Filomat 37:21 (2023), 7057–7075
https://doi.org/10.2298/FIL2321057K

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

The admissible portfolio selection problem with transaction costs and a neural network scheme

Raziye Keyshams[a], Alireza Nazemi[a,*]

[a] *Faculty of Mathematical Sciences, Shahrood University of Technology, P.O. Box 3619995161-316, Tel-Fax No:+9823-32300235, Shahrood, Iran.*

Abstract. In this paper, we study the portfolio optimization model with transaction costs under the assumption that there exist admissible errors on expected returns and risks of assets. We propose an admissible efficient portfolio selection problem and design a neural network for the proposed problem. The presented neural network framework guarantees to obtain the optimal solution of the admissible portfolio selection problem. The existence and convergence of the trajectories of the network are studied. The Lyapunov stability and globally convergence of the considered neural network are also shown. We provide a numerical example to illustrate the proposed effective approach.

1. Introduction

In 1952, Markowitz [1] studied his pioneering work which established the basement of modern portfolio analysis. Markowitz's mean–variance model has visited as a foundation for the development of modern financial theory over the past five decades. The main aim of this problem is to maximize the expected return for a certain level of risk or minimize the risk for a certain return. But both theory and practice indicate that variance is not a good risk measure (see Artzneretal. [2, 3]). Therefore, many scholars seek new risk measures for portfolio selection and risk management. With the continuous attempt of various researchers, Markowitz's seminal work has been widely continued. Since then, a large number of papers on portfolio optimization problem have appeared in the literature. Furthermore, various methods have been proposed for the modeling of investment risk. For example, see [4]-[13].

It is well known that transaction cost is one of the principal concerns for portfolio managers. It has an important effect on the portfolio optimization and the frequency of time rebalancing the portfolio. There are some reports on this subject (see [14]–[20]). Among them, Arnott and Wagner [14] established that ignoring transaction costs would result in an inefficient portfolio. The experimental analysis done by Yoshimoto [15] also confirmed this fact. Babazadeh and Esfahanipour in [16] was developed a novel portfolio optimization model in which Value at Risk (VAR) is utilized as a risk measure to account extreme risk so that VaR is estimated use of Extreme Value Theory. An enhanced Non-dominated Sorting Genetic Algorithm-II is developed by utilizing a new repairing mechanism to solve the obtained portfolio problem. Meghwani and

2020 *Mathematics Subject Classification*. Primary 91G10; Secondary 68T07, 90C20, 34Dxx, 40Axx
Keywords. Portfolio selection, Admissible return, Admissible risk, Transaction costs, Quadratic programming, Neural network, Stability, Convergence
Received: 09 April 2021; Revised: 07 March 2022; Accepted: 14 March 2023
Communicated by Predrag Stanimirović
* Corresponding author: Alireza Nazemi
Email addresses: raziyekeyshams@gmail.com (Raziye Keyshams), nazemi20042003@gmail.com (Alireza Nazemi)

Thakur in [17] proposed a tri-objective portfolio optimization model with risk, return and transaction cost as the objectives, where three different models with different risk measures–variance, VaR and conditional VaR are discussed. They adapted three multi-objective evolutionary algorithms for the proposed models. Simos et al. in [18] define and studied the time-varying Black–Litterman portfolio optimization under nonlinear constraints (TV-BLPONC) problem as a nonlinear programming (NLP) problem, where the nonlinear constraints refer to transaction costs and cardinality constraints. Using the beetle antennae search (BAS) algorithm, they introduced a computational method to solve the TV-BLPONC problem. Katsikis et al. in [19] defined the time-varying mean-variance portfolio selection under transaction costs and cardinality constraint problem as a time-varying NLP problem. Using the BAS algorithm, they also provided an online solution to the NLP problem. Katsikis et al. in [20] defined and studied the time-varying minimum-cost portfolio insurance under transaction costs problem in the form of a time-varying NLP problem. Using the BAS algorithm, they also provided an online solution to the static NLP problem. By adding the transaction costs to the classical minimum cost portfolio insurance problem, Katsikis and Mourtas in [21] defined and studied the minimum cost portfolio insurance under transaction costs problem as a NLP problem. They use the BAS algorithm to provide a solution to the NLP problem.

When some more realistic constraints such as transaction costs, bounded constraints, liquidity constraints, minimum transaction lots constraints, and cardinality constraints are studied, the portfolio selection problem becomes a complex programming problem and common optimization algorithms fail to find the optimal solution efficiently. Therefore, many researchers solve the complex constrained portfolio problems by using heuristic algorithms, see [22]-[35]. Among utilizing of heuristics in the new researches, using real-world data sets on a binary Markowitz-based portfolio selection problem, Mourtas and Katsikis in [22] validated the excellent performance of V-shaped transfer function-based binary BAS on large input data and demonstrated that it is a marvelous alternative against other ordinary memetic meta-heuristic optimization algorithms. By converting the classical minimum-cost portfolio insurance problem to a multi-period minimum-cost portfolio insurance problem, Katsikis and Mourtas in [23] defined and investigated the obtained problem under transaction costs problem as a nonlinear programming problem. They introduced a well-tuned approach that can solve the given problem. Katsikis and Mourtas in [24] defined a binary optimal tangency portfolio under cardinality constraint problem and studied it as a NLP problem. They employed a binary BAS algorithm to provide a solution to the NLP problem. Medvedeva et al. in [25] described and analyzed the randomized time-varying knapsack problem (RTVKP), which is applied in the field of finance and converted into a portfolio insurance problem, as a time-varying integer linear programming problem. In this way, they presented the on-line solution to the RTVKP combinatorial optimization problem and highlight the restrictions of static methods. Katsikis and Mourtas in [26] presented the ORPIT Matlab toolbox which applies the theory of vector lattices to solve (a) the problem of option replication and (b) the cost minimization problem of portfolio insurance as well as related sub problems. Katsikis and Mourtas in [27] proposed an algorithmic process that finds the minimum-cost insured portfolio in the case where the space of marketed securities is a subspace of $C[a, b]$. They proposed a heuristic method for computing appropriate intervals $[a, b]$, so that the existence of a positive basis is guaranteed.

Over the years, neural networks for optimization (see for example [38]–[54]) and their engineering applications have been widely investigated. These neural networks are essentially governed by a set of dynamic systems characterized by an energy function, which is the combination of the objective function and constraints of the original optimization problem, and three common techniques, such as penalty functions, Lagrange functions and primal and dual functions. In addition, neural networks for solving optimization problems are hardware-implementable; that is, the neural networks can be implemented by using integrated circuits. Compared with traditional numerical optimization algorithms, the neural network approach has several potential advantages. First, the structure of a neural network can be implemented effectively using very large scale integration and optical technologies. Second, neural networks can solve many optimization problems with time-varying parameters. Third, the dynamical techniques and the numerical ODE techniques can be applied directly to the continuous-time neural network for solving constrained optimization problems effectively. Fourth, the neural networks have fast convergence rate in real-time solutions. Therefore, neural network methods for optimization have been received considerable attention. For the mentioned reasons, in this paper, we will examine the admissible portfolio selection problem with

transaction costs and bounded constraints using a high performance neural network method with good stability properties and convergence results. An attractive feature of the neural network method described in this paper is that the starting point need not be in the feasible region. It should be noted that nevertheless, this admissible portfolio selection problem with transaction costs and bounded constraints has not yet been well established by a neural network scheme.

The organization of this paper is as follows. The admissible portfolio selection model with transaction costs and bounded constraints is presented in Section 2. A neural network algorithm is designed to solve the corresponding portfolio problem in Section 3. Comparing with some other models is stated in Section 4. A numerical example is given to illustrate our proposed effective means and approaches in Section 5. Some concluding remarks are given in Section 6.

2. The admissible portfolio selection model with transaction costs

We consider a portfolio selection problem with n risky assets. Let r_j be the expected return rate of asset j and let x_j be the proportion of capital to be invested in asset j, $j = 1, 2, \ldots, n$. In order to describe conveniently, we set $x = (x_1, x_2, \ldots, x_n)^T$, $r = (r_1, r_2, \ldots, r_n)^T$, $e = (1, 1, \ldots, 1)^T$. Then the expected return and variance associated with the portfolio $x = (x_1, x_2, \ldots, x_n)^T$ are, respectively, given by

$$E(r) = r^T x, \quad D(r) = x^T v x,$$

where $v = (\sigma_{ij})_{n\times n}$ is the covariance matrix of expected returns.

