Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale, Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Handling discrete decisions in bilevel optimization via neural network embeddings

Isabela Fons Moreno-Palancas*, Raquel Salcedo Díaz, Rubén Ruiz Femenia and José A. Caballero

University of Alicante, Department of Chemical Engineering, San Vicente del Raspeig, Alicante, Spain
* Corresponding Author: isabela.fons@ua.es.

ABSTRACT

Bilevel optimization is an active area of research within the operations research community due to its ability to capture the interdependencies between two levels of decisions. This study introduces a metamodeling approach for addressing mixed-integer bilevel optimization problems, exploiting the approximation capabilities of neural networks. The proposed methodology employs neural network embeddings to approximate the optimal follower's response, bypassing the inner optimization problem by parametrizing it with continuous leader's decisions. The use of Rectified Linear Unit (ReLU) activations allows the forward pass of the neural network to be represented as a set of mixed-integer linear constraints. Thereby, the bilevel structure is simplified into a single-level optimization model. A case study based on a two-echelon supply chain demonstrates the effectiveness of the approach, with solutions comparable to traditional bilevel optimization methods. The results suggest that neural network embeddings offer a promising alternative for tackling complex bilevel problems even when discrete decisions are involved and unveil the trade-off between prediction accuracy and computational demands. This methodology provides a versatile and efficient framework for solving mixed-integer bilevel optimization problems across diverse domains.

Keywords: Bilevel Optimization, Neural Network Embeddings, Supply Chain Planning, Surrogate Modelling, MILP reformulation

INTRODUCTION

Many problems relevant to process engineering involve multiple stakeholders with conflicting interests and whose individual decisions are affected by each other's actions. Such hierarchical decision-making processes can be effectively modelled as bilevel optimization problems, where upper-level problems are constrained by inner optimization problems.

The nested structure of these models makes bilevel optimization one of the most complex branches of mathematical programming. Specifically, the development of solution algorithms is particularly difficult as it depends on the nature of the inner optimization problem and the interaction between both levels. This complexity is further amplified when the problems involve mixed-integer variables, leading to mixed-integer bilevel problems (MIBPs).

From Moore and Bard's foundational studies on mixed-integer bilevel problems [1], significant progress has been made under the assumption of linearity. In general, the methods to solve mixed integer linear bilevel problems (MILBPs) typically begin with a high-point relaxation—optimizing the upper-level objective while considering both levels' constraints—and systematically explore the feasible region to identify optimal solutions. Strategies such as branch-and-bound, branch-and-cut, Benders decomposition, parametric integer programming, or their combinations have been effectively applied [2]. While these methods often converge to exact solutions for MILBP, solving mixed-integer nonlinear bilevel problems (MINLBPs) remains more challenging, with most algorithms achieving only ε-optimal solutions.

Given the inherent difficulties of MIBPs, data-driven alternatives have gained considerable attention over the recent years. Despite the evident advantages of sophisticated methods such as reinforcement learning (RL), which has demonstrated considerable potential and in the field of game theory [3,4], neural network embeddings remain a powerful and flexible alternative to conventional methods [5,6,7].

Our proposed approach builds on the idea of using fully connected deep neural networks to approximate the follower's optimal response. When the network induces non-linearity via piecewise linear activation functions, it can be formulated as mixed-integer linear constraints and embedded into the leader's optimization problem. We illustrate the capabilities of the methodology in the optimization of a simplified supply chain, which highlights the potential advantages of this approach and reveals its remaining challenges.

