "*integer programming*", "*branch-and-bound*", "*branch-and-cut*", "*neural network*", "*graph neural network*", "*learning to branch*", "*cut selection*", "*neural diving*". Se aplicaron criterios de inclusión: (a) artículo revisado por pares o preprint ampliamente citado; (b) evaluación empírica contra un solver base —por ejemplo, SCIP— o heurísticas estándar; (c) descripción reproducible del método. Se excluyeron trabajos sin comparación o sin métricas de desempeño.

Síntesis: los aportes se organizaron por tarea dentro del solver (ramificación, heurísticas primales, cortes, selección de nodo/variable, configuración) para derivar un flujo de diseño: representación del problema → generación de datos → arquitectura → entrenamiento (imitación/RL/supervisado) → integración con solver → evaluación.

Con base en la síntesis, la propuesta sigue un enfoque "*solver-in-the-loop*": la RNA no reemplaza el método exacto, sino que aprende a tomar decisiones locales de alta influencia. Para ello, se recomienda codificar el PLEM como grafo bipartito (variables–restricciones) y entrenar la política a partir de demostraciones —imitación de *strong branching*— o a partir de un Proceso de Decisión de Markov con estructura de árbol —*Tree-Markov Decision Process, Tree-MDP*— según la disponibilidad de expertos y datos (Gasse et al., 2019; Etheve et al., 2020; Scavuzzo et al., 2024).

RESULTADOS

Como se ha podido evidenciar, los problemas de programación entera por su estructura pueden representarse mediante RNA; sin embargo, la contribución clave en la literatura reciente es integrar RNA para mejorar decisiones internas de solucionadores exactos. A continuación, se sintetiza evidencia de ventajas reportadas frente a heurísticas tradicionales.

Evidencia empírica (selección de trabajos): (1) Gasse et al. (2019) aprenden una política de ramificación con GNN —*Graph Neural Networks*— que imita *strong branching*, reduciendo el tamaño del árbol y tiempos en *benchmarks* difíciles. (2) Nair et al. (2021) introducen *Neural Diving* —heurística primal— y *Neural Branching*, reportando mejoras de 2×–10× (y mayores en casos específicos) frente a *SCIP* —*Solving Constraint Integer Programs*— en conjuntos reales y *MIPLIB* —*Mixed Integer Programming Library*—. (3) Deza y Khalil

(2023) muestran que la selección de cortes es un punto crítico y revisan resultados donde *Machine Learning (ML)* mejora la calidad/eficiencia del *branch-and-cut*. (4) Du et al. (2025) proponen una política conjunta de selección de nodo y variable, superando reglas por defecto de SCIP en tiempo y tamaño del árbol. (5) Scavuzzo et al. (2024) sistematizan métricas, representaciones y buenas prácticas para evaluar ML en *Branch-and-Bound* (ver figura 8).

Figura 8. *Estructura modelo de PE*

$$MIN\ O\ MAX\ Z = C_{11}X_{11} + C_{21}X_{21} + C_{31}X_{31} + \ldots + C_{ij}X_{ij}$$

s.a.

$$a_{11}X_{11} + a_{12}X_{12} + a_{13}X_{13} + \ldots + a_{1j}X_{1j} = b_1$$

$$a_{21}X_{21} + a_{22}X_{22} + a_{23}X_{23} + \ldots + a_{2j}X_{2j} = b_2$$

$$a_{11}X_{11} + a_{21}X_{21} + a_{31}X_{31} + \ldots + a_{i1}X_{i1} = d_1$$

$$a_{12}X_{12} + a_{22}X_{22} + a_{32}X_{32} + \ldots + a_{i2}X_{i2} = d_2$$

$$X_{ij} = [0,1]$$

Los modelos de programación entera podrán tener objetivos de maximización de beneficios, ganancias, rendimientos, entre otros y podrán tener minimización de costos, tiempos, inversión entre otros, donde los C_{ij} representan los coeficientes que acompañan a las variables de acuerdo con el objetivo que se desea lograr. Los parámetros aij representan los coeficientes que acompañan a las variables en las restricciones los cuales pueden ser inversiones, tiempos, pesos, entre otros o en el modelo de asignación simplemente será el valor de 1. Finalmente, los parámetros *bi* y *dj* son condiciones que se deben cumplir en las restricciones.

