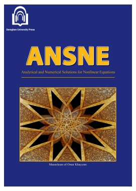

Analytical and Numerical Solutions for Nonlinear Equations
©Available online at https://ansne.du.ac.ir/
Online ISSN: 3060–785X
2025, Volume 10, Issue 2, pp. 237–254

Research article

Nonlinear Optimization Problems with Bipolar Fuzzy Relation Equations using Neural Networks

Ali Abbasi Molai[1], Hassan Dana Mazraeh[1,*], Kourosh Parand[2,3]

[1] School of Mathematics and Computer Science, Damghan University, Damghan, Iran

[2] Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran

[3] International Business University,Toronto, Canada

*** Corresponding author(s):** dana@du.ac.ir

Received: 24/11/2025 **Revised:** 26/12/2025 **Accepted:** 27/12/2025 **Published:** 30/12/2025 10.22128/ansne.2025.3161.1181

Abstract

In this paper, we present a novel application of neural networks for solving nonlinear optimization problems subject to bipolar max-min fuzzy relation equation constraints. The feasible solution set for these problems is generally non-convex, which makes conventional nonlinear optimization methods less suitable for solving them. To address this challenge, we propose the use of neural networks and some rules for simplification of the problem. To find an input vector $x \in [0,1]^n$ that satisfies the constraints and minimizes (or maximizes) the objective function, n neural networks are trained simultaneously. Each neural network identifies the corresponding variable of the vector $x \in [0,1]^n$. The loss function integrates both the constraints and the objective function. Our experiments demonstrate that the proposed method can solve these problems with high accuracy and reasonable computational time. The proposed method is compared to the existing methods.

Keywords: Neural networks, Nonlinear optimization problems, Bipolar fuzzy relation equations, Novel architecture, Max-Min composition

Mathematics Subject Classification (2020): 90-08, 68T07

1 Introduction

The fuzzy relation programming problem has attracted the attention of researchers due to its applications in the real world since Fuzzy Relation Equations (FREs) were proposed by Sanchez in 1976 [1]. The linear programming problem with the max-min FRE constraints was firstly appeared to minimize the costs in the textile industry by Fang and Li [2]. They decomposed the problem into two subproblems and applied the branch and bound method to solve it. To reduce the computational complexity, researchers proposed some procedures or rules to shrink the search space of the optimal solution [3–7]. However, we cannot formulate all of the real world problems by the linear objective functions. Hence, some researchers focussed on the problem of nonlinear programming problem with the FRE constraints [8–24]. Different classes of nonlinear programming problem with FRE or FRI constraints were studied and researchers presented algorithms to solve them. The linear fractional programming problem with FRE constraints was considered by Wu et al [11] with the max-Archimedean t-norm composition. This problem was converted to a traditional linear fractional programming problem using the special structure of FRE system. Li and Fang [12] investigated the problem with Sup-T composition where T is a continuous Archimedean triangular norm. They converted it to a 0-1 linear fractional programming problem in polynomial time and applied the parametrization techniques to solve it. The quadratic programming problem with FRI constraints was converted to a separable programming problem using the properties of symmetric matrices, cholesky's decomposition, and least square technique [13]. A matrix-based algorithm was designed to solve the separable programming problem with FRE constraints in [14]. Yang and her colleagues have extensively studied geometric fuzzy relation programming and other classes of fuzzy relation programming from theoretical and practical perspectives [15–24]. Since the feasible domain of the nonlinear programming problem is non-convex, in a general case, the traditional methods cannot be applied to solve the problem. Moreover, we cannot design an exact algorithm to solve the problem, in a general case. Therefore, researchers used evolutionary algorithms to solve such problems. Genetic Algorithms (GAs) are commonly used to generate the approximate optimal solutions for optimization problems by biologically inspired operators such as selection, crossover, and mutation. Lu and Fang [8] applied a genetic algorithm to find the approximate optimal solution for the nonlinear programming problem provided to the FRE constraints with the max-min operator. The individuals for the initial population are selected from its feasible solution set. During the mutation and crossover process, the new generated populations are feasible. The nonlinear optimization problem with FRE constraints was investigated by Khorram and Hassanzade [9] along with the max-average composition. With regard to the special structure of its feasible domain, a modified GA was designed to find its approximate optimal solution. A genetic algorithm was proposed for the problem with the max-product composition operator by Hassanzade et al [10] in two cases of the convex and non-convex solution sets. Its efficiency was studied by some test problems. Also, a genetic algorithm was designed to solve the problem with the max-Lukasiewicz composition [25] and the max-Dubois-Prade composition [26]. The linear objective function optimization subject to the max-t FRE constraints was studied for a wide class of t-norms where t is an Archimedean-t-norm in [27]. The authors [27] applied the concept of covering the problem to establish its equivalent 0-1 integer programming problem. Then, a binary-coded genetic algorithm was designed to find its optimal solution. The linear programming problem provided to a generalized fuzzy relational inequalities was investigated with an arbitrary continuous t-norm and fuzzy inequality replaces ordinary inequality in the constraints. The authors in [28] applied a modified PSO to solve the problem. The used FREs and FRIs in the above problems are increasing with respect to each of the variables. Some applications need to variables with bipolar characters. We can consider the importance of the variables to model the product public awareness in revenue management problem [29]. The problem can be reformulated in terms of a system of bipolar max-min FREs. Freson et al [29] first studied such systems and determined their complete solution set. They also investigated the corresponding linear programming problem to it and presented an approach to trace its optimal solution using the structure of its feasible domain. The problems of checking feasibility and finding the optimal solution are NP-complete [30] and NP-hard, respectively. Li and Liu [31] studied the problem with the max-lukasiewicz t-norm and converted it into an 0-1 integer linear programming problem. They presented a systematic approach to find its optimal solution. Liu et al [32] showed that each component of its optimal solution could either be the corresponding component in the lower or upper bound vector of its feasible domain. Using the property, a simple value matrix along with some rules was proposed to solve the problem without converting it into a 0-1 integer linear programming problem. Yang [33] presented a bipolar path approach to solve the system of max-Lukasiewicz BPFRE. The linear optimization problem was also investigated with the max-parametric Hamacher composition in [34]. Some simplification rules are given to reduce its dimensions and find some of its optimal variables, directly. An algorithm was designed to solve the simplified problem with regard to a produced upper bound for the optimal objective value. Some rules were presented to simplify the problem based on removing the redundancy constraints [35] and the covering [36]. As it was mentioned previously, a number of phenomena have a nonlinear nature. Therefore, we have to study them. Some researchers investigated the nonlinear programming problems provided to BFRE in terms of special classes such as quadratic [37] and geometric [38, 39] functions.

Some sufficient conditions were proposed for the quadratic optimization with BFRE constraints to detect some of its optimal components or one of its optimal solutions in [37]. Then, the reduced problem can be converted to a 0-1 mixed integer programming problem and solved by the traditional techniques. The problem was discussed with a geometric function along with some simplification rules and a modified branch-and-bound method was extended on a value matrix to solve the problem in [38, 39]. Zhou et al [40] considered the nonlinear programming problem with the max-Lukasiewicz triangular norm. Since the system of BFRE can be reformulated in terms of a system of 0-1 mixed integer programming, they converted the original problem to an 0-1 mixed integer nonlinear programming problem. The classic integer programming techniques have a high computational complexity. Hence, we are motivated to look for efficient and less expensive computational methods. Nowadays, metaheuristic algorithms are the best candidates to solve problems with a high computational complexity. Since the nonlinear programming problem with the max-min BFRE constraints is an NP-hard problem, we need to select an efficient algorithm to solve the problem. Recently, researchers applied some of the algorithms to solve fuzzy relation programming problems. As it was mentioned previously, they applied some genetic algorithms [8–10, 25, 26] to find the approximate optimal objective value of the nonlinear programming problem with FRE constraints using the operators of the max-min, max-product, max-average, max-Dubois-Prade, max-Lukasiewicz composition. The objective function of the studied problem in [27, 28] is linear, and its constraints as FRIs and FREs. A two-phase-ant colony algorithm was presented to solve nonlinear programming problems with the max-min FREs in [41]. The authors in [42] extended the models to a nonlinear programming problem subject to a system of the max-min bipolar fuzzy relation equations. The proposed GA in [42] covered the resolution of a more general case from the nonlinear programming problem with bipolar FRE constraints with the desired accuracy. Now, we intend to improve the accuracy of the optimal solution in a desirable time. To do this, we design a neural network to find the optimal objective value of the problem with better precision and execution time.