MATHEMATICAL FRAMEWORK

In this study, we focus on mixed-integer bilevel due to its countless applications in process engineering in areas such as supply chain management and energy systems. The leader's problem, defined in Eq. 1, aims to optimize the function $F(x^L, y^L, x^F, y^F)$ subject to constraints $G(x^L, y^L, x^F, y^F)$. Here, $x^L \in \mathbb{R}^{q^L}$ and $y^L \in \mathbb{Z}^{p^L}$ represent the leader's continuous and integer decision variables, while $x^F \in \mathbb{R}^{q^F}$ and $y^F \in \mathbb{Z}^{p^F}$ denote the follower's continuous and integer decision variables, respectively.

$$\begin{aligned} \min \quad & F(x^L, y^L, x^F, y^F) \\ s.t., \quad & G(x^L, y^L, x^F, y^F) \leq 0 \\ & (x^F, y^F) \in \psi(x^L, y^L) \\ & x^L \in \mathbb{R}^{q^L} \\ & y^L \in \mathbb{Z}^{p^L} \end{aligned} \tag{1}$$

The function $\psi(x^L, y^L)$ corresponds to the parametric solution of the follower's optimization problem, which is presented in Eq. 2. This hierarchical structure introduces significant computational challenges, as the leader must anticipate the follower's optimal response while solving their own problem. Analogously, the leader's decisions influence the follower's objective $f(x^L, y^L, x^F, y^F)$ and feasible set, defined by constraints $g(x^L, y^L, x^F, y^F)$.

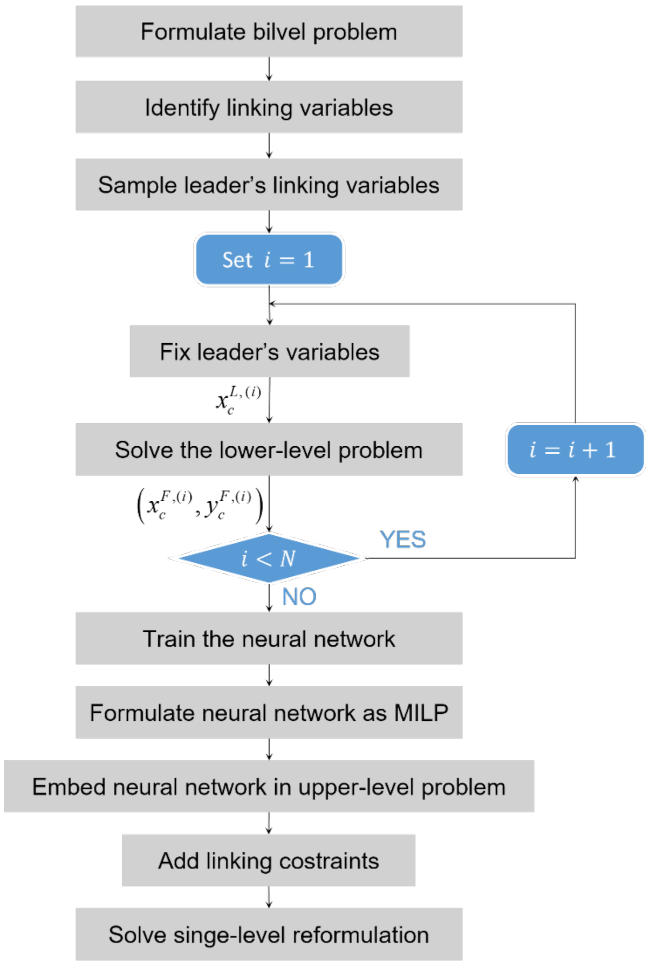
$$\begin{aligned} \psi(x^L, y^L) = \quad \text{argmin} \quad & f(x^F, y^F, x^L, y^L) \\ s.t., \quad & g(x^F, y^F, x^L, y^L) \leq 0 \\ & x^F \in \mathbb{R}^{q^F} \\ & y^F \in \mathbb{Z}^{p^F} \end{aligned} \tag{2}$$

METHODOLOGY

Our approach is grounded in the work developed by Dumouchelle et al. (2022) [8], which utilized neural network embeddings to tackle two-stage stochastic programming problems. The proposed workflow, outlined in **Figure 1**, involves the following steps.

1. Identification of the linking variables

As starting point, the bilevel model must be formulated and its structure must be carefully analysed to understand the interaction between both levels. This inspection is essential to identify linking variables that connect the upper and lower levels and contain the information exchanged between the players.