Markowitz's mean-variance model of the portfolio selection problem may be described by the following quadratic programming:

$$\begin{aligned} &\text{minimize} && x^T v x \\ &\text{subject to} && r^T x = r_0, \\ &&& e^T x = 1, \\ &&& x \geq 0. \end{aligned} \tag{1}$$

In order to apply the model (1) in a practical investment problem, we need to estimate r and $v = (\sigma_{ij})_{n\times n}$. Let the observation data on returns of assets over m periods be given. At the discrete time k ($k = 1, 2, \ldots, m$), n kinds of returns are denoted as a vector $r_k = (r_{k1}, r_{k2}, \ldots, r_{kn})^T$. The expected return $\bar{r} = (\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n)^T$ and covariance $\bar{v} = (\bar{\sigma}_{ij})_{n\times n}$ are given as follows

$$\bar{r}_j = \frac{\sum_{k=1}^{m}(h_{kj} r_{kj})}{\sum_{k=1}^{m} h_{kj}}, \quad j = 1, 2, \ldots, n, \tag{2}$$

$$\bar{\sigma}_{ij} = \frac{\sum_{k=1}^{m}(r_{ki} - \bar{r}_i)(r_{kj} - \bar{r}_j) h_{kij}}{\sum_{k=1}^{m} h_{kij}}, \quad i, j = 1, 2, \ldots, n, \tag{3}$$

where h_{kj} is a possibility grade to reflect a similarity degree between the future state of asset j and the kth sample offered by experts, h_{kij} is a possibility grade to reflect a similarity degree between the future state of the relation between asset i and asset j and the *kth* sample offered by experts. Since r_j; $j = 1, 2, \ldots, n$ are affected by uncertain factors, the expected returns and risks of assets cannot be predicted accurately. In Zhang and Nie [55], Zhang et al. [56], and Zhang and Wang [57], the admissible average return and covariance are respectively defined as

$$r_j^* = \bar{r}_j + \phi_j, \quad \phi_{jl} \leq \phi_j \leq \phi_{jh}, \quad j = 1, 2, \ldots, n, \tag{4}$$

and

$$\sigma_{ij}^* = \bar{\sigma}_{ij} + \varepsilon_{ij}, \quad \varepsilon_{ijl} \leq \varepsilon_{ij} \leq \varepsilon_{ijh}, \quad j = 1, 2, \ldots, n, \tag{5}$$

where ϕ_j is the admissible error for $\bar{r}_j$,ϕ_{jl} and ϕ_{jh} are the lower and upper bounds of ϕ_j; ε_{ij} is the admissible error for $\bar{\sigma}_{ij}$, ε_{ijl} and ε_{ijh} are the lower and upper bounds of ε_{ij}.
In order to describe conveniently, we use the following notations:

$$a = (\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n)^T, \quad \Sigma = (\bar{\sigma}_{ij})_{n\times n},$$
$$\Phi = (\phi_1, \phi_2, \ldots, \phi_n)^T, \quad V_\varepsilon = (\varepsilon_{ij})_{n\times n},$$
$$\Phi_l = (\phi_{1l}, \phi_{2l}, \ldots, \phi_{nl})^T, \quad \Phi_h = (\phi_{1h}, \phi_{2h}, \ldots, \phi_{nh})^T,$$
$$V_{\varepsilon l} = (\varepsilon_{ijl})_{n\times n}, \quad V_{\varepsilon h} = (\varepsilon_{ijh})_{n\times n}.$$

$a + \Phi$ is called as the admissible return vector. $\Sigma + V_\varepsilon$ is called as the admissible covariance matrix. Thus, the admissible return and covariance associated with portfolio x are defined respectively by

$$\widehat{r} = (a + \Phi)^T x, \tag{6}$$

and

$$\widehat{\sigma^2} = x^T(\Sigma + V_\varepsilon)x. \tag{7}$$

For any $x \geq 0$ the admissible return and covariance associated with portfolio x satisfy

$$(a + \Phi_l)^T x \leq \widehat{r} \leq (a + \Phi_h)^T x,$$

and

$$x^T(\Sigma + V_{\varepsilon l})x \leq \widehat{\sigma^2} \leq x^T(\Sigma + V_{\varepsilon h})x.$$

We now consider the admissible portfolio selection problem with transaction costs. Here we assume the transaction cost is a V-shaped function of the difference between a new portfolio $x = (x_1, x_2, \ldots, x_n)$, and the existing portfolio $x_0 = (x_1^0, x_2^0, \ldots, x_n^0)$. That's to say, the transaction cost for asset i is $c_i = k_i|x_i - x_i^0|$, the total transaction cost is $\sum_{i=1}^n c_i = \sum_{i=1}^n k_i|x_i - x_i^0|$ Thus, the admissible return after paying transaction costs is

$$(a + \Phi)^T x - \sum_{i=1}^n k_i|x_i - x_i^0|.$$

Moreover, it is known that very small weighting of an asset will have no distinct influence on the portfolio's return, but the administrative and monitoring costs will be increased. Similarly, very high weighting in any asset will cause investors to suffer from a larger risk. So bounded constraints should be considered when we construct portfolios in reality. Therefore, the admissible portfolio selection problem with transaction costs and bounded constraints can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & x^T(\Sigma + V_\varepsilon)x \\ \text{subject to} \quad & (a + \Phi)^T x - \sum_{i=1}^n k_i|x_i - x_i^0| \geq r_0, \\ & l \leq x \leq u, \\ & e^T x = 1, \end{aligned} \tag{8}$$

where $l = (l_1, l_2, \ldots, l_n)^T$, $u = (u_1, u_2, \ldots, u_n)^T$, and l_i and u_i are the minimum and maximum investment proportions in asset i, $i = 1, 2, \ldots, n$. If $(\Phi, V_\varepsilon) = (\Phi_h, V_{\varepsilon l})$ then (4) can be rewritten as follows:

$$\begin{aligned} \text{minimize} \quad & x^T(\Sigma + V_{\varepsilon l})x \\ \text{subjec to} \quad & (a + \Phi_h)^T x - \sum_{i=1}^n k_i|x_i - x_i^0| \geq r_0, \\ & l \leq x \leq u, \\ & e^T x = 1. \end{aligned} \tag{9}$$

If $(\Phi, V_\varepsilon) = (\Phi_l, V_{\varepsilon h})$ then (4) can be rewritten as follows:

$$\begin{aligned} &\text{minimize} \quad x^T(\Sigma + V_{\varepsilon h})x \\ &\text{subject to} \quad (a + \Phi_l)^T x - \sum_{i=1}^{n} k_i|x_i - x_i^0| \geq r_0, \\ &\qquad\quad l \leq x \leq u, \\ &\qquad\quad e^T x = 1. \end{aligned} \tag{10}$$

The optimal solution of the problems (9) and (10) is called the upper and lower admissible efficient portfolio respectively. The problem (9) means that the investor estimates the return and risk optimistically. The problem (10) means that the investor estimates the return and risk pessimistically. The problem (4) covers the scenario where the investor makes his portfolio selection neither too optimistically nor pessimistically.

3. A recurrent neural network algorithm for the admissible portfolio selection

It is generally admitted that the optimal portfolio selection problem with transaction costs is a complex mathematical programming problem. Because of the complexity of the problem, the traditional optimization algorithms cannot be applied. Thus, we use a neural network model to solve admissible portfolio selection model with transaction costs and bounded constraints.

It is clear that the upper and lower admissible efficient portfolio optimization problems (9) and (10) are quadratic programming problems. We thus solve these problems by a recurrent neural network model.

Let us consider the following general form of quadratic programming problem as

$$\text{minimize } \frac{1}{2}x^T Q x + d^T x, \tag{11}$$

$$\text{subject to } \begin{cases} Ax - b \leq 0, \\ e^T x - 1 = 0, \end{cases} \tag{12}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric and positive semidefinite matrix, $A \in \mathbb{R}^{(2n+1)\times n}$, $b \in \mathbb{R}^{2n+1}$, $e = (1, 1, ..., 1)^T$ and $x \in \mathbb{R}^n$.

We now give a neural network model for solving problem (11) and (12). To simplify the network architecture, we give the following lemma.

Lemma 3.1. *$x^*$is an optimal solution of* (11) *and* (12) *if and only if there exist $u^* \geq 0$ such that $(x^{*T}, u^{*T})^T$ satisfies:*

$$(I - \widetilde{P})[Qx^* + d + A^T u^*] + \widetilde{Q}(e^T x^* - 1) = 0, \tag{13}$$

$$(u^* + Ax^* - b)^+ - u^* = 0, \tag{14}$$

where $\widetilde{P} = e(e^T e)^{-1}e^T$, $\widetilde{Q} = e(e^T e)^{-1}$, and $(u)^+ = ([u_1]^+, \cdots, [u_{2n+1}]^+)^T$, $[u_i]^+ = \max\{0, u_i\}$.