In recent years, neural networks have shown remarkable capability in solving complex scientific problems, positioning them as a powerful alternative to traditional methods in various fields of science and engineering [43–47]. Investigating the capability of neural networks for solving nonlinear optimization problems constrained by bipolar max-min fuzzy relation equations is essential due to the intrinsic complexity and non-convexity of these problems. Traditional methods, such as branch-and-bound, integer programming, or metaheuristic approaches like genetic algorithms, often suffer from high computational costs, limited scalability, and suboptimal accuracy when dealing with such constraints. Neural networks, with their ability to model and approximate complex relationships, offer a promising alternative by simultaneously addressing the objective function and constraints through a unified loss function. This exploration is critical for advancing computational techniques in fields where BFRE constraints naturally arise, such as decision-making, engineering optimization, and resource allocation, enabling more efficient and accurate solutions to these inherently challenging problems.

This paper introduces an innovative application of neural networks to solve nonlinear optimization problems constrained by bipolar max-min fuzzy relation equations. The methodology integrates both constraints and objective function into a unified loss function. This results in high accuracy and computational efficiency, surpassing traditional methods and metaheuristics like genetic algorithms, Particle Swarm Optimization (PSO) algorithm, and Ant Colony algorithm. The integration of neural networks as optimization tools represents a novel pattern which offers a powerful, scalable, and precise alternative for tackling complex BFRE-constrained problems.

The structure of this paper is as follows. Section 2 formulates the nonlinear programming problem subject to the BFRE constraints and investigates the structure of its feasible domain. In Section 3, some rules are proposed to simplify the problem.An innovative approach based on neural networks to solve the problem is detailed in Section 4. The experimental results obtained using neural networks are presented in Section 5 and the results are compared to the existing methods. An analysis of these results is provided in Section 6. Finally, the conclusions are summarized in the last section.

2 Formulation of The Nonlinear Programming Problem Subject to the BFRE Constraints

In this section, the nonlinear programming problem is first formulated with the BFRE constraints using the max-min composition. Then, we investigate the structure of its feasible domain and present some necessary and sufficient conditions to check the consistency of the BFRE system. Suppose that $A^+ = [a_{ij}^+]$ and $A^- = [a_{ij}^-]$ be two $m \times n$ fuzzy relation matrices with $a_{ij}^+, a_{ij}^- \in [0,1]$, for each $i \in I = \{1,\dots,m\}$ and $j \in J = \{1,\dots,n\}$, and $b = [b_i]_{m\times 1} \in [0,1]^m$. We can formulate the nonlinear programming problem as follows:

$$\min(or\ \max) \quad f(x), \tag{1}$$

$$\text{s.t.} \quad A^+ \circ x \vee A^- \circ \neg x = b, \tag{2}$$

$$x \in [0,1]^n, \tag{3}$$

where the function of $f : R^n \to R$ is a nonlinear function. The operators of "$\circ$" and "$\bigvee$" denote the max-min composition and the maximum operation. The vector of $x = (x_1,\dots,x_n)^T \in [0,1]^n$ is the vector of decision variables to be specified and $\neg x = (1-x_1,\dots,1-x_n)^T$ is defined. The system (2)-(3) denotes the constraints of the problem (1)-(3) and it consists of the vectors $x \in [0,1]^n$ satisfying in the following relations:

$$\max_{j\in J}\ \max\left\{a_{ij}^+ \wedge x_j, a_{ij}^- \wedge (1-x_j)\right\} = b_i, \quad \forall i \in I. \tag{4}$$

The solution set of the system (2)-(3) (or system (4)) is denoted by $S(A^+, A^-, b)$. The system (2)-(3) is said to be consistent if its solution set, i.e., $S(A^+, A^-, b)$, is not empty. Otherwise, it is called inconsistent. Now, we present some necessary and sufficient conditions to check the consistency of system (2)-(3). First of all, we focus on the consistency of one of the equations (4) in the following lemma.

Lemma 1. *The necessary and sufficient condition for consistency of the i^{th} equation of the system* (4) *is as follows:*

$$b_i \in \left[\max_{j=1,\dots,n} min\left(a^+_{ij}, a^-_{ij}, \frac{1}{2}\right), \max_{j=1,\dots,n} \max\left(a^+_{ij}, a^-_{ij}\right)\right]. \tag{5}$$

Proof. See the Ref. [29]. □

Now, the above result is extended to the general case, i.e., system (4), in the following lemma.

Lemma 2. *The necessary and sufficient condition for consistency of the system* (4) *is as follows:*

$$\forall\, i \in I \;\; b_i \in \left[\max_{j=1,\dots,n} min\left(a^+_{ij}, a^-_{ij}, \frac{1}{2}\right), \max_{j=1,\dots,n} \max\left(a^+_{ij}, a^-_{ij}\right)\right]. \tag{6}$$

Proof. See the Ref. [29]. □

Now, the structure of the complete solution set of the system (2)-(3) (or system (4)) is briefly reviewed from ref. [29] as follows.
Let $g^{i+} = \left((g^{i+})_1, \dots, (g^{i+})_n\right)^T$ be a vector with the following components:

$$\left(g^{i+}\right)_j = \begin{cases} 1 & if\, a^+_{ij} \le b_i, \\ b_i & if\, a^+_{ij} > b_i, \end{cases} \quad j = 1, \dots, n\,.$$

The set $S^{i+} = \left\{s^{i+}_k = ((s^{i+}_k)_1, \dots, (s^{i+}_k)_n) \,\middle|\, a^+_{ik} \ge b_i\right\}$ is introduced where $(s^{i+}_k)_j = b_i \delta_{kj}$, $j = 1, \dots, n$, and δ_{kj} is Kronecker's delta. Also, let $g^{i*} = \left((g^{i*})_1, \dots, (g^{i*})_n\right)^T$ be a vector with the following components:

$$\left(g^{i*}\right)_j = \begin{cases} 1 & if\, a^-_{ij} \le b_i, \\ b_i & if\, a^-_{ij} > b_i, \end{cases} \quad j = 1, \dots, n\,,$$

The vector of s^{i-} is constructed based on the vector of g^{i*} as follows:

$$s^{i-} = \overline{g^{i*}} = (1 - (g^{i*})_1, \dots, 1 - (g^{i*})_n).$$

The set $S^{i*} = \left\{s^{i*}_k = ((s^{i*}_k)_1, \dots, (s^{i*}_k)_n) \,\middle|\, a^-_{ik} \ge b_i\right\}$ is defined where $(s^{i*}_k)_j = b_i \delta_{kj}$, for $j = 1, \dots, n$. Furthermore, let

$$G^{i-} = \{g \mid \exists s \in S^{i*} \; s.t. \;\; g = \bar{s}\}.$$

Based on the above notations, the complete solution set of the system (2)-(3) (or system (4)) is as follows.

$$D = \bigcap_{i=1}^{m} [s^{i-}, g^{i+}] \cap \left(\left(\bigcup_{s \in S^{i+}} [s \vee s^{i-}, g^{i+}]\right) \cup \left(\bigcup_{g \in G^{i-}} [s^{i-}, g \wedge g^{i+}]\right)\right),$$

where the notation of $\wedge$ denotes the minimum operation.
If $s^- = \sup_{i=1,\dots,m} s^{i-}$ and $g^+ = \inf_{i=1,\dots,m} g^{i+}$, then vectors s^- and g^+ are lower and upper bound of the solution set of the system (2)-(3) (or system (4)), respectively. Hence, the values $LB_i = s^-_i$ and $UB_i = g^+_i$ are the lower and upper bound for the optimal value of variable x_i in the problem (1)-(3). The above points provide the conditions to present an algorithm based on the neural network to find the approximate optimal objective value of the problem (1)-(3).

3 Some Procedures for Simplifying the Problem (1) to (3)

In this section, we present some rules to simplify the problem (1)-(3).

Rule 1. Suppose that $S(A^+,A^-,b) \neq \emptyset$. If for $i \in I$, $b_i = 0$, then for any $j \in J$, we have $a^+_{ij} = a^-_{ij} = 0$.

Proof. Suppose that $b_i = 0$, for $i \in I$. Then for any $j \in J$, $\max\limits_{j\in J} \max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\} = 0$. It results that for any $j \in J$, $\max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\} \leq 0$. Since $a^+_{ij}, a^-_{ij}, x_j, 1-x_j \in [0,1]$, therefore, we must have $a^+_{ij} = a^-_{ij} = 0$, for $i \in I$ and for all $j \in J$. □

In fact, Rule 1 says that if $b_i = 0$, for an $i \in I$, then we can remove the i^{th} constraint from system (2)-(3).