Formulate bilvel problem
Identify linking variables
Sample leader's linking variables
Set $i = 1$
Fix leader's variables
$x_c^{L,(i)}$
Solve the lower-level problem
$i = i + 1$
$\left(x_c^{F,(i)}, y_c^{F,(i)}\right)$
$i < N$
YES
NO
Train the neural network
Formulate neural network as MILP
Embed neural network in upper-level problem
Add linking costraints
Solve singe-level reformulation

Figure 1: Overview of the proposed methodology

Linking variables play a key role in this work, since they constitute the inputs and outputs of the neural network used to approximate the optimal response of the follower. Notably, not all of the leader's variables necessarily influence the follower's decisions, and vice-versa. Identifying this subset helps reducing the dimensionality of the input and output spaces, simplifying the network architecture. Therefore, the linking variables will be a subset of the leader's decision variables $(x_c^L, y_c^L) \subseteq (x^L, y^L)$.

Although in general both leader's continuous and discrete decisions impact the follower's solution, we exclusively contemplate lower-level problems parametrized by continuous leader's decisions (i.e., $\psi(x_c^L)$), since training neural networks with endogenous integer variables is out of the scope of this study.

2. Generation of training data

The neural network is trained in a supervised manner. Thus, a set of input-output pairs is required so that it learns the mapping ψ from $x_c^L \mapsto (x^F, y^F)$. Such dataset is generated through an iterative procedure.

First, a set of feasible solutions of the upper-level problem is randomly sampled using strategies such as Latin Hypercube Sampling. Both the sampling method and the number of data points must ensure a representative coverage of the feasible region in problem 1. Then, for each sampled leader's solution, the corresponding label is computed by solving the lower-level problem in Eq. 2, where the leader's variables function as parameters.

3. Training of the neural network

A fully connected neural network $(x_F, y_F) = \mathcal{N}(x_L; \theta)$ is trained to predict the optimal follower's decisions as a function of the leader's decisions. Here, coefficients θ represent the weights and biases of the network that must be optimized during training. Rectified Linear Unit (ReLU) activations introduce non-linearity to the outputs of each neuron in hidden layers, whereas linear transformations are applied in the output layer.

The prepared dataset is divided into training (70%), validation (20%) and test (10%) subsets. The training set is used to fit model weights, the validation set for architecture selection and hyperparameter tuning, and the test set to evaluate model performance.

4. Neural network as MILP

Mathematically, the output of a ReLU-activated neuron j in layer m can be written as

$$h_j^m = \max\left(0, \textstyle\sum_{i=1}^{n_{m-1}} w_{ij}^{m-1} h_i^{m-1} + b_j^{m-1}\right) \tag{3}$$

where w_{ij}^m is the element of matrix W^m at the i-th row and j-th column and b_j^m is the j-th element of vector b^m, being W^m and b^m the weights and biases associated with layer m, respectively.

Since the expression in Eq. 3 merely describes a piecewise linear function, the forward propagation step of a neural network with ReLU activations can be modelled as a set of linear constraints. Though different reformulation alternatives have been proposed in the literature, they all involve the introduction of binary variables to indicate which region of the ReLU function is active [9]. Following the extended Big-M reformulation proposed by Serra et al. (2017) [10], a representation of an l-layered feed-forward neural network is shown in Eq. 4, where n_m identifies the number of neurons in a given layer m.

$$\begin{array}{ll}
\sum_{i=1}^{n_0} w_{ij}^0 x^L + b_j^0 = \hat{h}_j^1 - \check{h}_j^1 & \forall\, j \in \{1, \dots, n_0\} \\
\sum_{i=1}^{n_{m-1}} w_{ij}^{m-1} \hat{h}_j^{m-1} + b_j^{m-1} = \hat{h}_j^m - \check{h}_j^m & \forall\, j \in \{1, \dots, n_m\}, \; m \in \{2, \dots, l-1\} \\
\sum_{i=1}^{n_l} w_{ij}^l \hat{h}_j^l + b_j^l = \beta_j & \forall\, j \in \{1, \dots, n_l\} \\
\hat{h}_j^m \le {M^+}_j^m z_j^m & \forall\, j \in \{1, \dots, n_m\}, \; m \in \{1, \dots, l-1\} \\
\check{h}_j^m \le {M^-}_j^m (1 - z_j^m) & \forall\, j \in \{1, \dots, n_m\}, \; m \in \{1, \dots, l-1\} \\
\hat{h}_j^m, \check{h}_j^m \ge 0 & \forall\, j \in \{1, \dots, n_m\}, \; m \in \{1, \dots, l-1\} \\
z_j^m \in \{0,1\} & \forall\, j \in \{1, \dots, n_m\}, \; m \in \{1, \dots, l-1\}
\end{array} \tag{4}$$