Proof. If x^* is an optimal solution of (11) and (12), according to the KKT conditions for convex optimization, there exists $(x^{*T}, u^{*T}, v^{*T})^T, u^* \geq 0$ such that the following equations are satisfied.

$$Qx^* + d + ev^* + A^T u^* = 0, \tag{15}$$

$$e^T x^* - 1 = 0, \tag{16}$$

$$Ax^* \leq b, \quad u^* \geq 0, \quad (u^*)^T(Ax^* - b) = 0. \tag{17}$$

It is clear that (14) and (17) are equivalent. That is,

$$(Ax^* - b) \leq 0, \quad u^* \geq 0, \quad (u^*)^T(Ax^* - b) = 0 \Longleftrightarrow (u^* + Ax^* - b)^+ - u^* = 0.$$

Next, we only prove the solutions of (13) and the solutions of (15) and (16) are equivalent. From (15), one has

$$-e^T(Qx^* + d + ev^* + A^Tu^*) = 0.$$

Add the above equation with (16), we have

$$e^Tx^* - 1 - e^T(Qx^* + d) - e^Tev^* - e^TA^Tu^* = 0.$$

Thus,

$$ev^* = e(e^Te)^{-1}(e^Tx^* - 1) - e(e^Te)^{-1}e^T(Qx^* + d + A^Tu^*). \tag{18}$$

Substituting (18) into (15), one has

$$[I - e(e^Te)^{-1}e^T](Qx^* + d + A^Tu^*) + e(e^Te)^{-1}(e^Tx^* - 1) = 0.$$

Let $\widetilde{P} = e(e^Te)^{-1}e^T$ and $Q = e(e^Te)^{-1}$, then the above equation can be written as

$$(I - \widetilde{P})[Qx^* + d + A^Tu^*] + Q(e^Tx^* - 1) = 0.$$

Conversely, if there exists u^* such that $(x^{*T}, u^{*T})^T$ satisfies (13). Multiply e^T to both sides of (13), one has

$$e^T(I - \widetilde{P})[Qx^* + d + A^Tu^*] + e^TQ(e^Tx^* - 1) = 0.$$

Noting that $e^T(I - \widetilde{P}) = 0$ and $e^TQ = I$; we thus have

$$\begin{aligned} &e^Tx^* - 1 = 0,\\ &(I - \widetilde{P})(Qx^* + d + A^Tu^*) = 0. \end{aligned}$$

Let $v^* = -(e^Te)^{-1}e^T(Qx^* + d + A^Tu^*)$. One has

$$\begin{aligned} Qx^* + d + ev^* + A^Tu^* &= Qx^* + d + A^Tu^* - e(e^Te)^{-1}e^T(Qx^* + d + A^Tu^*)\\ &= (I - \widetilde{P})(Qx^* + d + A^Tu^*) = 0. \end{aligned}$$

Thus, there exists $(x^{*T}, u^{*T}, v^{*T})^T$ such that (15) and (16) in the KKT conditions are satisfied. This completes the proof. □

From the Lemma 3.1, we propose a capable neural network for solving (11) and (12) as follows:

$$\begin{cases} \dfrac{dx}{dt} = \eta\{-(I - \widetilde{P})[Qx + d + A^T(u + Ax - b)^+] - \widetilde{Q}(e^Tx - 1)\},\\ \dfrac{2du}{dt} = \eta\{-u + (u + Ax - b)^+\}, \end{cases} \tag{19}$$

where η is a scaling factor and indicates the convergence rate of the neural network (19). Throughout this paper, we let $\eta = 1$.

In the following, we give a more technical description of the proposed neural network algorithm whose framework is shown in Algorithm 1. The procedures start with choosing an arbitrary initial vector. Then iterative procedures are controlled by the principal while-loop of Algorithm 1. In each iteration, calculate the right side of system (19), update initial vector, calculation of error and stopping criteria, must be scheduled. The pseudo code that explains the implementation of the proposed algorithm is stated as follows:

Algorithm 1: The working of the proposed neural network algorithm.
Step 1: Initialization
Arbitrary choose initial vector $x \in \mathbb{R}^n$ **and** $u \in \mathbb{R}^{2n+1}$, $\Delta t > 0$, $t = 0$ **and error** $\epsilon = 10^{-9}$, **while** $t \leq T$ **(maximum iteration)**
Step 2: With an arbitrary value of η, **calculate**

$$\nu(t) = \eta\{-(I - \widetilde{P})[Qx + d + A^T(u + Ax - b)^+] - \widetilde{Q}(e^T x - 1)\},$$
$$\vartheta(t) = \frac{\eta}{2}\{-u + (u + Bx - d)^+\}.$$

Step 3: Update the initial vector

$$x(t + \Delta t) = x(t) + \Delta t \nu(t),$$
$$u(t + \Delta t) = u(t) + \Delta t \vartheta(t).$$

Step 4: Calculation

$$r(t) = \sum_{i=1}^{n} \nu_i^2(t),\ s(t) = \sum_{j=1}^{2n+1} \vartheta_j(t)^2,$$

Step 5: Stopping criteria: if r **and** $s < \epsilon$ **; else** $t = t + 1$.
end.
Output:

$$x(t + \Delta t),\ u(t + \Delta t).$$

Remark 3.1. *By Lemma 3.1, it is easy to know that* x^* *is an optimal solution of* (11) *and* (12) *if and only if there exists* $u^* \geq 0$ *such that* $(x^{*T}, u^{*T})^T$ *is an equilibrium point of* (19). *So, when the neural network converges to its equilibrium point, the state trajectory* $x(t)$ *converges to the optimal solution of problem* (11) *and* (12).

In this section, we will prove the global convergence of the neural network (19). Let Ω^e denote the equilibrium point set of neural network (19).

Lemma 3.2. *Let* $(x^{*T}, u^{*T})^T \in \Omega^e$ *be an equilibrium point of* (19) *and*

$$\begin{aligned} V(x,u) = {} & \frac{1}{2}x^T Q x + d^T x - (\frac{1}{2}x^{*T} Q x^* + d^T x^*) + \frac{1}{2}\|(u + Ax - b)^+\|^2 - \frac{1}{2}\|(u^* + Ax^* - b)^+\|^2 \\ & - (x - x^*)^T(Qx^* + d + A^T u^*) - (u - u^*)^T u^* + \frac{1}{2}\|x - x^*\|^2 + \frac{1}{2}\|u - u^*\|^2. \end{aligned} \tag{20}$$

Then

$$(I)\ V(x,u) \geq \frac{1}{2}\|x - x^*\|^2 + \frac{1}{2}\|u - u^*\|^2,$$
$$(II)\ \frac{\mathrm{d}V}{\mathrm{d}t} \leq -\|(I - \widetilde{P})[Qx + d + A^T(u + Ax - b)^+] + \widetilde{Q}(e^T x - 1)\|^2 - \frac{1}{2}\|u - (u + Ax - b)^+\|^2.$$

Proof. (I) It is easy to verify that $\frac{1}{2}x^TQx + d^Tx + \frac{1}{2}\|(u + Ax - b)^+\|^2$ is a convex differentiable function [37] and

$$\nabla(\frac{1}{2}x^TQx + d^Tx + \frac{1}{2}\|(u + Ax - b)^+\|^2) = \begin{bmatrix} Qx + d + A^T(u + Ax - b)^+ \\ (u + Ax - b)^+ \end{bmatrix}.$$

Thus,

$$\begin{aligned}\frac{1}{2}x^TQx + d^Tx - (\frac{1}{2}x^{*T}Qx^* + d^Tx^*) + \frac{1}{2}\|(u + Ax - b)^+\|^2 - \frac{1}{2}\|(u^* + Ax^* - d)^+\|^2 \\ \geq (x - x^*)^T(Qx + d + A^Tu^*)(u - u^*)^Tu^*.\end{aligned}$$

i.e.