Rule 2. Suppose that $S(A^+,A^-,b) \neq \emptyset$. If for each $j \in J$, such that $a^+_{ij}, a^-_{ij} < b_i$, then we can remove $\max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\}$ from the i^{th} constraint of system (2).

Proof. Since $a^+_{ij}, a^-_{ij} < b_i$, then $\max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\} < \max\left\{a^+_{ij}, a^-_{ij}\right\} < b_i$. Therefore, the statement of $\max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\}$ has no effect on the solution set of the system (2)-(3) and it can be removed from the i^{th} constraint of the system (2)-(3). □

The next rule introduces components from two matrices A^+ and A^- that are ineffective in the process of finding the optimal solution of the problem (1)-(3).

Rule 3. Suppose that $S(A^+,A^-,b) \neq \emptyset$. If for each $j_0 \in J$, such that $a^+_{ij_0} < b_i$ and $a^-_{ij_0} \geq b_i$, then we can remove the statement $a^+_{ij_0} \wedge x_{j_0}$ from the i^{th} constraint of system (2).

Proof. Since $a^+_{ij_0} < b_i$ and $a^-_{ij_0} \geq b_i$, then $a^+_{ij_0} \wedge x_{j_0} < b_i$. Hence, the solution set of $\max\limits_{j\in J} \max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\} = b_i$ is equivalent to that of $\max\left\{\max\limits_{j\in J-\{j_0\}} \max\left\{a^+_{ij} \wedge x_j, a^-_{ij} \wedge (1-x_j)\right\}, a^-_{ij_0} \wedge (1-x_{j_0})\right\} = b_i$ and the statement $a^-_{ij} \wedge (1-x_j)$ cannot be removed because of $a^-_{ij_0} \geq b_i$. □

We can express a similar rule to Rule 3 as follows.

Rule 4. Suppose that $S(A^+,A^-,b) \neq \emptyset$. If for each $j_0 \in J$, such that $a^+_{ij_0} \geq b_i$ and $a^-_{ij_0} < b_i$, then we can remove the statement $a^-_{ij_0} \wedge (1-x_{j_0})$ from the i^{th} constraint of system (2).

Proof. It is similar to the proof of Rule 3. □

The components $a^+_{ij_0}$ and $a^-_{ij_0}$ mentioned in Rules 2, 3, and 4 are called ineffective in the process of finding the optimal solution of the problem (1)-(3) and the corresponding rules with them are simplification procedures. The above points provide the conditions to present an algorithm based on the neural network to find the approximate optimal objective value of the problem (1)-(3).

4 An Innovative Approach Based on Neural Networks for Solving Problems (1) to (3)

In this section, we present an innovative approach that utilizes neural networks to solve problem (1)-(3). In typical supervised learning tasks, neural networks are used to approximate functions with regard to a given set of inputs and outputs. The goal is to find a model that best generalizes the relationship between them [48]. However, in this research, the objective is to find the optimal input for the problem (1)-(3) such that it constitutes a feasible solution while minimizing (or maximizing) the objective function. To achieve this, we employ neural networks as search tools to explore the feasible solution space of the problem. Figure 1 illustrates the basic framework of using neural networks for solving the problem (1)-(3).

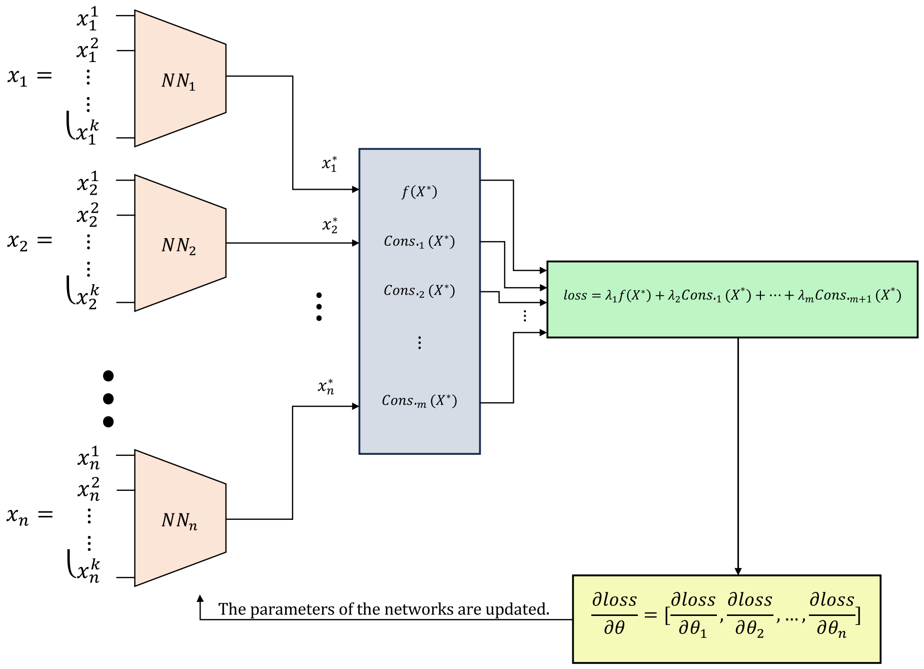
$x_1 =$
x_1^1
x_1^2
x_1^k
NN_1
$x_2 =$
x_2^1
x_2^2
x_2^k
NN_2
$x_n =$
x_n^1
x_n^2
x_n^k
NN_n
x_1^*
x_2^*
x_n^*
$f(X^*)$
$Cons._1(X^*)$
$Cons._2(X^*)$
$Cons._m(X^*)$
$loss = \lambda_1 f(X^*) + \lambda_2 Cons._1(X^*) + \cdots + \lambda_m Cons._{m+1}(X^*)$
$\frac{\partial loss}{\partial \theta} = [\frac{\partial loss}{\partial \theta_1}, \frac{\partial loss}{\partial \theta_2}, \ldots, \frac{\partial loss}{\partial \theta_n}]$
The parameters of the networks are updated.

Figure 1. The main framework of the presented method for solving Problem (1)-(3).

In Figure 1, each multilayer neural network, denoted as NN_j for $j = 1,\ldots,n$, is tasked with finding the optimal value for a specific entry of the input vector X. Each network receives discrete input values with a step size of h within the interval $[a,b]$. The i^{th} input of the j^{th} network NN_j is given by:

$$x_j^i = a + (i-1) \times h, \quad i = 1,\ldots,k,$$

where $k = \frac{b-a}{h}$. The output of each network NN_j is a real number corresponding to an entry in the input vector X. The goal of each network is to determine the optimal value, x_j^*, for its respective entry in the optimal input vector, $X^* = [x_1^*, x_2^*, x_3^*, \ldots, x_n^*]$. For example, Figure 2 illustrates the architecture of a fully-connected neural network with three hidden layers, containing 15, 20, and 15 nodes, respectively. This corresponds to the networks NN_j shown in Figure 1. Each element of the input vector X lies in the range $[0,1]$ with a step size of $h = 0.1$. More precisely, each network NN_jvreceives a fixed vector of uniformly spaced values in $[0,1]$, while the trainable parameters of the network encode the optimization process. The output of the network represents the optimized value of the variable x_j. This design transforms the neural network into a function-based search mechanism over the feasible interval, rather than a direct parametrization of x_j.
This representation has two advantages:

1. It provides stable gradients and avoids degeneracy associated with directly optimizing scalar latent variables.
2. It allows the incorporation of problem-dependent lower and upper bounds (derived analytically from BFRE structure) through output mapping, which significantly improves convergence speed and feasibility satisfaction.

All activation functions are sigmoid by producing outputs in the range $[0,1]$ [49] and making them well-suited for this research.

The loss function is constructed based on the outputs of the networks $NN_j, j = 1,\ldots,n$, which correspond to $x_j, j = 1,\ldots,n$ in the vector $X = [x_1, x_2, \ldots, x_n]$. To achieve this, the objective function $f(X)$ in the problem (1)-(3) and the constraints related to the problem, as shown in Eq. (4), are first evaluated. The loss function is then defined as follows:

$$\text{Loss} = \lambda_1 f(X) + \lambda_2 \text{Cons.}_1(X) + \cdots + \lambda_{m+1} \text{Cons.}_{m+1}(X), \tag{7}$$

where $\lambda_k, k = 1,\ldots,m+1$ are hyperparameters used for weights of the terms.