Note that the pair $(\hat{h}_j^m, \check{h}_j^m)$ is complementary, meaning that only one can be non-zero, and only the non-negative counterpart $\hat{h}_j^m$ of the weighted sum is propagated through the network. This exclusive disjunction is enforced by binary variables z_j^m. The outputs of the neural network are denoted as β_j.

In addition, Big-M coefficients M^+ and M^- must be carefully chosen to ensure tight linear programming (LP) relaxations. For this purpose, we adopt the propagation rule introduced by Serra et al. [10].

5. Single-level reformulation

Replacing the inner optimization problem by the MILP representation of the surrogate model of $\psi(x^L)$ transforms the original bilevel problem in Eq. 1 into a single-level model, which can be directly handled by state-of-the art optimization solvers. The proposed single-level reformulation is presented in Eq. 5.

$$\begin{array}{ll}
\min & F(x^L, y^L, x^F, y^F) \\
s.t., & G(x^L, y^L, x^F, y^F) \le 0 \\
 & \text{Equations in 4} \\
 & \text{Linking block} \\
 & x^L \in \mathbb{R}^{q^L} \\
 & y^L \in \mathbb{Z}^{p^L}
\end{array} \tag{5}$$

The Linking block refers to the set of constraints that map $\beta \mapsto (x^F, y^F)$, being therefore case-dependent.

RESULTS AND DISCUSSION

To illustrate the application of the proposed methodology, we consider the simplified two-echelon supply chain in **Figure 2**, adapted from Yue and You (2017) [11].