$$V(x, u) \geq \frac{1}{2}\|x - x^*\|^2 + \frac{1}{2}\|u - u^*\|^2.$$

(II) Calculating the derivative of V with respect to t,

$$\begin{aligned}\frac{dV}{dt} =& \frac{\partial V}{\partial x}\frac{dx}{dt} + \frac{\partial V}{\partial u}\frac{du}{dt} \\ =& -[Qx + d + A^T(u + Ax - b)^+ - (Qx^* + d) - A^Tu^* + x - x^*]^T \\ & \{(I - \widetilde{P})[Qx + d + A^T(u + Ax - b)^+] + \widetilde{Q}(e^Tx - 1)\} \\ & - \frac{1}{2}[u - (u + Ax - b)^+]^T[(u + Ax - b)^+ - 2u^* + u] \\ =& -[Q(x - x^*) + x - x^* + A^T(u + Ax - b)^+ - A^Tu^*]^T \\ & \{(I - \widetilde{P})[Q(x - x^*) + A^T(u + Ax - b)^+ - A^Tu^*)] + \widetilde{P}(x - x^*)\} \\ & - \frac{1}{2}[u - (u + Ax - b)^+]^T[u - (u + Ax - b)^+ + 2(u + Ax - b)^+ - 2u^*] \\ =& -\{[Q(x - x^*)]^T(I - \widetilde{P})[Q(x - x^*)] \\ & + [Q(x - x^*)]^T(I - \widetilde{P})[A^T(u + Ax - b)^+ - A^Tu^*] \\ & + [Q(x - x^*)]^T\widetilde{P}(x - x^*) \\ & + (x - x^*)^T(I - \widetilde{P})[A^T(u + Ax - b)^+ - A^Tu^*] \\ & + (x - x^*)^T\widetilde{P}(x - x^*) \\ & + [A^T(u + Ax - b)^+ - A^Tu^*]^T(I - \widetilde{P})[Q(x - x^*)] \\ & + [A^T(u + Ax - b)^+ - A^Tu^*]^T(I - \widetilde{P})[A^T(u + Ax - b)^+ - A^Tu^*] \\ & + [A^T(u + Ax - b)^+ - A^Tu^*]^T\widetilde{P}(x - x^*)\} \\ & - \frac{1}{2}\|u - (u + Ax - b)^+\|^2 + [(u + Ax - b)^+ - u]^T[(u + Ax - b)^+ - u^*].\end{aligned}$$

Noting that

$$(I - \widetilde{P})^2 = I - \widetilde{P},\ \widetilde{P}^2 = \widetilde{P},\ \widetilde{P}(I - \widetilde{P}) = 0,\ (u + Ax - b)^+ - u = Ax - b + (-u + b - Ax)^+$$

we have

$$\begin{aligned}\frac{dV}{dt} = \frac{dV}{dt} =& -[Q(x - x^*)]^T(I - \widetilde{P})^2[Q(x - x^*)] \\ & - 2[Q(x - x^*)]^T(I - \widetilde{P})^2[A^T(u + Ax - b)^+ - A^Tu^*] \\ & - [A^T(u + Ax - b)^+ - A^Tu^*]^T(I - \widetilde{P})^2[A^T(u + Ax - b)^+ - A^Tu^*]\end{aligned}$$

$$
\begin{aligned}
&- (x-x^*)^T\widetilde{P}^2(x-x^*) - (x-x^*)^T[Q(x-x^*)] \\
&- (x-x^*)^T[A^T(u+Ax-b)^+ - A^Tu^*] \\
&- \frac{1}{2}\|u-(u+Ax-b)^+\|^2 + [Ax-b+(-u+b-Ax)^+]^T[(u+Ax-b)^+ - u^*] \\
&= -\|(I-\widetilde{P})[Q(x-x^*)] + (I-\widetilde{P})[A^T(u+Ax-b)^+ - A^Tu^*] + \widetilde{P}(x-x^*)\|^2 \\
&- (x-x^*)^T[Q(x-x^*)] - (x-x^*)^T[A^T(u+Ax-b)^+ - A^Tu^*] \\
&- \frac{1}{2}\|u-(u+Ax-b)^+\|^2 + (Ax-b)^T(u+Ax-b)^+ - (Ax-b)^Tu^* \\
&+ [(-u+b-Ax)^+]^T(u+Ax-b)^+ - [(-u+b-Ax)^+]^Tu^* \\
&= -\|(I-\widetilde{P})[Qx+d+A^T(u+Ax-b)^+] + \widetilde{Q}(e^Tx-1)\|^2 \\
&- (x-x^*)^T[Q(x-x^*)] - (x-x^*)^T[A^T(u+Ax-b)^+ - A^Tu^*] \\
&- \frac{1}{2}\|u-(u+Ax-b)^+\|^2 + (Ax-b)^T[(u+Ax-b)^+] - (Ax-b)^Tu^* \\
&+ [(-u+b-Ax)^+]^T(u+Ax-b)^+ - [(-u+b-Ax)^+]^Tu^* \\
&= -\|(I-\widetilde{P})[Qx+d+A^T(u+Ax-b)^+] + \widetilde{Q}(e^Tx-1)\|^2 \\
&- \frac{1}{2}\|u-(u+Ax-b)^+\|^2 - (x-x^*)^T[Q(x-x^*)] - [(-u+b-Ax)^+]^Tu^* \\
&+ [A(x^*-x)+A(x-x^*)]^Tu^* - (Ax^*-b)^Tu^* \\
&+ [A(x-x^*)+A(x^*-x)]^T(u+Ax-b)^+ + (Ax^*-b)^T(u+Ax-b)^+.
\end{aligned}
$$

It is easy to verify

$$
\begin{aligned}
&[(-u+b-Ax)^+]^T(u+Ax-b)^+ = 0, \\
&(Ax^*-b)^Tu^* = 0, \\
&[(-u+b-Ax)^+]^Tu^* \geq 0, \\
&(Ax^*-b)^T[(u+Ax-b)^+] \leq 0.
\end{aligned}
$$

Since Q is a positive semi-definite matrix, we have

$$(x-x^*)^T[Q(x-x^*)] \geq 0.$$

Thus

$$\frac{dV}{dt} \leq -\|(I-\widetilde{P})[Qx+d+A^T(u+Ax-b)^+] + \widetilde{Q}(e^Tx-1)\|^2 - \frac{1}{2}\|u-(u+Ax-b)^+\|^2 \leq 0. \tag{21}$$

This completes the proof. □

Remark 3.2. *It is easy to see that $V(x,u)$ is a Lyapunov function. From the proof of Lemma 3.2, we have $\frac{dV}{dt} \leq 0$. Thus, the neural network* (19) *is stable in the sense of Lyapunov.*

Theorem 3.1. *For any initial point $(x(0)^T, u(0)^T)^T \in \mathbb{R}^{3n+1}$, there exists a unique continuous solution $(x(t)^T, u(t)^T)^T \in \mathbb{R}^{3n+1}$ of* (19) *for $t \geq 0$.*

Proof. The projection operation $(.)^+$ is nonexpansive. It is clear that $(I-\widetilde{P})[Qx-d+A^T(u+Ax-b)^+]+\widetilde{Q}(e^Tx-1)$ and $(u+Ax-b)^+-u$ are locally Lipschitz continuous. From the local existence theorem of ordinary differential equation [63], there exists one unique continuous solution of (19) on $[0,T)$.

Let $[0, T)$ be its maximal interval of existence. By Lemma 3.2, we know that V is a non-increasing function with respect to t, so

$$\frac{1}{2}\|x(t) - x^*\|^2 + \frac{1}{2}\|u(t) - u^*\|^2 \leq V(x(t), u(t)) \leq V(x(0), u(0)), \quad \forall t \geq 0, \tag{22}$$

This shows the state trajectory of neural network (19) is bounded. Thus $T = +\infty$. This completes the proof. □

Theorem 3.2. *The state trajectory of neural network* (19) *converges to an equilibrium point for any initial point* $(x(0)^T, u(0)^T)^T \in \mathbb{R}^{3n+1}$. *In particular, the neural network* (19) *with any initial point* $(x(0)^T, u(0)^T)^T \in \mathbb{R}^{3n+1}$ *is globally asymptotically stable when* Ω^e *has unique equilibrium point.*

Proof. Define

$$D(x, u) = \|(I - \widetilde{P})[Qx + d + A^T(u + Ax - b)^+] + \widetilde{Q}(e^T x - 1)\|^2 + \frac{1}{2}\|u - (u + Ax - b)^+\|^2.$$

Then $D(x, u) = 0$, if and only if $(x^T, u^T)^T$ is an equilibrium point of neural network (19). From the proof of Theorem 3.1, we have the state trajectory $(x(t)^T, u(t)^T)^T$ of neural network (19) is bounded. Therefore, there exists an increasing sequence $\{t_n\}$ with $\lim_{n\to\infty} t_n \longrightarrow \infty$ and a limit point $(\hat{x}^T, \hat{u}^T)^T$ such that $\lim_{n\to\infty} x(t_n) \longrightarrow \hat{x}$ and $\lim_{n\to\infty} u(t_n) \longrightarrow \hat{u}$. Next, we will prove $(\hat{x}^T, \hat{u}^T)^T \in \Omega^e$ and the state trajectory $(x(t)^T, u(t)^T)^T$ globally converge to the equilibrium point $(\hat{x}^T, \hat{u}^T)^T$.