El modelo anterior presentado se puede representar por medio de matrices y vectores teniendo en cuenta su estructura dentro de la formulación de la siguiente forma (ver figura 9).

Como se puede observar tanto en la primera ecuación como en la segunda se multiplica por la misma matriz de variables, por ende se podrá lograr una formulación general del problema de la siguiente forma (ver figura 10).

Figura 9. *Representación Matricial modelo de PE*

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1j} & 1 & 0 & \cdots & 0 & 0 & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2j} & 0 & 1 & \cdots & 0 & 0 & b_2 \\ a_{11} & a_{21} & a_{31} & \cdots & a_{i1} & 0 & 0 & 1 & 0 & 0 & d_1 \\ a_{12} & a_{22} & a_{32} & \cdots & a_{i2} & 0 & 0 & 0 & 1 & 0 & d_2 \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1j} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2j} \\ X_{11} & X_{21} & X_{31} & \dots & X_{i1} \\ X_{12} & X_{22} & X_{32} & & X_{i2} \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1j} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2j} \\ X_{11} & X_{21} & X_{31} & \dots & X_{i1} \\ X_{12} & X_{22} & X_{32} & & X_{i2} \end{bmatrix}$$

$$[C_{11} \quad C_{12} \quad C_{13} \cdots \quad C_{ij} \quad 0 \quad \cdots \quad 0 \quad Z] \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1j} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2j} \\ X_{11} & X_{21} & X_{31} & \dots & X_{i1} \\ X_{12} & X_{22} & X_{32} & & X_{i2} \end{bmatrix} = [0 \quad \cdots \quad 0]$$

Figura 10. *Formulación general matricial*

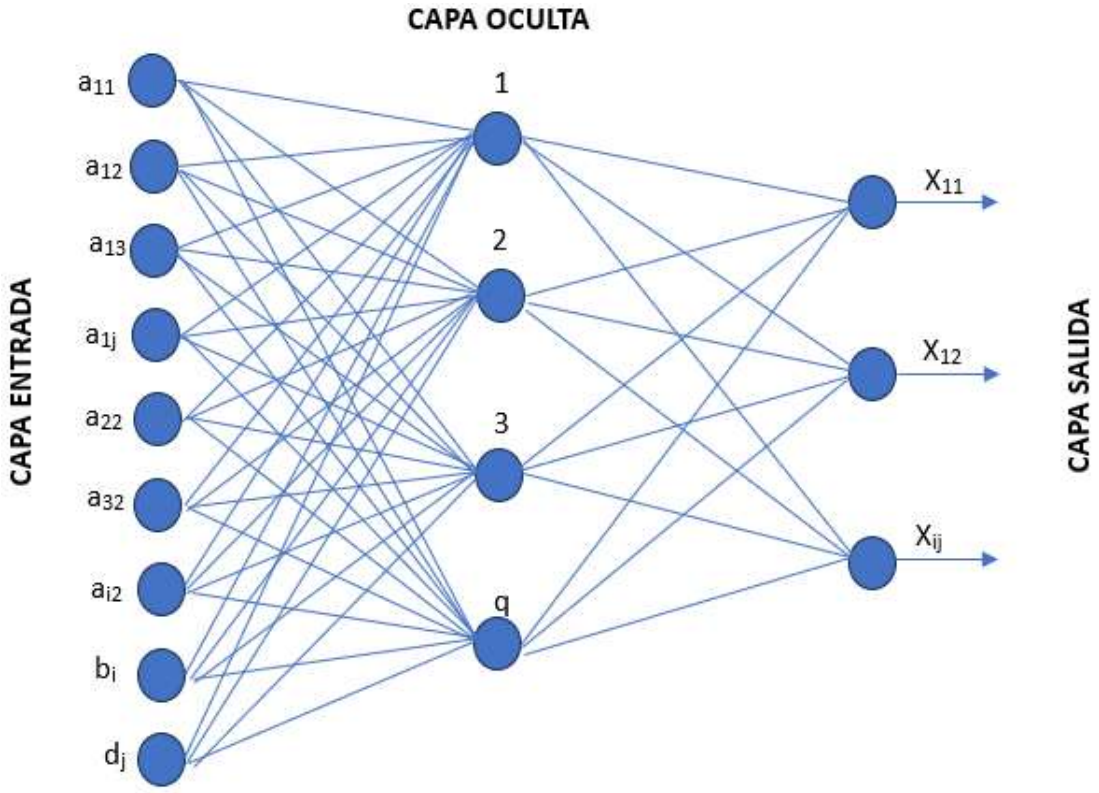
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1j} & 1 & 0 & \cdots & 0 & 0 & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2j} & 0 & 1 & \cdots & 0 & 0 & b_2 \\ a_{11} & a_{21} & a_{31} & \cdots & a_{i1} & 0 & 0 & 1 & 0 & 0 & d_1 \\ a_{12} & a_{22} & a_{32} & \cdots & a_{i2} & 0 & 0 & 0 & 1 & 0 & d_2 \\ C_{11} & C_{12} & C_{13} & \cdots & C_{ij} & 0 & 0 & 0 & \dots & 0 & Z \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1j} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2j} \\ X_{11} & X_{21} & X_{31} & \dots & X_{i1} \\ X_{12} & X_{22} & X_{32} & & X_{i2} \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \dots & X_{1j} \\ X_{21} & X_{22} & X_{23} & \cdots & X_{2j} \\ X_{11} & X_{21} & X_{31} & \dots & X_{i1} \\ X_{12} & X_{22} & X_{32} & & X_{i2} \end{bmatrix}$$