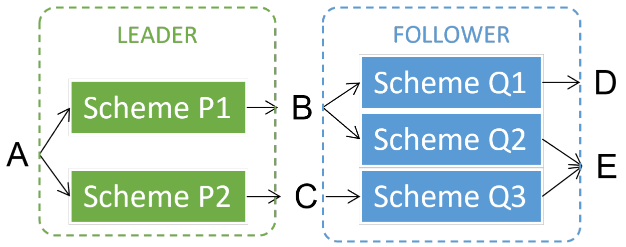


Figure 2: Supply chain superstructure

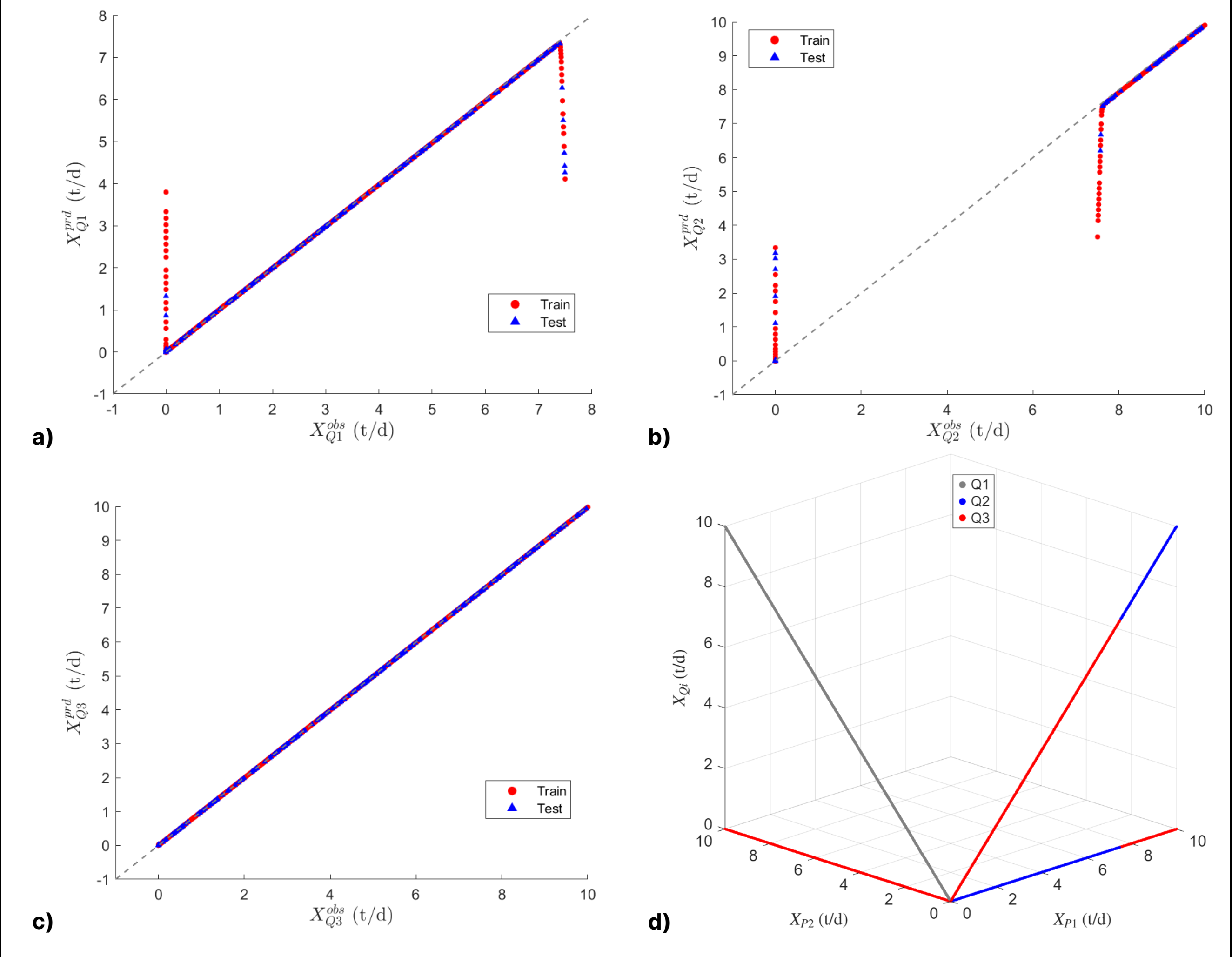


Figure 3: Observations versus predictions for the network outputs across training and testing datasets. Subplots (a), (b), and (c) show the predicted versus observed values for X_{Q1}, X_{Q2} and X_{Q3}, respectively, with the dashed line representing the ideal $y = x$ relationship. Subplot (d) provides a 3D visualization of the network outputs (X_{Q1}, X_{Q2} and X_{Q3}) as a function of the inputs (X_{P1} and X_{P2}).

In this Stackelberg game with a single leader and a single follower, both participants aim at maximizing their own profits. The optimization problems for the upper and lower levels are formulated below.

- Upper-Level Problem (Leader):

$$\begin{aligned}
\max\ & \text{sales}^L - \text{costs}^L \\
s.t.\ & \text{sales}^L = p_B^L X_{P1} + p_C^L X_{P2} + p_E^L (X_{Q1} + X_{Q2}) \\
& \text{costs}^L = \textstyle\sum_{i \in I^L} (p_A^L + v_i) X_i + f_i Y_i \\
& \textstyle\sum_{i \in I^L} X_i \le C^L \\
& \textstyle\sum_{i \in I^L} Y_i \le 1_i \\
& X_i \ge C^L Y_i \qquad \forall\, i \in I^L \\
& X_i \in \mathbb{R}^+, Y_i \in \{0,1\} \quad \forall\, i \in I^L \\
& I^L = \{P1, P2\} \\
& (X_{Q1}, X_{Q2}, X_{Q3}, Y_{Q1}, Y_{Q2}, Y_{Q3}) = \psi(X_{P1}, X_{P2})
\end{aligned} \tag{6}$$