Firstly, we prove that $D(\hat{x}, \hat{u}) = 0$, i.e. $(\hat{x}^T, \hat{u}^T)^T$ is an equilibrium point of the neural network (19). If it does not hold, that is $D(\hat{x}, \hat{u}) > 0$. Since $D(x, u)$ is continuous with respect to x and u, respectively, there exist $\varepsilon > 0$, $q > 0$ and a ε neighborhood of $(\hat{x}^T, \hat{u}^T)^T$,

$$B((\hat{x}^T, \hat{u}^T)^T, \varepsilon) = \{(x^T, u^T)^T \ : \ \|x - \hat{x}\| + \|u - \hat{u}\| \leq \varepsilon\},$$

such that $D(x, u) > q$, for all $(x^T, u^T)^T \in B((\hat{x}, \hat{u}), \varepsilon)$. Noting that $\lim_{n\to\infty} x(t_n) \longrightarrow \hat{x}$ and $\lim_{n\to\infty} u(t_n) \longrightarrow \hat{u}$, there exists a positive integer N, such that for all $n \geq N$, $\|x(t_n) - \hat{x}\| \leq \frac{1}{4}\varepsilon$, $\|u(t_n) - \hat{u}\| \leq \frac{1}{4}\varepsilon$.

From (19) and the boundedness of $(x(t)^T, u(t)^T)^T$ we have $\dot{x}$ and $\dot{u}$ are also bounded, denoted by M. Considering $t \in [t_n - \frac{\varepsilon}{8M}, t_n + \frac{\varepsilon}{8M}]$, $n \geq N$ we have

$$\begin{aligned}
\|x(t) - \hat{x}\| + \|u(t) - \hat{u}\| \leq & \|x(t) - x(t_n)\| + \|u(t) - u(t_n)\| + \|x(t_n) - \hat{x}\| + \|u(t_n) - \hat{u}\| \\
= & \|\dot{x}(\xi_1)\| \times |t - t_n| + \|\dot{u}(\xi_2)\| \times |t - t_n| \\
& + \|x(t_n) - \hat{x}\| + \|u(t_n) - \hat{u}\| \\
\leq & 2M|t - t_n| + \frac{\varepsilon}{2} \leq \varepsilon.
\end{aligned}$$

That is, for all $t \in [t_n - \frac{\varepsilon}{8M}, t_n + \frac{\varepsilon}{8M}]$, $n \geq N$, $(x^T, u^T)^T \in B((\hat{x}, \hat{u}), \varepsilon)$. Therefore, $D(x(t), u(t)) > q$, for all $t \in [t_n - \frac{\varepsilon}{8M}, t_n + \frac{\varepsilon}{8M}]$, $n \geq N$. Since $\lim_{n\to\infty} t_n = \infty$, and the Lebesgue measure of set $t \in \cup_{n\geq N}[t_n - \frac{\varepsilon}{8M}, t_n + \frac{\varepsilon}{8M}]$ is infinite, then we have

$$\int_0^\infty D(x(t), u(t))\, dt = \infty. \tag{23}$$

However,

$$\int_0^\infty D(x(t), u(t))\, dt = \lim_{s\to\infty} \int_0^s D(x(t), u(t))\, dt$$

$$\begin{aligned}
&\leq -\lim_{s\to\infty}\int_0^s \dot{V}(x(t),u(t))\,dt\\
&=\lim_{s\to\infty}[V(x(0),u(0))-V(x(s),u(s))]\\
&\leq V(x(0),u(0)),
\end{aligned}$$

which contradicts (23). So, $D(\hat{x},\hat{u})=0$, which means $(\hat{x}^T,\hat{u}^T)^T\in\Omega^e$. Secondly, we prove the state trajectory $(x(t)^T,u(t)^T)^T$ globally converge to the equilibrium point $(\hat{x},\hat{u})$. Define a Lyapunov function

$$\begin{aligned}
\hat{V}(x,u)=&\frac{1}{2}x^TQx+d^Tx-(\frac{1}{2}\hat{x}^TQ\hat{x}+d^T\hat{x})+\frac{1}{2}\|(u+Ax-b)^+\|^2-\frac{1}{2}\|(\hat{u}+A\hat{x}-b)^+\|^2\\
&-(x-\hat{x})^T(Q\hat{x}+d+A^T\hat{u})-(u-\hat{u})^T\hat{u}+\frac{1}{2}\|x-\hat{x}\|^2+\frac{1}{2}\|u-\hat{u}\|^2.
\end{aligned}$$

one has $\hat{V}(\hat{x},\hat{u})=0$. Since $\lim_{n\to\infty}x(t_n)\longrightarrow\hat{x}$ and $\lim_{n\to\infty}u(t_n)\longrightarrow\hat{u}$, $\forall\varepsilon>0$, there exists a t_k, such that $\hat{V}(x(t_k),u(t_k))<\varepsilon$. By the Lemma 3.2, we have

$$\frac{1}{2}\|x(t)-\hat{x}\|^2+\frac{1}{2}\|u(t)-\hat{u}\|^2\leq\hat{V}(x(t),u(t)),$$

and $\hat{V}$ is nonincreasing. Therefore, for all $t\geq t_k$, we have

$$\frac{1}{2}\|x(t)-\hat{x}\|^2+\frac{1}{2}\|u(t)-\hat{u}\|^2\leq\hat{V}(x(t),u(t))\leq\hat{V}(x(t_k),u(t_k))\leq\varepsilon.$$

That means, $\lim_{n\to\infty}x(t)=\hat{x}$, and $\lim_{n\to\infty}u(t)=\hat{u}$. So the state trajectory of the neural network (19) globally converges to the equilibrium point $(\hat{x}^T,\hat{u}^T)^T$. In particular, if $\Omega^e=\{(x^{*T},u^{*T})^T\}$, then the state trajectory $(x(t)^T,u(t)^T)^T$ with any initial point $(x(0)^T,u(0)^T)^T$ will approach to $(x^{*T},u^{*T})^T$ by the analysis above. Then the neural network is globally asymptotically stable. □

4. Comparison with some neural networks

In order to see how well the presented neural network (19) can be applied to solve (11) and (12), we compare it with some existing neural network models.

There is a kind of neural network model called the gradient model. In order to use the gradient neural network model, a constrained optimization problem can be approximated by an unconstrained optimization problem. Then the energy function is constructed by the penalty function method. It is noticeable that the gradient neural network model has an advantage as the model may be defined directly using the derivatives of the energy function. But its shortcoming is that the convergence is not guaranteed, especially in the case of unbounded solution sets [61]. Moreover, the gradient neural network based on the penalty function requires any adjustable parameter called the penalty parameter. For instance, utilizing the penalty method, the constrained optimization problem (11) and (12) can be approximated by the following unconstrained optimization problem

$$\text{minimize}\ \ E_1(x)=\frac{1}{2}x^TQx+d^Tx+\frac{\gamma}{2}\left\{\sum_{k=1}^{2n+1}[(Ax-b)_k^+]^2+(e^Tx-1)^2\right\},$$

where γ is a penalty parameter. The gradient neural network model is then given by

$$\frac{dx}{dt}\quad=\quad-\nabla E_1(x)=-\Big(Qx+d+\gamma[A^T(Ax-b)^+e^T(ex-1)]\Big). \tag{24}$$

The system in (24) is referred to as Kennedy and Chua's neural network model [38]. This network is not capable to find an exact optimal solution due to a finite penalty parameter and is difficult to implement when

the penalty parameter is very large. Thus, this network only converges to an approximate solution of (11) and (12) for any given finite penalty parameter. It can be also shown that Kennedy–Chua's neural network (24) is not globally convergent to an exact optimal solution of some convex programming problems. For instance, one can see the first Example in [54].