Minimizing the loss function in Eq. (7) yields the optimal solution $X^* = [x_1^*, x_2^*, \ldots, x_n^*]$. To achieve this, the parameters of the networks $NN_j, j = 1,\ldots,n$ are optimized using the Adam optimization algorithm in the PyTorch package. The hyperparameters $\lambda_k, k = 1,\ldots,m+1$

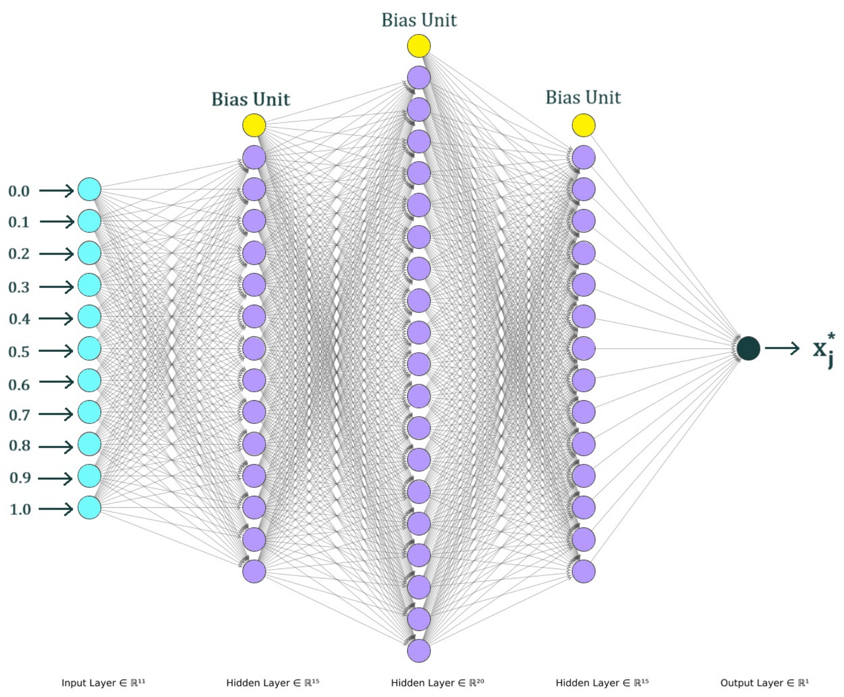

Figure 2. The architecture of a fully-connected neural network with three hidden layers corresponding to the networks NN_j shown in Figure 1.

regulate the penalty for constraint violations in Eq. (4). Based on our experiments, selecting appropriate values for these hyperparameters ensures that all the constraints in Eq. (4) are satisfied.

To obtain the optimal solution X^*, careful tuning of the learning rate η and the hyperparameters $\lambda_k, k = 1,\dots,m+1$, are essential. In subsequent sections, we will present the best solutions obtained using the proposed method for several examples, along with the optimal values for the learning rate and hyperparameters $\lambda_k, k = 1,\dots,m+1$, for each case.

As it has been mentioned in Section 2, we can determine upper and lower bound for the optimal variables of the problem. This capability is utilized to solve the examples presented in the next section. In the proposed method, the output activation function of each neural network is a sigmoid function. Since the sigmoid function outputs are values within the range $[0,1]$, the upper and lower bounds for the optimal variables can be applied as follows:

$$x_j = LB_j + o_{NN_j} \times (UB_j - LB_j),$$

where o_{NN_j} is the output of the neural network NN_j, constrained to $[0,1]$ by the sigmoid activation function, and LB_j and UB_j are the lower and upper bounds of the j-th entry in X^*, respectively.
The main novel aspects of this research are as follows:

1. Variable-wise neural decomposition: Each decision variable is represented by an independent neural network with its own parameter space, rather than using a single network to output the entire solution vector.

2. Analytically guided output constraints: Lower and upper bounds derived from the BFRE solution structure are embedded directly into the network output mapping.

3. Unified but decoupled optimization: While a single global loss function enforces objective minimization and constraint satisfaction, parameter updates occur independently within each network, improving stability and interpretability.

These features distinguish our approach from standard penalty-based neural optimization methods, which typically rely on a single multi-output network without exploiting analytical properties of the feasible domain.

5 Experimental Results

In this paper, we employ j neural networks to address a problem involving j unknown variables, denoted as $X = [x_1, x_2, \ldots, x_j]$. Each neural network NN$_j$, responsible for estimating x_j, comprises two hidden layers. The first hidden layer contains 20 nodes, while the second has 40 nodes. The output layer produces a value approximating x_j, and the input layer consists of 11 nodes.

As illustrated in Figure 2, the input layer receives discrete values defined by

$$x_j^i = a + (i-1)\cdot h, \quad i = 1, \ldots, k,$$

where $h = 0.1$. For all problems considered in this paper, the input values are restricted to the range $[0,1]$. Consequently, the input layer receives 11 equally spaced values:

$$x_j^i = (i-1)\cdot h, \quad i = 1, \ldots, 11.$$

All activation functions used in this work are Sigmoid functions ensuring that their output values lie within the interval $[0,1]$. When it is necessary, the output value of the final activation function, denoted as o_j, is mapped to a different interval $[\alpha, \beta]$ using the following transformation:

$$x_j = \alpha + o_j \cdot \beta. \tag{8}$$

In all examples, training is terminated when one of the following conditions is met:

1. The relative improvement of the loss function over a fixed window of epochs (typically 500 epochs) falls below a predefined threshold.

2. The objective value and constraint violation terms remain unchanged (up to numerical precision) for a sufficiently long sequence of epochs.

3. A predefined maximum number of epochs is reached.

In the reported examples, convergence was effectively detected by condition (2), where increasing the number of epochs did not yield further improvement in either the objective value or constraint satisfaction. Furthermore, all results were obtained on a personal computer equipped with an Intelő Core i7 CPU, 12 GB of RAM, using PyTorch version 2.5.0 (CPU-only).
Now, we are ready to illustrate the processes by the following examples.

Example 1. Consider the problem (1)-(3) using the following data.

$$\min_X f(x_1,x_2,x_3) = 3000x_1 + 1000x_1^3 + 2000x_2 + 666.667x_2^3, \tag{9}$$

$$A^+ = \begin{pmatrix} 0.44 & 0.24 & 0.44 \\ 0.22 & 0.98 & 0.46 \\ 0.55 & 0.21 & 0.42 \end{pmatrix},$$

$$A^- = \begin{pmatrix} 0.22 & 0.05 & 0.44 \\ 0.22 & 0.44 & 0.66 \\ 0.11 & 0.64 & 0.20 \end{pmatrix}, b = (0.44, 0.66, 0.5)^T,$$

$$x \in [0,1]^3.$$

Since in this example there are three constraints, the loss function associated with this problem is as follows:

$$\text{Loss} = \lambda_1 f(X) + \lambda_2 \text{Cons.}_1(X) + \lambda_3 \text{Cons.}_2(X) + \lambda_4 \text{Cons.}_3(X). \tag{10}$$

Table 1 presents the results obtained using the proposed method for various numbers of epochs. Based on our experiments, the values for the penalty coefficients λ_1, λ_2, λ_3, and λ_4, in Eq. (10) can be considered as 1, 10^3, 10^3, and 10^{17}, respectively, to satisfy the constraints of the problem (1)-(3).

Table 1. The performance of the proposed method for Example 1.

Epochs	x_1	x_2	x_3	$f(X)$	Time (S)
2000	1.2738107216136996e-05	0.5	0.07130065560340881	1083.37158203125	11.89
4000	2.7765538561652647e-06	0.5	0.07130065560340881	1083.3416748046875	28.26
6000	8.678634344505554e-07	0.5	0.07130065560340881	1083.3359375	36.28
8000	2.9791172551085765e-07	0.5	0.07130065560340881	1083.334228515625	48.25
10000	1.0648702897242401e-07	0.5	0.07130065560340881	1083.333740234375	59.60
12000	3.8740228802680576e-08	0.5	0.07130065560340881	1083.33349609375	73.67
14000	1.4118270108554043e-08	0.5	0.07130065560340881	1083.33349609375	86.17
14648	1.0162978014705004e-08	0.5	0.07130064070224762	1083.3333740234375	94.93

As shown in Table 1, the lowest value for $f(x)$ of Example 1 is as: 1083.3333740234375 at 14648 epochs. The results didn't improve by increasing the number of epochs beyond 14648. Moreover, all constraints specified in Eq. (9) are satisfied with the values reported in the last row of Table 1.

Figure 3 illustrates the loss values obtained for different epochs using the proposed method. As it is evident, the loss values generally exhibit a decreasing trend. Although the values fluctuate significantly during the initial epochs, they eventually converge to the minimum value of 1083.3333740234375 around epoch 14648.