- Lower-Level Problem (Follower):

$$\begin{aligned}
\psi(X_{P1}, X_{P2}) = \\
\max\ & \text{sales}^F - \text{costs}^F \\
s.t.\ & \text{sales}^F = p_D^F X_{Q1} + p_E^R (X_{Q2} + X_{Q3}) \\
& \text{costs}^F = p_B^F X_{P1} + p_C^F X_{P2} + \textstyle\sum_{i \in I^F} v_i X_i + f_i Y_i \\
& \textstyle\sum_{i \in I^F} X_i \le C^F \\
& \textstyle\sum_{i \in I^F} Y_i \le 1_i \\
& X_i \ge C^F Y_i \qquad \forall\, i \in I^F \\
& X_i \in \mathbb{R}^+, Y_i \in \{0,1\} \quad \forall\, i \in I^F \\
& I^F = \{Q1, Q2, Q3\}
\end{aligned} \tag{7}$$

To approximate the solution of problem 7, a neural network was trained to predict the follower's optimal real-valued decisions $X_{i \in I^F}$, while binary variables $Y_{i \in I^F}$ are subsequently inferred from the network's predictions. In the above formulations, the parameters C^L and C^F represent the productive capacity of the leader and the follower, respectively. Similarly, p_s^L and p_s^F refer to the selling or purchase price of species s, depending on the player.

A dataset containing 5000 points was generated by solving problem 7 for different combinations of (X_{P1}, X_{P2}). CPLEX was used as optimization algorithm and the data generation time rose to 964 s.

Different network architectures (e.g., varying the number of layers and neurons per layer) were considered. First, we employed Hyperband search to explore the predefined hyperparameter space and select the best architecture. The learning rate was further optimized using the learning rate range test [12]. The network with the optimal hyperparameter configuration was finally trained for 200 epochs on a combined training and validation set using Adam optimizer. Its performance was then assessed both on the train and test sets, reporting the corresponding mean absolute error (MAE) in **Table 1**. All models were implemented and trained using TensorFlow 2.16.1.

The results, as shown in **Table 1**, indicate low MAE scores, proving the suitability of the neural network as an accurate surrogate for the optimal lower-level solution. The accuracy of the network outputs, as indicated by the low MAE scores, is further validated by the observation-versus-prediction plots in **Figure 3**. Subplots **a**–**c** illustrate that the predictions closely align with the labels across both training and testing datasets, as the points cluster tightly around the diagonal line. Minor deviations are observed only in regions where the original function exhibits discontinuities, consistent with the patterns seen in the 3D representation in **Figure 3d**. Such sharp transitions are inherently challenging for neural networks—or most regression models—to predict, since they are well-suited for capturing differentiable patterns. In this case, these mispredictions are localized and do not compromise the method's overall effectiveness, as the model properly captures the underlying system's behaviour.

Nevertheless, its performance could be improved by either exposing the network to more data during training or adopting more complex network architectures, which, however, are more prone to overfitting. The former approach would increase the data generation time, which is already the bottleneck of the process. In contrast, the latter would negatively impact the convergence rate of the single-level model due to the expansion in size of the neural network embedding. Hence, there is a clear trade-off between prediction accuracy and computational demand.

Table 1: Neural network configuration and performance

Neurons per layer	[16,16,16]
Learning rate	$2 \cdot 10^{-3}$
Batch size	32
Optimizer	Adam
MAE	0.052 (train); 0.054 (test)
Training time (s)	43.4

Table 2: Results comparison

	NN solution	KKT solution
Profit Leader (MU/h)	350	350
Profit Follower (MU/h)	150	150
X_{P1} (ton/d)	0	0
X_{P2} (ton/d)	10	10
X_{Q1} (ton/d)	0	0
X_{Q2} (ton/d)	10	10
X_{Q3} (ton/d)	0	0
Solving time (s)	0.79	0.53

The single-level reformulation of the original problem, based on the neural surrogate, is given in Eq. 8.