In [62], Nazemi and Nazemi presented a gradient model for dealing with the problem (11) and (12) as

$$\frac{d(y(t))}{dt} = -\nabla E_2(y(t)), \tag{25}$$

$$y(0) = y_0, \;\; y(t) = (x(t), u(t), v(t))^T, \tag{26}$$

where

$$E_2(y) = \frac{1}{2}\|\rho(y)\|^2, \tag{27}$$

$$\rho(y) = \begin{bmatrix} Qx + d + e^T v + A^T u \\ -e^T x + 1 \\ \phi^{\varepsilon}_{\mathrm{FB}}(u, b - Ax) \end{bmatrix} = 0, \tag{28}$$

and

$$\phi^{\varepsilon}_{\mathrm{FB}}(a, b) = (a + b) - \sqrt{a^2 + b^2 + \varepsilon}, \;\; \varepsilon \to 0_+.$$

In [42], Nazemi proposed a Lagrangian neural network model to solve (11) and (12) as

$$\frac{dy}{dt} = \Phi(y), \tag{29}$$

$$y(t_0) = y_0, \;\; y(t) = (x(t), u(t), v(t))^T, \;\; u(t_0) > 0, \tag{30}$$

where

$$\Phi(y) = \begin{bmatrix} -\left(Qx + d + \frac{1}{2}A^T u^2 + e^T v\right) \\ \mathrm{diag}(u_1, ..., u_{2n+1})(Ax - b) \\ e^T x - 1 \end{bmatrix}.$$

Effati and Ranjbar in [60] proposed a dual neural network which can be modeled for (11) and (12) without equality constraint $e^T x - 1 = 0$ as

$$\theta = -Q^{-1}(\tilde{d} + A^T u), \tag{31}$$

$$\frac{du}{dt} = \tilde{D}u + \tilde{d}, \; u \geq 0, \tag{32}$$

$$u(t_0) = u_0, \tag{33}$$

where u is the dual Lagrangian variable and $\tilde{D} = -AQ^{-1}A^T$ and $\tilde{d} = -b - AQ^{-1}d$.

In the models (25)-(26) and (29)-(30), the state variables are $x \in \mathbb{R}^n, u \in \mathbb{R}^{2n+1}$ and $v \in \mathbb{R}$. So this neural network has $3n + 2$ neurons. However, our neural network has only $3n + 1$ neurons. More importantly, the convergence of the models (25)-(26), (29)-(30) and (31)-(33) is guaranteed only under the condition that Q is positive definite matrix for all $x \in \mathbb{R}^n$ and $u_k \geq 0$. Thus, these models can not solve linear programming problems which limits their application. Moreover, our neural network can converge to the optimal solution under the condition of positive semidefinite matrix Q for all $x \in \mathbb{R}^n$ and $u_k \geq 0$.

In [54], Nazemi also proposed the following neural network for solving the problem (11) and (12):

$$\frac{dx}{dt} = -\left(Qx + d + A^T(u + Ax + b)^+ + e^T v\right), \tag{34}$$

$$\frac{du}{dt} = (u + Ax - b)^+ - u, \tag{35}$$

$$\frac{dv}{dt} = e^T x - 1. \tag{36}$$

Leung et al. in [58], constructed the following neural network for solving the problem (11) and (12) as

$$\frac{dx}{dt} = -u^T(Ax - b) \cdot A^T u + A^T[b - Ax - |b - Ax|] \tag{37}$$
$$-\nabla^2_{xx} L(y)\nabla_x L(y) - e(e^T x - 1), \tag{38}$$
$$\frac{du}{dt} = -u^T(Ax - b) \cdot A + A\nabla_x L(u) - (u - |u|), \tag{39}$$
$$\frac{dv}{dt} = e^T \nabla_x L(y), \tag{40}$$
$$y(t_0) = y_0, \quad y(t) = (x(t), u(t), v(t))^T, \tag{41}$$

where $\nabla^2_{xx} L(y)$ is the Hessian matrix of the function $L(y)$ with respect to x, and

$$L(y) = \frac{1}{2}x^T Qx + d^T x + u^T(Ax - b) - v^T(e^T x - 1),$$

$$\nabla_x L(y) = Qx + d + A^T u - ev.$$

In the models (34)-(36) and (37)-(41), the state variables are $x \in \mathbb{R}^n$, $u \in \mathbb{R}^{2n+1}$ and $v \in \mathbb{R}$. So these neural networks have $3n + 2$ neurons. However, our neural network has only $3n + 1$ neurons.

5. Numerical example

Example 5.1. *[59] To illustrate our proposed effective means and approaches in this paper, we consider a practical example introduced by Markowitz [36]. In this example,*

$x^0 = (0.1, 0.4, 0, 0, 0, 0, 0.3, 0.2, 0)$,
$l = (0, 0, 0, 0, 0, 0, 0, 0, 0)$,
$u = (0.2, 0.2, 0.3, 0.3, 0.35, 0.40, 0.2, 0.25, 0.3)$,
the possibility grade h_k is defined by

$$h_{kij} = h_{ki} = h_k = 0.1 + 0.3\frac{k-1}{17}, \quad k = 1, 2, \ldots, 18.$$

$V_\epsilon = 0$, Φ_h and Φ_l are given by

$$\Phi_h = (0.014, 0.014, 0.039, 0.041, 0.047, 0.01, 0.027, 0.043, 0.028)^T, \quad \Phi_l = -\Phi_h.$$

Using (2) we obtained $a, a + \Phi_l, a + \Phi_h$ as follows:

$$a = (0.07099, 0.07012, 0.19645, 0.20610, 0.23390, 0.05317, 0.13619, 0.21603, 0.14386)^T,$$

$$a + \Phi_h = (0.08499, 0.08412, 0.23545, 0.24710, 0.28090, 0.06317, 0.16319, 0.25903, 0.17186)^T,$$

$$a + \Phi_l = (0.05699, 0.05612, 0.15745, 0.16510, 0.18690, 0.04317, 0.10919, 0.17303, 0.11586)^T.$$

Using (3) we obtained the covariance matrix $\bar{V}$ as follows:

$$\bar{V} = \begin{pmatrix}
0.04042 & 0.01593 & 0.01933 & 0.03277 & 0.01046 & 0.02665 & 0.0183 & 0.02816 & 0.02077 \\
0.01593 & 0.01119 & 0.01701 & 0.01875 & 0.00771 & 0.00705 & 0.00912 & 0.02106 & 0.01518 \\
0.01933 & 0.01701 & 0.09852 & 0.06287 & 0.04545 & 0.00694 & 0.0035 & 0.07752 & 0.03114 \\
0.03277 & 0.01875 & 0.06287 & 0.07954 & 0.04465 & 0.02072 & 0.00904 & 0.07894 & 0.01964 \\
0.01046 & 0.00771 & 0.04545 & 0.04465 & 0.09713 & 0.00585 & 0.01096 & 0.086 & 0.0285 \\
0.02665 & 0.00705 & 0.00694 & 0.02072 & 0.00585 & 0.0405 & 0.00953 & 0.01952 & 0.01095 \\
0.0183 & 0.00912 & 0.0035 & 0.00904 & 0.01096 & 0.00953 & 0.01986 & 0.01784 & 0.0079 \\
0.02816 & 0.02106 & 0.07752 & 0.07894 & 0.086 & 0.01952 & 0.01784 & 0.13343 & 0.03455 \\
0.02077 & 0.01518 & 0.03114 & 0.01964 & 0.0285 & 0.01095 & 0.0079 & 0.03455 & 0.06211
\end{pmatrix}.$$

We solve the upper and lower admissible efficient portfolio optimization problems (9) and (10) with the proposed model (19). All simulation results show that the state trajectories of the proposed model converge to the unique optimal solution x^*. Figures (1) and (2) display the values of $x(t)$ of the neural network model in (19). We test the influence of the parameter η in neural network (19) on the value of $\|x(t) - x^*\|^2$ of the problems (9) and (10). From Figures (3) and (4) we see that when $\eta = 1$, the neural network (19) generates the slowest decrease of $\|x(t) - x^*\|^2$, whereas when $\eta = 20$, it generates the fastest decrease of $\|x(t) - x^*\|^2$. We see that a larger η yields a better convergence rate of the error $\|x(t) - x^*\|^2$. Thus, we see that a larger η yields a better convergence rate of the error $\|x(t) - x^*\|^2$. An l_2 norm error between x and x^* with 10 random initial points of the problems (9) and (10) are also shown in Figures (5) and (6) which shows the proposed model is globally convergent to x^*. In fact, Theorem 3.2 guarantees that the stated model in (19) converges globally to x^*.

Return	*Am.T.*	*A.T.&T.*	*U.S.S.*	*G.M.*	*A.T.&S.*	*C.C.*	*Bdn.*	*Frstn.*	*S.S.*	*Risk*
0.05	0.0563	0.2000	0.0383	0	0.1004	0.2900	0.2000	0	0.1150	0.0161
0.10	0.0698	0.2000	0.0545	0	0.0926	0.2881	0.2000	0	0.0948	0.0161
0.15	0.0368	0.2000	0.1116	0.0168	0.1534	0.1222	0.2000	0	0.1590	0.0190

Table 1: The upper admissible efficient portfolios.