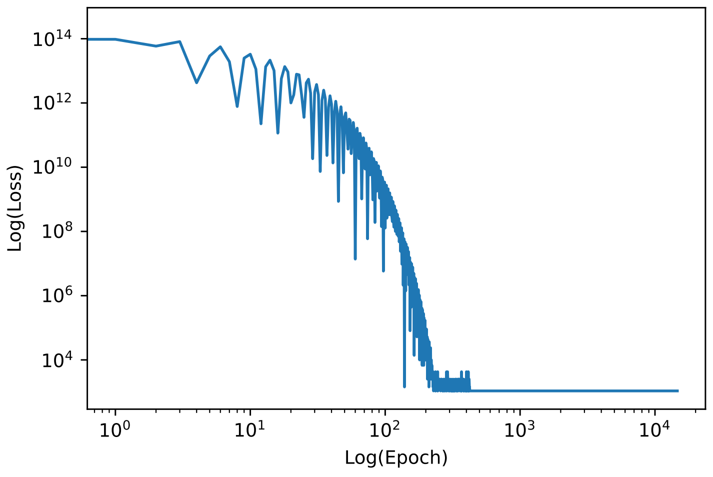

Figure 3. The loss values for epochs from 1 to 14648 for Example 1.

The problem defined by Example 1 was previously addressed using a genetic algorithm in [42]. The study in [42] achieved the lowest value for $f(x)$ with the genetic algorithm as 1083.333375. In comparison, the method proposed in this paper demonstrates a slight improvement over the genetic algorithm.

Example 2. Consider the problem (1)-(3) using the following data.

$$\min_{X} f(x_1,x_2,x_3,x_4) = (x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4, \tag{11}$$

$$A^+ = \begin{pmatrix} 0.5176 & 0.2278 & 0.8993 & 0.9858 \\ 0.1370 & 0.4585 & 0.6334 & 0.2790 \\ 0.4093 & 0.7399 & 0.0313 & 0.3039 \end{pmatrix},$$

$$A^- = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, b = (0.7208, 0.6334, 0.4725)^T,$$

$$x \in [0,1]^4.$$

Since this example involves three constraints, the loss function associated with the problem is defined as:

$$\text{Loss} = \lambda_1 f(X) + \lambda_2 \text{Cons.}_1(X) + \lambda_3 \text{Cons.}_2(X) + \lambda_4 \text{Cons.}_3(X). \tag{12}$$

Table 2 summarizes the results obtained using the proposed method for different numbers of epochs. From our experiments, the values of the penalty coefficients λ_1, λ_2, λ_3, and λ_4, in Eq. (12) can be considered as 1, 10^{10}, 10^2, and 10^{18}, respectively, to satisfy the constraints of the problem (1)-(3) in this example.

Table 2. The performance of the proposed method for Example 2.

Epochs	x_1	x_2	x_3	x_4	$f(X)$	Time (S)
5000	0.8374998569488525	0.4724999964237213	0.72079944610596	0.5143082141876221	32.1457138061523438	23.43
10000	0.8364351391792297	0.4724999964237213	0.72079926729202	0.5165360569953918	32.1249122619628906	48.51
15000	0.8232926130294800	0.4724999964237213	0.72080171108246	0.5387480258941650	31.8968467712402344	72.63
20000	0.6890978217124939	0.4724999964237213	0.72077339887619	0.6313853263854980	30.2343349456787109	97.68
25000	0.0331224314868450	0.4724999964237213	0.72083407640457	0.4448863565921783	24.1901931762695313	121.75
28477	0.0073700868524611	0.4724999964237213	0.72079998254776	0.4264936149120331	24.0189971923828125	141.69

As shown in Table 2, the lowest value of $f(x)$, as defined in the problem shown in Eq. (11), was found to be 24.0189971923828125 at 28,477 epochs. Increasing the number of epochs beyond 28,477 did not yield further improvements. Moreover, all constraints of this example were satisfied with the values reported in the last row of Table 2.

Figure 4 illustrates the loss values obtained over different epochs using the proposed method. As it is evident, the loss values exhibit a generally decreasing trend. Although significant fluctuations occur during the initial epochs, the loss eventually converges to the minimum value of 24.0189971923828125 around epoch 28,477.

The problem (1)-(3) of this example was previously addressed using a genetic algorithm and a modified imperialist competitive algorithm in [50]. The lowest values of $f(x)$ achieved in that study using the genetic algorithm and the modified imperialist competitive algorithm were reported as 24.0898 and 24.0200, respectively. In comparison, the method proposed in this paper demonstrates a clear and important improvement over both the genetic algorithm and the modified imperialist competitive algorithm.

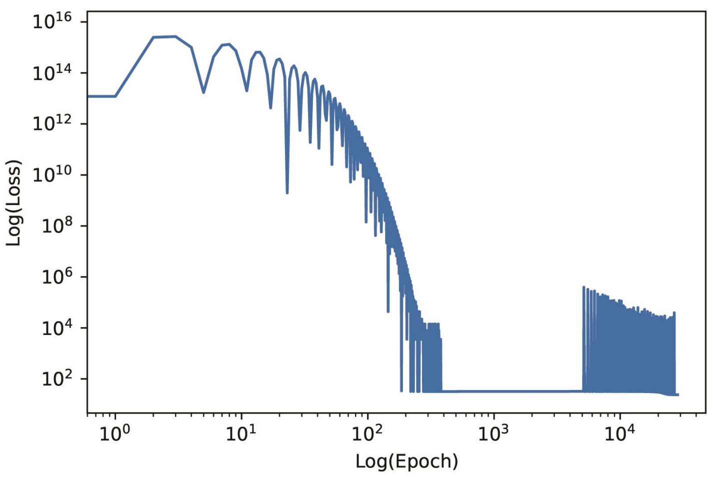

Figure 4. The loss values for epochs from 1 to 28477 for Example 2.

Example 3. Consider the problem (1)-(3) with the following data.

$$\min_X f(x_1,x_2,x_3,x_4,x_5) = x_1x_2x_3x_4x_5, \tag{13}$$

$$A^+ = \begin{pmatrix} 0.33 & 0.41 & 0.44 & 0.56 & 0.43 \\ 0.43 & 0.37 & 0.50 & 0.56 & 0.13 \\ 0.67 & 0.99 & 0.78 & 0.41 & 0.74 \\ 0.81 & 0.57 & 0.37 & 0.01 & 0.77 \end{pmatrix},$$

$$A^- = \begin{pmatrix} 0.32 & 0.50 & 0.43 & 0.64 & 0.12 \\ 0.24 & 0.87 & 0.24 & 0.02 & 0.01 \\ 0.56 & 0.13 & 0.89 & 0.91 & 0.08 \\ 0.11 & 0.23 & 0.45 & 0.33 & 0.02 \end{pmatrix}, b = (0.64, 0.69, 0.88, 0.45)^T,$$

$$x \in [0,1]^5.$$

In this example, we further restrict the range of variations of each variable of the problem (1)-(3) of this example by the rules outlined in Section 2. To accelerate the search for optimal values using neural networks, we first compute the lower and upper bounds for each x_j, $j = 1,2,3,4,5$, as follows. The upper bound of the solution set of each constraint in this example ia as follows:

$$g^{1+} = (1,1,1,1,1)^T, \qquad g^{2+} = (1,1,1,1,1)^T, \qquad g^{3+} = (1,0.88,1,1,1)^T, \qquad g^{4+} = (0.45,0.45,1,1,0.45)^T.$$

The overall upper bound of the solution set of the constraints of the problem (1)-(3) is computed as:

$$g^+ = \inf_{i=1,\dots,4} g^{i+} = (0.45,0.45,1,1,0.45)^T.$$

Furthermore, the lower bound of the solution set of each constraint for the problem (1)-(3) in this example is as follows:

$$s^{1-} = (0,0,0,0,0)^T, \qquad s^{2-} = (0,0.31,0,0,0)^T, \qquad s^{3-} = (0,0,0.12,0.12,0)^T, \qquad s^{4-} = (0,0,0,0,0)^T.$$

The overall lower bound of the solution set of the constraints of the problem (1)-(3) in this example is computed as:

$$s^- = \sup_{i=1,\dots,4} s^{i-} = (0,0.31,0.12,0.12,0)^T.$$

Based on s^- and g^+, we can consider the following intervals for each x_j $j = 1,2,3,4,5$ as follows:

$$x_1 \in [0,0.45], \quad x_2 \in [0.31,0.45], \quad x_3 \in [0.12,1], \quad x_4 \in [0.12,1], \quad x_5 \in [0,0.45].$$

The output values of the neural networks are then mapped to these corresponding intervals using the formula described in Eq. (8).