$$\begin{aligned}
\max\ & \text{sales}^L - \text{costs}^L \qquad (8)\\
s.t.\ & \text{sales}^L = p_B^L X_{P1} + p_C^L X_{P2} + p_E^L\left(X_{Q1} + X_{Q2}\right)\\
& \text{costs}^L = \textstyle\sum_{i\in I^L}(p_A^L + v_i)X_i + f_i Y_i\\
& \textstyle\sum_{i\in I^L} X_i \le C^L\\
& \textstyle\sum_{i\in I^L} Y_i \le 1_i\\
& \textstyle\sum_{i\in I^L} w_{ij}^0 \mathrm{X_i} + b_j^0 = \hat{h}_j^1 - \check{h}_j^1 \quad \forall\, j \in \{1,\dots,16\}\\
& \textstyle\sum_{i=1}^{16} w_{ij}^1 \hat{h}_j^1 + b_j^1 = \hat{h}_j^2 - \check{h}_j^2 \quad \forall\, j \in \{1,\dots,16\}\\
& \textstyle\sum_{i=1}^{16} w_{ij}^2 \hat{h}_j^2 + b_j^2 = \hat{h}_j^3 - \check{h}_j^3 \quad \forall\, j \in \{1,\dots,16\}\\
& \textstyle\sum_{i=1}^{16} w_{ij}^3 \hat{h}_j^3 + b_j^3 = \beta_j \quad \forall\, j \in \{1,2,3\}\\
& \hat{h}_j^m \le {M^+}_j^m z_j^m \quad \forall\, j \in \{1,\dots,n_m\}, m \in \{1,2,3\}\\
& \check{h}_j^m \le {M^-}_j^m\left(1 - z_j^m\right) \quad \forall j \in \{1,\dots,n_m\}, m \in \{1,2,3\}\\
& \hat{h}_j^m, \check{h}_j^m \ge 0 \quad \forall\, j \in \{1,\dots,n_m\}, m \in \{1,2,3\}\\
& z_j^m \in \{0,1\} \quad \forall\, j \in \{1,\dots,n_m\}, m \in \{1,2,3\}\\
& \begin{bmatrix} \boldsymbol{Y_{Qi}} \\ \beta_i \ge \varepsilon \\ \beta_i - \varepsilon \le X_{Qi} \le \beta_i + \varepsilon \\ 0 \le X_{Qi} \le C^L \end{bmatrix} \veebar \begin{bmatrix} \neg \boldsymbol{Y_{Qi}} \\ \beta_i \le \varepsilon \\ X_{Qi} = 0 \end{bmatrix} \quad i = \{1,2,3\}\\
& X_i \ge C^L Y_i \qquad \forall\, i \in I^L\\
& X_i \in \mathbb{R}^+, Y_i \in \{0,1\} \quad \forall\, i \in I^L\\
& I^L = \{P1, P2\}
\end{aligned}$$

The Big-M reformulation applied to the disjunction in Eq. 8 results in a set of linear constraints that constitute the so-called Linking block in Eq. 5. These equations are designed to mitigate the neural network approximation error and ensures that non-selected production schemes of the follower have a production rate of exactly 0. In this context, the parameter ε acts as a tolerance threshold that accounts for the prediction error of the neural network. Specifically, it limits an allowable margin around the predicted values that guarantees consistency between the network's outputs and the logical conditions of the optimization problem.

For comparison purposes, the original model is solved by applying a direct reformulation using Karush-Kuhn-Tucker (KKT) conditions for all feasible combinations of binary variables. Both models were implemented in Pyomo 6.8.2, using CPLEX 22.1.1.0 as MILP solver and reach identical results, as **Table 2** summarizes, due to the effect of linking constraints.

Since the objective of this work was to demonstrate the effectiveness and reliability of the proposed methodology in solving mixed-integer bilevel instances, the selected case study was not complex enough to evaluate the potential advantages of the methodology in terms of solution speed compared to traditional approaches. Consequently, future research will focus on exploring the benefits of neural network embeddings in large-scale models.