Con esta forma general de representar el problema por medio de matrices, se podrá comenzar a diseñar la posible configuración de la red, lo cual define la cantidad de neuronas que se necesitan en la capa de entrada y las salidas que vamos a obtener, que son las variables a las cuales finalmente se les quiere hallar un valor o un resultado óptimo. Los valores de entrada para la red serán todo los coeficientes o parámetros que se tienen definidos en la primera matriz.

En la figura 11 se puede evidenciar la estructura de la red neuronal artificial para un problema de programación entera. Si bien se ha hecho énfasis en un problema de programación entera tomando como base el modelo de asignación, de igual forma se puede aplicar para cualquier problema de programación entera pura o programación entera mixta, teniendo en cuenta las modificaciones a hacer dependiendo la aplicación.

Finalmente, la red neuronal artificial que representa el modelo de programación entera se podrá implementar y desarrollar en software como R o Phyton.

Figura 11. *Estructura red neuronal artificial para problema de programación entera*



Análisis y discusión

Los métodos clásicos (*branch-and-bound/branch-and-cut*) ofrecen optimalidad certificada, pero su desempeño depende de heurísticas y parámetros diseñados manualmente. Las metaheurísticas —por ejemplo, búsqueda tabú, colonia de hormigas— priorizan rapidez y factibilidad, pero no garantizan optimalidad. Las propuestas basadas en RNA se ubican en un punto intermedio: conservan garantías del solver exacto (—si la RNA guía decisiones pero no elimina el árbol— y buscan reducir costo computacional aprendiendo patrones de instancias repetidas (Scavuzzo et al., 2024).

Ventajas reportadas: (i) reducción del tamaño del árbol y del tiempo en distribuciones específicas; (ii) mejores soluciones primales tempranas (menor brecha primal–dual) a partir de heurísticas aprendidas; (iii) menor dependencia de ajuste manual cuando existe una familia estable de instancias. ***Limitaciones***: (a) necesidad de datos y riesgo de no generalizar fuera de la distribución de entrenamiento; (b) costo de entrenamiento e infraestructura; (c) interpretabilidad limitada y sensibilidad a cambios de presolve/configuración (Bengio et al., 2021; Scavuzzo et al., 2024).

En la tabla 1, se presenta un resumen de la idea central, las fortalezar y las limitaciones típicas de cada enfoque.