Return	*Am.T.*	*A.T.&T.*	*U.S.S.*	*G.M.*	*A.T.&S.*	*C.C.*	*Bdn.*	*Frstn.*	*S.S.*	*Risk*
0.05	0.0411	0.2000	0.0525	0	0.0622	0.3016	0.2000	0	0.1425	0.0161
0.10	0	0.2000	0.0675	0.0835	0.1772	0.1468	0.2000	0.0065	0.1184	0.0197
0.15	0	0	0.2243	0.0723	0.3500	0	0.1723	0.1811	0	0.0559

Table 2: The lower admissible efficient portfolios.

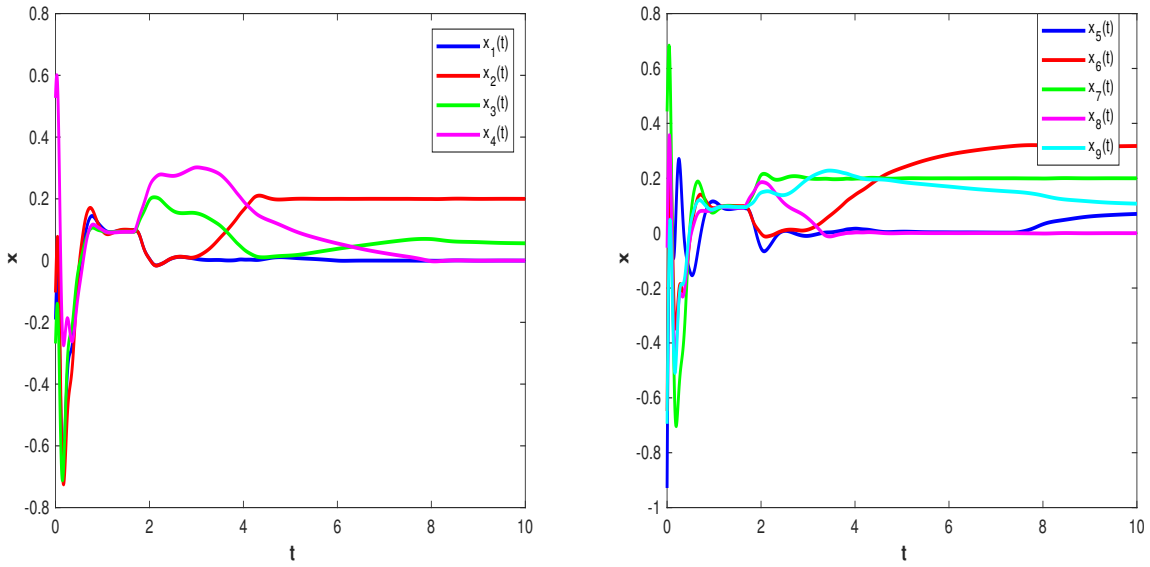

Figure 1: Transient behaviors of $x_1(t), ..., x_9(t)$ with $r_0 = 0.05$ for the problem (9).

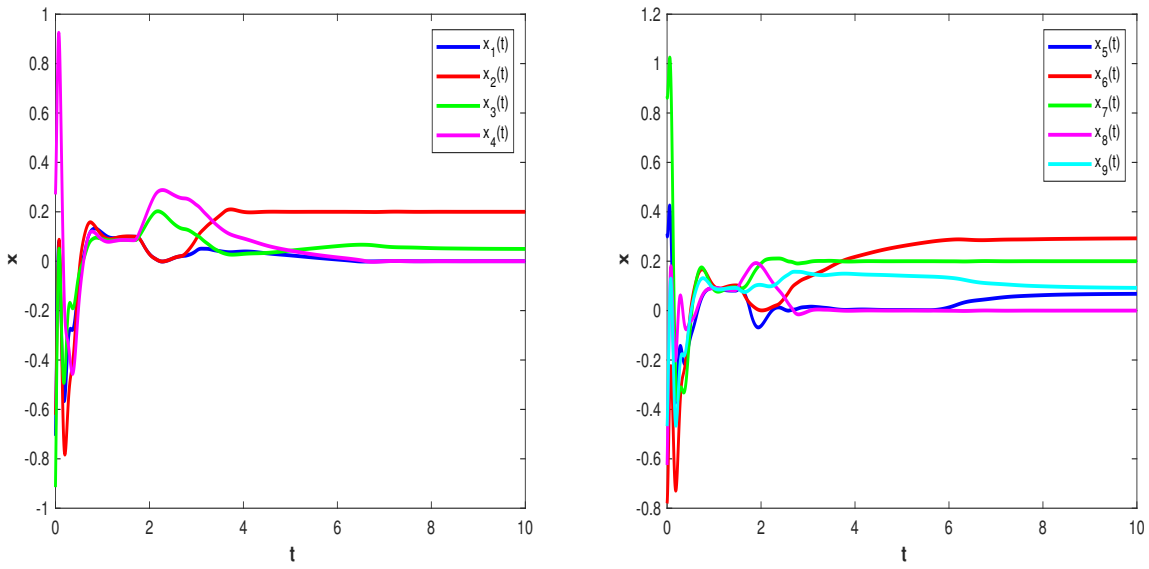

Figure 2: Transient behaviors of $x_1(t), ..., x_9(t)$ with $r_0 = 0.05$ for the problem (10).

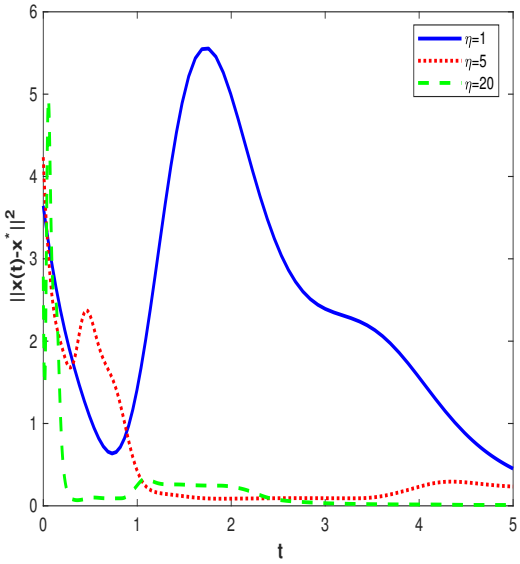

Figure 3: Convergence behavior of $\|x(t) - x^*\|^2$ with different values of η and $r_0 = 0.05$ for the problem (9).

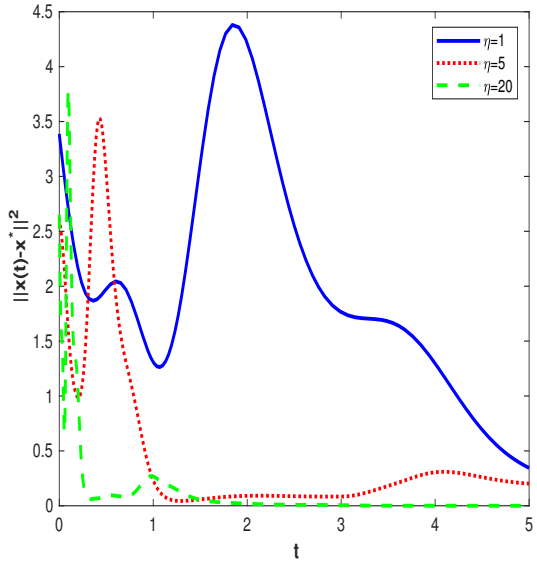

Figure 4: Convergence behavior of $\|x(t) - x^*\|^2$ with different values of η and $r_0 = 0.05$ for the problem (10).

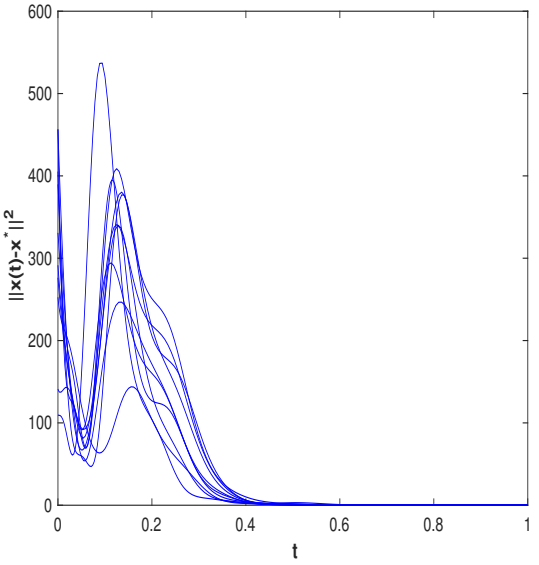

Figure 5: The convergence behavior of $\|x(t) - x^*\|^2$ with $\eta = 15$, 10 different initial points and $r_0 = 0.05$ for the problem (9).