CONCLUSION

This contribution presents a metamodeling approach to mixed integer bilevel optimization, where the inner-level problem is parametrized by continuous decisions of the leader. In this study, the lower level is bypassed using a neural network that approximates the optimal follower's response to the leader's decisions. By employing Rectified Linear Unit activations, the neural network can be reformulated as a set of mixed integer linear constraints that are integrated into the leader's problem as a single-level reformulation. Additional constraints are introduced to map the network outputs to the corresponding follower decision variables and to infer their discrete behaviour where needed.

The approach was showcased on a two-echelon supply chain case study, achieving solutions whose quality is comparable to traditional bilevel optimization methods. These results suggest that further effort to improve the use neural network embeddings in mixed-integer bilevel optimization tasks could lead to significant advantages, especially when applied to larger-scale instances. Future research should primarily focus on ensuring prediction accuracy, which remains a critical challenge of the proposed methodology, while carefully balancing model complexity, computational resources and reliability to enhance its competitiveness in terms of computational efficiency.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge financial support from the Spanish Ministry of Science, Innovation, and Universities via the doctoral grant for training university teachers FPU22/01276 and the R&D and Innovation Project PID2021-124139NB-C21.

REFERENCES

1. Moore, J. T. and J. F. Bard (1990). The mixed integer linear bilevel programming problem. *Oper. Res.* 38(5):911–921 https://doi.org/10.1287/opre.38.5.911
2. Kleinert T, Labbé M, Ljubić I, Schmidt M. A Survey on Mixed-Integer Programming Techniques in Bilevel Optimization. *EURO J. Comput. Optim.* 9: 100007 (2021) https://doi.org/10.1016/j.ejco.2021.100007
3. Qiang Z, Yefei Y, Fangfang M. A Stackelberg-based deep reinforcement learning approach for dynamic cooperative advertising in a two-echelon supply chain. *Comp. Chem. Eng.* 196:109048 (2025) *https://doi.org/10.1016/j.compchemeng.2025.109048*
4. Moti M, Uddin RS, Hai MA, Saleh TB, Alam MGR, Hassan MM, Hassan MR, Blockchain Based Smart-Grid Stackelberg Model for Electricity Trading and Price Forecasting Using Reinforcement Learning. *Appl. Sci.* 12(10):5144 (2022) https://doi.org/10.3390/app12105144
5. Jiménez RX. Bilevel optimisation with embedded neural networks: Application to scheduling and control integration. *arXiv* (2023) https://doi.org/10.48550/arXiv.2304.11244
6. Vlah D, Sepetanc K, Pandzic H. Solving bilevel optimal bidding problems using deep convolutional neural networks. IEEE Syst. J. 17(2): 2767-2778 (2023) https://doi.org/10.1109/JSYST.2022.3232942
7. Dumouchelle J, Julien E, Kurtz J, Khalil E. Neur2BiLO: Neural Bilevel Optimization. *arXiv* (2024) https://doi.org/10.48550/arXiv.2402.02552
8. Dumouchelle J, Patel R, Khalil EB, Bodur M. Neur2SP: Neural Two-Stage Stochastic Programming. *arXiv* (2022) https://doi.org/10.48550/arXiv.2205.12006
9. Aftabi N, Moradi N, Mahroo F. Feed-forward neural networks as mixed-integer program. *arXiv* (2024) https://doi.org/10.48550/arXiv.2402.06697
10. Serra T, Tjandraatmadja C, Ramalingam S. Bounding and counting linear regions of deep neural networks. *arXiv* (2017) https://doi.org/10.48550/arXiv.1711.02114
11. Yue D, You F. Stackelberg-game-based modeling and optimization for supply chain design and operations: A mixed integer bilevel programming framework. *Comput. Chem. Eng.* 102:81-95 (2017) https://doi.org/10.1016/j.compchemeng.2016.07.026
12. Smith LN. Cyclical Learning Rates for Training Neural Networks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Santa Rosa, CA, USA. 464-472 (2017) https://doi.org/10.1109/WACV.2017.58