Tabla 13. *Comparación resumen de los enfoques*

Enfoque	Idea central	Fortalezas	Limitaciones típicas
Solver exacto (ByB/ByC)	Búsqueda + relajaciones LP + cortes	Optimalidad certificada; robustez	Tiempo elevado; heurísticas manuales
Heurísticas/ metaheurísticas	Exploración guiada del espacio	Rapidez; simplicidad	Sin garantía de optimalidad; sensibilidad a parámetros
RNA dentro del solver	Aprender decisiones (ramificación, cortes, heurística primal)	Acelera tareas críticas; mantiene certificación si se integra bien	Requiere datos; generalización; costo de entrenamiento

CONCLUSIONES

La revisión de la literatura reciente confirma que la integración de Redes Neuronales Artificiales (RNA) no busca sustituir los métodos exactos, sino potenciar su eficiencia interna. Al delegar decisiones críticas —como la ramificación (*branching*), la selección de cortes y las heurísticas primales— a modelos de aprendizaje, se logra un punto de equilibrio entre la optimalidad garantizada de los métodos clásicos y la velocidad de las metaheurísticas (Gasse et al., 2019; Nair et al., 2021; Deza y Khalil, 2023).

En este contexto, la propuesta metodológica de representar los problemas de Programación Lineal Entera Mixta (PLEM) como grafos bipartitos procesados mediante Redes Neuronales de Grafos (GNN) resulta altamente coherente con los avances de mayor desempeño (Scavuzzo et al., 2024). Este enfoque, modelado bajo un Proceso de Decisión de Markov en Árbol (*tree-MDP*) e integrado bajo el esquema *solver-in-the-loop*, permite que el sistema aprenda patrones estructurales de familias de instancias específicas, superando las reglas por defecto de motores como *SCIP — Solving Constraint Integer Programs*— (Du et al., 2025).

Para validar estas ventajas frente a los métodos tradicionales, se concluye que la evaluación experimental debe ser multidimensional, reportando obligatoriamente: (i) el tiempo total de solución, (ii) la reducción del tamaño del árbol de búsqueda (ByB), (iii) la evolución de la brecha (gap) primal-dual y (iv) la capacidad de generalización ante instancias fuera de la muestra de entrenamiento.

Finalmente, como trabajo futuro, se plantea la implementación de un pipeline completo para familias de instancias objetivo, priorizando la reproducibilidad mediante el uso de librerías estándar como *MIPLIB*, protocolos de semillas (seeds) controladas y configuraciones de presolve transparentes.

REFERENCIAS

Bengio, Y., Lodi, A., y Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2), 405–421. https://doi.org/10.1016/j.ejor.2020.07.063

Deza, A., y Khalil, E. B. (2023). Machine Learning for Cutting Planes in Integer Programming: A Survey. *Proceedings of IJCAI-23*, 6592–6600. https://doi.org/10.24963/ijcai.2023/739

Du, S., Tong, J., y Fan, W. (2025). Learning efficient branch-and-bound for solving Mixed Integer Linear Programs. *Applied Soft Computing*, 172, 112863. https://doi.org/10.1016/j.asoc.2025.112863

Etheve, M., Alès, Z., Bissuel, C., Juan, O., y Kedad-Sidhoum, S. (2020). *Reinforcement Learning for Variable Selection in a Branch and Bound Algorithm*. arXiv:2005.10026. https://doi.org/10.48550/arXiv.2005.10026

Hillier, F. S., y Lieberman, G. J. (2020). *Introduction to Operations Research* (11th ed.). McGraw-Hill Higher Education.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., y Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*. https://doi.org/10.48550/arXiv.1906.01629

Gupta, P., Gasse, M., Khalil, E. B., Kumar, M. P., Lodi, A., y Bengio, Y. (2020). Hybrid models for learning to branch. *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv. https://doi.org/10.48550/arXiv.2006.15212

González Ariza, Á. L., y García Llinás, G. (2015). *Manual práctico de investigacion de Operaciones I* (4ta edición). Universidad del Norte.