Since this example involves four constraints, the loss function for the problem is defined as:

$$\text{Loss} = \lambda_1 f(X) + \lambda_2 \text{Cons.}_1(X) + \lambda_3 \text{Cons.}_2(X) + \lambda_4 \text{Cons.}_3(X) + \lambda_5 \text{Cons.}_4(X). \tag{14}$$

Table 3 summarizes the results obtained using the proposed method for different numbers of epochs. From the experiments, the values of the penalty coefficients λ_1, λ_2, λ_3, λ_4, and λ_5 in Eq. (14) are determined to be 20, 1, 1, 1, and 1, respectively, to satisfy the constraints of the problem (1)-(3) in this example.

Table 3. The performance of the proposed method for Example 3.

Epochs	x_1	x_2	x_3	x_4	x_5	$f(X)$	Time (S)
10	3.3874243047e-15	0.3100000024	0.1199999973	0.1199999973	9.2464379688e-17	1.3981966229e-33	1.16
20	6.6535027373e-19	0.3100000024	0.1199999973	0.1199999973	7.0353512477e-21	2.0896162700e-41	1.51
30	2.5227932702e-20	0.3100000024	0.1199999973	0.1199999973	1.8544149623e-22	2.1019476965e-44	2.11
40	6.9627031411e-21	0.3100000024	0.1199999973	0.1199999973	4.4359340018e-23	1.4012984643e-45	2.43
50	4.2146920161e-21	0.3100000024	0.1199999973	0.1199999973	2.5395226409e-23	0.0	2.54

As shown in Table 3, the lowest value of $f(x)$ in Eq. (13) is 0.0, at 50 epochs. All constraints of Example 3 are satisfied with values reported in the last row of Table 3.

Figure 5 illustrates the loss values for different epochs using the proposed method. The loss values exhibit a steady decreasing trend, highlighting the method's effectiveness.

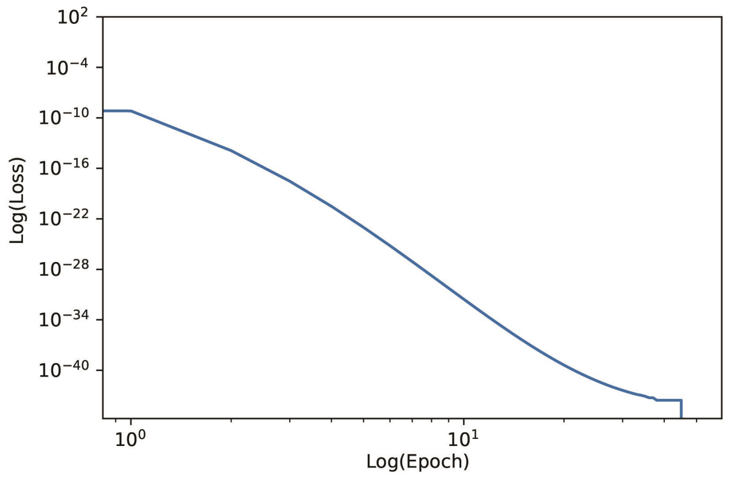

Figure 5. The loss values for epochs from 1 to 50 for Example 3.

The problem (1)-(3) in this example was previously addressed using a Genetic Algorithm (GA) and a Modified Imperialist Competitive Algorithm (MICA) in [50]. The lowest values of $f(x)$ achieved in the study were as 2.5043×10^{-21} using the genetic algorithm and 0.0 using the modified imperialist competitive algorithm, respectively. Compared to these, the proposed method demonstrates improved accuracy over the genetic algorithm and significantly better execution time than both meta-heuristic algorithms. Specifically, the execution times for the genetic algorithm and the modified imperialist competitive algorithm were 162.71 seconds and 101.577 seconds, respectively, while the proposed neural network method finds the optimal solution in just 2.54 seconds. These points highlight the superior efficiency and accuracy of the proposed method with respect to GA and MICA.

Example 4. Consider the problem (1)-(3) with the following data.

$$\max_X f(x_1,x_2,x_3,x_4,x_5,x_6) = e^{x_1+x_3-x_4} + \sin(x_2+x_5+x_6), \tag{15}$$

$$A^+ = \begin{pmatrix} 0.15 & 0.75 & 0.25 & 0.12 & 0.28 & 0.37 \\ 0.35 & 0.85 & 0.41 & 0.8 & 0.35 & 0.15 \\ 0.7 & 0.22 & 0.25 & 0.28 & 0.12 & 0.2 \\ 0.9 & 0.51 & 0.85 & 0.4 & 0.2 & 1 \\ 0.1 & 0.11 & 0.21 & 0.25 & 0.8 & 0.65 \end{pmatrix},$$

$$A^- = \begin{pmatrix} 0.2 & 0.23 & 0.51 & 0.3 & 0.7 & 0.15 \\ 0.4 & 0.8 & 0.6 & 0.4 & 0.32 & 0.16 \\ 0.4 & 0.9 & 0.5 & 0.65 & 0.4 & 0.24 \\ 0.14 & 0.18 & 0.65 & 0.54 & 0.38 & 0.6 \\ 0.45 & 0.5 & 0.4 & 0.83 & 0.6 & 0.4 \end{pmatrix}, b = (0.51, 0.6, 0.65, 0.8, 0.83)^T,$$

$$x \in [0,1]^6.$$

In this example, we further restrict the range of variations of each variable of the problem (1)-(3) of this example by the rules outlined in Section 2. To accelerate the search for optimal values using neural networks, we firstly compute the lower and upper bounds for each x_j, $j = 1,2,3,4,5,6$. To do this, the upper bound of the solution set of each constraint of this example is computed as follows:

$$g^{1+} = (1, 0.51, 1, 1, 1, 1)^T, \quad g^{2+} = (1, 0.6, 1, 0.6, 1, 1)^T, \quad g^{3+} = (0.65, 1, 1, 1, 1, 1)^T,$$
$$g^{4+} = (0.8, 1, 0.8, 1, 1, 0.8)^T, \quad g^{5+} = (1, 1, 1, 1, 1, 1)^T.$$

The overall upper bound g^+ of the solution set of the feasible domain of the problem (1)-(3) is computed using the element-wise infimum across all g^{i+}, for $i = 1,2,3,4,5$, as follows:

$$g^+ = \inf_{i=1,\dots,5} g^{i+} = (0.65, 0.51, 0.8, 0.6, 1, 0.8)^T.$$

Similarly, the lower bounds are defined as:

$$g^{1-} = (0, 0, 0, 0, 0.49, 0)^T, \quad g^{2-} = (0, 0.4, 0, 0, 0, 01)^T, \quad g^{3-} = (0, 0.35, 0, 0, 0, 0)^T,$$
$$g^{4-} = (0, 0, 0, 0, 0, 0)^T, \quad g^{5-} = (0, 0, 0, 0, 0, 0)^T.$$

The overall lower bound s^- is computed using the element-wise supremum across all s^{i-}, for ($i = 1,2,3,4,5$), as follows:

$$s^- = \sup_{j=1,\dots,5} s^{j-} = (0, 0.4, 0, 0, 0.49, 0)^T.$$

Based on the values of s^- and g^+, the intervals associated to each x_j, for $j = 1,2,3,4,5,6$, are determined as follows:

$$x_1 \in [0, 0.65], \quad x_2 \in [0.4, 0.51], \quad x_3 \in [0, 0.8],$$
$$x_4 \in [0, 0.60], \quad x_5 \in [0.49, 1.00], \quad x_6 \in [0, 0.8].$$

The output values of the neural networks are then mapped to these corresponding intervals using the formula described in Eq. (8).

This example is a maximization problem subject to five constraints. Since the ADAM optimizer in the PyTorch package minimizes the objective function, we must reformulate the original maximization problem as a minimization problem. Accordingly, the loss function is defined as:

$$\text{Loss} = \kappa - \left[\lambda_1 f(X) + \lambda_2 \text{Cons.}_1(X) + \lambda_3 \text{Cons.}_2(X) + \lambda_4 \text{Cons.}_3(X) + \lambda_5 \text{Cons.}_4(X) + \lambda_6 \text{Cons.}_5(X)\right]. \tag{16}$$

In Equation (16), $\kappa \in \mathbb{R}$ denotes the maximum value of $f(X)$ when the constraints are not considered. In this example, $\kappa \approx 8.3891$.