Llinás, R. R. (2003). *El cerebro y el mito del yo: el papel de las neuronas en el pensamiento y el comportamiento humanos*. Norma.

Linares, P., Ramos, A., Sánchez, P., Sarabia, A., y Vitoriano, B. (2001). *Modelos matemáticos de optimización*. Universidad Pontificia Comillas.

López, R. F., y Fernández, J. M. F. (2008). *Las redes neuronales artificiales*. Netbiblo.

Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Vinyals, O., et al. (2021). *Solving Mixed Integer Programs Using Neural Networks*. arXiv:2012.13349. https://doi.org/10.48550/arXiv.2012.13349

Niebel, B. W., y Freivalds, A. (2014). *Ingeniería Industrial de Niebel: métodos, estándares y diseño del trabajo*. McGraw Hill.

Ortiz, J. D. C., y Caro, G. V. (2017). *Introducción a la investigación de operaciones*. Universidad de Sevilla, Sección de Publicaciones ETSI.

Pérez Peña, R. (2019). *Introducción a los modelos de optimización*. UniPiloto.

Prawda, J. (2004). *Métodos y modelos de investigación de operaciones. Vol. 1. Modelos determinísticos*. Limusa.

Salas, H. G. (2009). *Programación lineal aplicada*. Ecoe Ediciones.

Salas, R. (2004). *Redes neuronales artificiales* [Manuscrito no publicado]. Departamento de Computación, Universidad de Valparaíso.

Sánchez, M. M., Maggi, M. T., y Paredes, M. C. (2019). Resistencia al cambio en las organizaciones: propuesta para minimizarlo. *Palermo Business Review*, (19), 39-53. https://www.palermo.edu/economicas/cbrs/pdf/pbr19/PBR_19_02.pdf

Scavuzzo, L., Aardal, K., Lodi, A., y Yorke-Smith, N. (2024). Machine learning augmented branch and bound for mixed integer linear programming. *Mathematical Programming, Series B*. https://doi.org/10.1007/s10107-024-02130-y

Silva, B., y Torres, L. M. (2014). Acerca de una versión dinámica del problema de la mochila. *Revista Politécnica*, 34(1), 142-142. https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/253

Taha, H. A. (2024). *Investigación de operaciones* (11.ª ed.). Pearson.

Vargas Suárez, L., Cano Robles, I., y Ríos Mercado, R. Z. (2004). Aplicación de la programación lineal en el sector forestal. *Ingenierías*, 7(24), 19-26. http://eprints.uanl.mx/10144/1/24_aplicacion_de_la_programacion.pdf

Winston, W. L., y Goldberg, J. B. (2005). Investigación de operaciones: aplicaciones y algoritmos (Vol. 4). Thomson.

World Economic Forum. (2024). *The Global Risks Report 2024 (19th ed.)*. https://www.weforum.org/publications/global-risks-report-2024/

Zhang, J., Liu, C., Li, X., Zhen, H.-L., Yuan, M., Li, Y., y Yan, J. (2023). A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519, 205–217. https://doi.org/10.1016/j.neucom.2022.11.024

Autores

Julián Silva Rodríguez. Ingeniero Industrial, Magister en Ingeniería Industrial, Doctor en Ingeniería de Procesos; Docente-Investigador, Grupo de Investigación Gestión Industrial y Administrativa - GIGIA, Programa de Ingeniería Industrial, Universidad Manuela Beltrán, Bogotá, Colombia.

ORCID: https://orcid.org/0000-0001-7497-8632
Email: julian.silva@docentes.umb.edu.co

Andrés Fernández Rosas. Ingeniero Industrial, Magister en Administración, Candidato a Doctor en Gestión; Docente-Investigador, Grupo de Investigación de Estudios Empresariales en Entornos Inciertos - EEENI, Programa de Administración de Empresas, Universidad Pedagógica y Tecnológica de Colombia, Tunja, Colombia..

ORCID: https://orcid.org/0009-0008-8446-5871
Email: andres.fernandez03@uptc.edu.co

Recibido: 06-06-2025 **Aceptado:** 20-11-2025