Table 4. The performance of the proposed method for Example 4.

Epochs	x_1	x_2	x_3	x_4	x_5	x_6	$f(X)$	Time (S)
25	0.64999998	0.40000004	0.80000001	1.10140641e-09	0.49000001	0.76980329	5.25915622	1.34
50	0.64999998	0.40000001	0.80000001	2.40665071e-10	0.49000001	0.67097926	5.26306677	2.52
75	0.64999998	0.40000001	0.80000001	2.08984191e-10	0.49000001	0.67648107	5.26310539	3.07
100	0.64999998	0.40000001	0.80000001	2.06451370e-10	0.49000001	0.67968005	5.26311445	3.44
125	0.64999998	0.40000001	0.80000001	2.06240816e-10	0.49000001	0.68090981	5.26311493	4.05

Table 4 summarizes the results obtained using the proposed method for different numbers of epochs. Through experimentations, the values of the penalty coefficients λ_1, λ_2, λ_3, λ_4, λ_5, and λ_6 in Eq. (16) are determined as 10, 1000, 100, 1, 1, and 1, respectively.

As shown in Table 4, the highest value of $f(x)$, as defined in Eq. (15), is 5.26311493, at 125 epochs. All constraints specified in this example are satisfied with the corresponding values reported in the last row of Table 4.

Figure 6 illustrates the loss values across different epochs using the proposed method. The loss exhibits a steady decreasing trend which highlighting the effectiveness of the approach in converging to the optimal values. The problem (1)-(3) in this example was previously

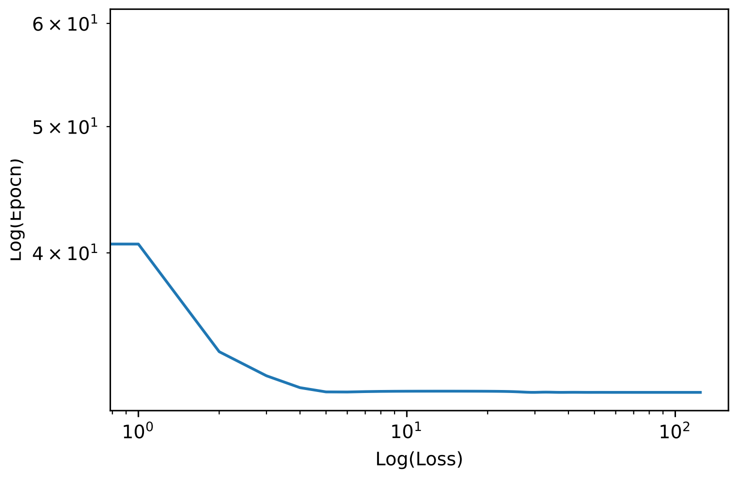

Figure 6. The loss values for epochs from 1 to 50 for Example 4.

addressed using a genetic algorithm in [42], where the highest reported value for $f(x)$ was 4.116427. In comparison, the proposed method in this paper achieved a significantly higher value of 5.26311493. This method suggests an very important improvement over the genetic algorithm.

6 Discussion

According to the results reported in Section 5, the proposed neural network architecture shows a high capability for solving the problems presented in Examples 1 to 4. By assigning a separate network to each variable x and employing a unified loss function with penalty coefficients, each network maintained an independent parameter space. The Adam optimization algorithm updated these parameter spaces individually, resulting in highly accurate solutions.

Our experiments revealed that the settings of various hyperparameters significantly impact on the accuracy of the solutions. The key hyperparameters in this study included the penalty coefficients, learning rate, and activation functions.

Incorrect penalty values often led to the violation of constraints. To identify optimal penalty values for each example, we trained the networks several times using different penalty configurations and reported the best performing values. Using separate networks for each variable provided flexibility as follows. When a constraint was violated, we could focus on the corresponding network and fine-tune its hyperparameters, leading to accurate results.

The learning rate of the Adam optimization algorithm also played a crucial role in solution accuracy. Based on our experiments, the optimal learning rates for Examples 1, 2, 3, and 4 were 0.01, 0.01, 0.1, and 0.1, respectively. In Examples 3 and 4, we further restricted the search range of variables. This simplification allowed the Adam optimizer to converge to a solution more efficiently, even with a higher learning rate. Conversely, for Examples 1 and 2, where the problems had lower dimensions, the networks converged to accurate solutions without requiring search range restrictions.

Figures 3, 4, 5, and 6 show that reducing the dimensions and further restricting the search domain significantly accelerated the convergence of the networks. Without these procedures, the Adam optimizer struggled to find accurate solutions for Examples 3 and 4. However, for Examples 1 and 2, the problems with the lower dimensions ensured good convergence even without these adjustments.

Using n independent networks offers the following advantages over a single multi-output network:

1. Reduced interference between variables, particularly important in highly non-convex feasible domains induced by BFRE constraints.
2. Improved numerical stability, as each network adapts locally to constraint violations related to its corresponding variable.
3. Greater flexibility in hyperparameter tuning, allowing targeted adjustments when specific constraints are violated.

While a single multi-output network is feasible, our experiments indicate that the proposed decomposed architecture converges faster and more reliably for BFRE-constrained problems.

Finally, the choice of activation functions significantly influenced the networks' convergence speed. The best results were obtained when sigmoid activation functions were used in all layers.

7 Conclusions

This study introduces a novel methodology employing neural networks to address nonlinear optimization problems constrained by bipolar max-min fuzzy relation equations. The proposed approach effectively integrates the constraints and objective function into a unified loss function, enabling simultaneous optimization. Compared to traditional methods and metaheuristic algorithms, such as genetic algorithms and modified imperialist competitive algorithms, the neural network-based framework demonstrates superior accuracy and efficiency in solving highly complex and non-convex problems.

Through extensive experimentations for various examples, this method consistently achieved high precision in satisfying constraints and optimizing the objective function. The results highlight the scalability of the neural network architecture, which effectively adapts to problems of varying dimensionality. By leveraging the independent parameter space of each neural network and optimizing via the Adam algorithm, the approach ensures robust performance, even for problems with intricate solution domains. Additionally, the flexibility to fine-tune hyperparameters, such as penalty coefficients and learning rates, enhances its practical applicability.

The findings suggest that neural networks hold significant potential as a computational tool for solving nonlinear optimization problems with fuzzy constraints, particularly in domains where conventional techniques face limitations due to computational complexity or suboptimal convergence. Future research could explore extensions to more generalized fuzzy relational systems or incorporate advanced neural architectures to further enhance performance.

This paper establishes a solid foundation for employing neural networks in complex optimization scenarios, paving the way for their application in diverse fields such as engineering design, decision-making, and resource allocation. The demonstrated benefits of precision, efficiency, and scalability underscore the potential of neural network-based solutions to revolutionize the field of nonlinear programming under fuzzy constraints. In future research directions, (1) we will use the combination of the SVM method and the colonialist competitive algorithm to solve the problem (1)-(3) for linear and nonlinear cases. (2) we will focus on the extension of neural networks to solve the optimization problems with fuzzy relation inequality constraints. (3) we will study some new applications of the problem (1)-(3) in the area of digital data service.

Authors' Contributions

Ali Abbasi Molai, Conceptualization, Writing the original draft, Reviewing, Editing, and Investigation; **Hassan Dana Mazraeh**, Writing the original draft, Reviewing, Editing, Conceptualization, Methodology, Programming, and Investigation; **Kourosh Parand**, Editing, Methodology, and Investigation.

Data Availability

The manuscript has no associated data or the data will not be deposited.

Conflicts of Interest

The authors declare no competing interests.

Ethical Considerations

The authors have diligently addressed ethical concerns, such as informed consent, plagiarism, data fabrication, misconduct, falsification, double publication, redundancy, submission, and other related matters.

Funding

This research did not receive any grant from funding agencies in the public, commercial, or nonprofit sectors.

References

[1] E. Sanchez, Resolution of composite fuzzy relation equations, 30, 38–48, (1976).

[2] S.-C. Fang and G. Li, Solving fuzzy relation equations with a linear objective function, 103, 107–113, (1999).

[3] L. Jian-Xin, A new algorithm for minimizing a linear objective function with fuzzy relation equation constraints, 159, 2278–2298, (2008).

[4] Y.-K. Wu and S.-M. Guu, Minimizing a linear function under a fuzzy maxmin relational equation constraint, 150, 147–162, (2005).

[5] F.-F. Guo and Z.-Q. Xia, An algorithm for solving optimization problems with one linear objective function and finitely many constraints of fuzzy relation inequalities, 5, 33–47, (2006).

[6] S.-M. Guu and Y.-K. Wu, Minimizing a linear objective function with fuzzy relation equation constraints, 1, 347–360, (2002).

[7] Y.-K. Wu, S.-M. Guu, and J.-C. Liu, An accelerated approach for solving fuzzy relation equations with a linear objective function, IEEE Trans. Fuzzy Syst., 10(4), 552–558, (2002).

[8] J. Lu and S.-C. Fang, Solving nonlinear optimization problems with fuzzy relation equations constraints, 119, 1–20, (2001).

[9] E. Khorram and R. Hassanzadeh, Solving nonlinear optimization problems subjected to fuzzy relation equation constraints with max-average composition using a modified genetic algorithm, 55, 1–14, (2008).

[10] R. Hassanzadeh, E. Khorram, I. Mahdavi, and N. Mahdavi-Amiri, A genetic algorithm for optimization problems with fuzzy relation constraints using max-product composition, 11, 551–560, (2011).

[11] Y.-K. Wu, S.-M. Guu, and J.-C. Liu, Reducing the search space of a linear fractional programming problem under fuzzy relational equations with max-Archimedean t-norm composition, 159, 3347–3359, (2008).

[12] P. Li and S.-C. Fang, Minimizing a linear fractional function subject to a system of sup-t equations with a continuous archimedean triangular norm, 22, 49–62, (2009).

[13] A. Abbasi Molai, A new algorithm for resolution of the quadratic programming problem with fuzzy relation inequality constraints, 72, 306–314, (2014).

[14] B. Hedayatfar, A. Abbasi Molai, and S. Aliannezhadi, Separable programming problems with the max-product fuzzy relation equation constraints, 16, 1–15, (2019).

[15] X. Yang, X.-G. Zhou, and B.-Y. Cao, Latticized linear programming subject to max-product fuzzy relation inequalities with application in wireless communication, 358–359, 44–55, (2016).

[16] X.-G. Zhou, B.-Y. Cao, and X. Yang, The set of optimal solutions of geometric programming problem with max-product fuzzy relational equations constraints, 18, 436–447, (2016).

[17] Z. Wang, G. Zhu, and X. Yang, Tri-composed fuzzy relation inequality with weighted-max-min composition and the relevant min-max optimization problem, 489, 109011, (2024).

[18] J. Qiu, G. Li, and X. Yang, Weighted-minimax programming subject to the rta max-product fuzzy relation inequality system and its optimal strong solution, 478, 108824, (2024).

[19] X. Yang, Random-term-absent addition-min fuzzy relation inequalities and their lexicographic minimum solutions, 440, 42–61, (2022).

[20] J. Qiu, G. Li, and X. Yang, Arbitrary-term-absent max-product fuzzy relation inequalities and its lexicographic minimal solution, 567, 167–184, (2021).

[21] X. Yang, J. Qiu, H. Guo, and X. Yang, Fuzzy relation weighted minimax programming with addition-min composition, 147, 106644, (2020).

[22] H. Lin and X. Yang, Dichotomy algorithm for solving weighted min-max programming problem with addition-min fuzzy relation inequalities constraint, 146, 106537, (2020).

[23] X.-G. Zhou, X. Yang, and B.-Y. Cao, Posynomial geometric programming problem subject to maxmin fuzzy relation equations, 328, 15–25, (2016).

[24] X. Yang, X.-G. Zhou, and B.-Y. Cao, Single-variable term semi-latticized fuzzy relation geometric programming with max-product operator, 325, 271–287, (2015).

[25] A. Ghodousian and A. Babalhavaeji, An efficient genetic algorithm for solving nonlinear optimization problems defined with fuzzy relational equations and max-lukasiewicz composition, 69, 475–492, (2018).

[26] A. Ghodousian, M. Naeeimi, and A. Babalhavaeji, Nonlinear optimization problem subjected to fuzzy relational equations defined by dubois-prade family of t-norms, 119, 167–180, (2018).

[27] A. Thapar, D. Pandey, and S. Gaur, Optimization of linear objective function with max-t fuzzy relation equations, 9, 1097–1101, (2009).

[28] A. Ghodousian and M. Raeisian Parvari, A modified pso algorithm for linear optimization problem subject to the generalized fuzzy relational inequalities with fuzzy constraints (fri-fc), 418-419, 317–345, (2017).

[29] S. Freson, B. De Baets, and H. De Meyer, Linear optimization with bipolar max-min constraints, 234, 3–15, (2013).

[30] P. Li and Q. Jin, Fuzzy relational equations with min-biimplication composition, 11, 227–240, (2012).

[31] P. Li and Y. Liu, Linear optimization with bipolar fuzzy relational equation constraints using the Lukasiewicz triangular norm, 18, 1399–1404, (2014).

[32] C.-C. Liu, Y.-Y. Lur, and Y.-K. Wu, Linear optimization of bipolar fuzzy relational equations with max-lukasiewicz composition, 360, 149–162, (2016).

[33] X. Yang, Resolution of bipolar fuzzy relation equations with max-ukasiewicz composition, 397, 41–60, (2020).

[34] S. Aliannezhadi, A. Abbasi Molai, and B. Hedayatfar, Linear optimization with bipolar max-parametric hamacher fuzzy relation equation constraints, Kybernetika, 52(4), 531–557, (2016).

[35] S. Aliannezhadi and A. Abbasi Molai, A new algorithm for solving linear programming problems with bipolar fuzzy relation equation constraints, Iran. J. Numer. Anal. Optim., 11(2), 407–435, (2021).

[36] A. Abbasi Molai, A covering-based algorithm for resolution of linear programming problems with max-product bipolar fuzzy relation equation constraints, J. Math. Model., 11(4), 709–730, (2023).

[37] A. Abbasi Molai, A quadratic optimization problem with bipolar fuzzy relation equation constraints, Iran. J. Fuzzy Syst., 19(6), 125–140, (2022).

[38] S. Aliannezhadi and A. Abbasi Molai, A new algorithm for geometric optimization with a single-term exponent constrained by bipolar fuzzy relation equations, Iran. J. Fuzzy Syst., 18(1), 137–150, (2021).

[39] S. Aliannezhadi, S. Ardalan, and A. Abbasi Molai, Maximizing a monomial geometric objective function subject to bipolar max-product fuzzy relation constraints, J. Intell. Fuzzy Syst., 32(1), 337–350, (2017).

[40] J. Zhou, Y. Yu, Y. Liu, and Y. Zhang, Solving nonlinear optimization problems with bipolar fuzzy relational equation constraints, J. Inequalities Appl., 32(1), 337–350, (2016).

[41] A. Ghodousian and Z. Sara, A two - phase - aco algorithm for solving nonlinear optimization problems subjected to fuzzy relational equations, Iran. J. Fuzzy Syst., 21(5), 151–174, (2024).

[42] H. Dana Mazraeh and A. Abbasi Molai, Resolution of nonlinear optimization problems subject to bipolar max-min fuzzy relation equation constraints using genetic algorithm, Iran. J. Fuzzy Syst., 15(2), 109–131, (2018).

[43] M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, 378, 686–707, (2019).

[44] H. Dana Mazraeh and K. Parand, A three-stage framework combining neural networks and monte carlo tree search for approximating analytical solutions to the thomasfermi equation, 87, 102582, (2025).

[45] H. Dana Mazraeh and K. Parand, Approximate symbolic solutions to differential equations using a novel combination of monte carlo tree search and physics-informed neural networks approach, 87, (2025).

[46] H. Dana Mazraeh and K. Parand, An innovative combination of deep q-networks and context-free grammars for symbolic solutions to differential equations, 142, 109733, (2025).

[47] H. Dana Mazraeh and K. Parand, Gepinn: An innovative hybrid method for a symbolic solution to the laneemden type equation based on grammatical evolution and physics-informed neural networks, 48, 100846, (2024).

[48] S. Raschka, Y. H. Y. (H.) Liu, and V. Mirjalili, Machine Learning with PyTorch and Scikit-Learn Develop machine learning and deep learning models with Python. Birmingham: Packt Publishing, 2022.

[49] C. C. Aggarwal, Neural Networks and Deep Learning. Springer Nature Switzerland: Springer Cham, 2023.

[50] A. Abbasi Molai and H. Dana Mazraeh, A modified imperialist competitive algorithm for solving nonlinear programming problems subject to mixed fuzzy relation equations, Int. J. Nonlinear Anal. Appl., 14(3), 19–32, (2023).