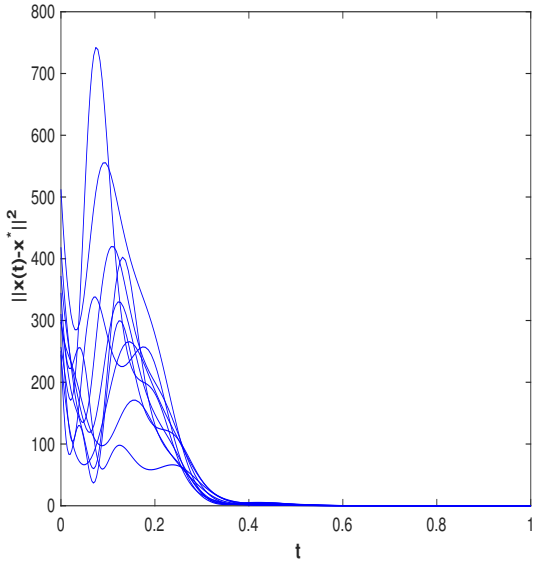

Figure 6: The convergence behavior of $\|x(t) - x^*\|^2$ with $\eta = 15$, 10 different initial points and $r_0 = 0.05$ for the problem (10).

6. Conclusion

In this paper, we discussed the admissible portfolio selection problem which includes transaction costs and bounded constraints. Although there are some traditional algorithms for our proposed problem, they cannot behave well. Therefore, based on a some concepts of convex programming and differential equations, we designed a capable neural network to solve the proposed portfolio selection problem. In contrast to some existing neural networks, the proposed neural network does not require any adjustable parameter and its structure is simple. Thus, the suggested model is more suitable to be implemented in hardware. In this manuscript we also analyze the influence of the parameter η in dynamic model (19) on the convergence rate of the trajectory and the convergence behavior of $\|x(t) - x^*\|^2$ and obtain that a larger η leads to a better convergence rate. The simulation results clearly demonstrate the convergence behavior and the characteristics of the proposed network. We depicted a numerical example of the portfolio selection problem to illustrate our proposed effective means and approaches. Experiment results show that the neural network algorithm is effective for solving complex constrained portfolio selection problem and transaction costs and bounded constraints have a great impact on the portfolio selection decision. Some future topics are to extend the proposed neural network framework for various portfolio optimization problems with different risk measures in probabilistic, credibilistic and uncertain spaces (see [64]) which are modeled as convex quadratic optimization problems.

References

[1] H. Markowitz, *Portfolio selection*, J. Finance **7** (1952), 77–91.
[2] P. Artzner, F. Eber, J. M. Eber, D. Heath, *Thinking coherently*, Risk **10** (1997), 68–71.
[3] P. Artzner, F. Delbaen, J. M. Eber, D. Heath, *Coherent measures of risk*, Math. Finance **9** (1999), 203–228.
[4] A. K. Bera, S. Y. Park, *Optimal portfolio diversification using the maximum entropy principle*, Econom. Rev. **27** (2008), 484–512.
[5] S. Basak ,A. Shapiro , *Value-at-risk based risk management: optimal policies and asset prices*, Rev. Financ. Stud. **14** (2001), 371–405.
[6] R. Campbell, R. Huisman, K. Koedijk, *Optimal portfolio selection in a value-at-risk framework*, J. Bank. Finance **25** (2001), 1789–1804.
[7] P. Jorion, *Value at Risk: The New Benchmark for Managing Financial Risk*, (3rd edition), McGraw-Hill, New York, 2007.
[8] R. T. Rockafellar, S. Uryasev, *Optimization of conditional value-at-risk*, J. Risk **2** (2000), 21–41.

[9] R. T. Rockafellar ,S. Uryasev, *Conditional value-at-risk for general loss distributions*, J. Bank. Finance **26** (2002), 1443–1471.
[10] X. Cai, K. L. Teo, X. Yang, X. Zhou, *Portfolio optimization under a minimax rule*, Manag. Sci. **46** (2000), 957–972.
[11] G. G. Polak, D. F. Rogers, D. J. Sweeney, *Risk management strategies via minimax portfolio optimization*, Eur. J. Oper. Res. **207** (2010), 409–419.
[12] Y. Sun , G. Aw , K. L. Teo , G. Zhou, *Portfolio optimization using a new probabilistic risk measure*, J. Ind. Manag. Optim. **11** (2015), 1275–1283.
[13] B. Li, Y. Sun, G. Aw, K. L. Teo, *Uncertain portfolio optimization problem under a minimax risk measure*, Appl. Math. Model. **76** (2019), 274–281.
[14] R. D. Arnott, W. H. Wanger, *The measurement and control of trading costs*, Financ. Anal. J. **46** (1990), 73–80.
[15] A. Yoshimoto, *The mean-variance approach to portfolio optimization subject to transaction costs*, J. Oper. Res. Soc. **39** (1996), 99–117.
[16] H. Babazadeh, A. Esfahanipour, *A novel multi period mean-VaR portfolio optimization model considering practical constraints and transaction cost*, J. Comput. Appl. Math. **361** (2019), 313–342.
[17] S. S.Meghwani, M. Thakur, *Multi-objective heuristic algorithms for practical portfolio optimization and rebalancing with transaction cost*, Appl. Soft Comput. **67** (2018), 865-894.
[18] T. E. Simos, S. D. Mourtas, V. N.Katsikis, *Time-varying Black-Litterman portfolio optimization using a bio-inspired approach and neuronets*, Appl. Soft Comput. **112** (2021), 107767.
[19] V. N. Katsikis, S. D. Mourtas, P. S. Stanimirovic, S. Li, X. Cao, *Time-varying mean-variance portfolio selection under transaction costs and cardinality constraint problem via Beetle Antennae Search Algorithm (BAS)*, Oper. Res. Forum (2021).
[20] V. N. Katsikis, S. D. Mourtas, P. S.Stanimirovic, S. L. Xinwei, *Time-varying minimum-cost portfolio insurance under transaction costs problem via Beetle Antennae Search Algorithm (BAS)*, Appl. Math. Comput. **385** (2020), 125453.
[21] V. N. Katsikis, S. D. Mourtas, *Optimal portfolio insurance under nonlinear transaction costs*, j. model. optim. **12** (2020), 117–124.
[22] S. D. Mourtas, V. N. Katsikis, *V-shaped BAS: applications on large portfolios selection problem*, Comput. Econ. (2021).
[23] V. N. Katsikis, S. D. Mourtas, *Portfolio insurance and intelligent algorithms*, Comput. Manag. (2021).
[24] V. N. Katsikis, S. D. Mourtas, *Binary beetle antennae search algorithm for tangency portfolio diversification*, j. model. optim. **13** (2021), 44–50.
[25] M. A. Medvedeva, V. N. Katsikis, S. D. Mourtas, T. E. Simos, *Randomized time-varying knapsack problems via binary beetle antennae search algorithm: Emphasis on applications in portfolio insurance*, Math. Methods Appl. Sci. (2021).
[26] V. N. Katsikis, S. D. Mourtas, *ORPIT: A Matlab toolbox for option replication and portfolio insurance in incomplete markets*, Comput. Econ. **56** (2020), 711–721.
[27] V. N. Katsikis, S. D. Mourtas, *A heuristic process on the existence of positive bases with applications to minimum-cost port-folio insurance in* $C[a, b]$, Appl. Math. Comput. **349** (2019), 221–244.
[28] Y. S. Xia, B. Liu, S. Y. Wang, K. K. Lai, *A model for portfolio selection with order of expected returns*, Comput. Oper. Res. **27** (2000), 409–422.
[29] T. J. Chang, N. Meade, J. E. Beasley, Y. M. Sharaiha, *Heuristics for cardinality constrained portfolio optimization*, Comput. Oper. Res. **27** (2000), 1271–1302.
[30] M. G. Speranza, *A heuristic algorithm for a portfolio optimization model applied to the Milan stock market*, Comput. Oper. Res. **23** (1996), 433–441.
[31] N. J. Jobst, M. D. Horniman, C. A. Lucas, G. Mitra, *Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints*, Quant. Finance **1** (2001), 1–13.
[32] A. Fernandez, S. Gomez, *Portfolio selection using neural networks*, Comput. Oper. Res. **34** (2007), 1177–1191.
[33] Y. Crama, M. Schyns, *Simulated annealing for complex portfolio selection problems*, Eur. J. Oper. Res. **150** (2003), 546–571.
[34] J. Kennedy, R. C. Eberhart, *Particle swarm optimization*, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[35] R. C. Eberhart, J. Kennedy, *A new optimizer using particle swarm theory*, in: Proceedings of the Sixth International Symposium on Micro Machine and Human, 1995, pp. 39–43.
[36] H. M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, Wiley, New York, 1959.
[37] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, *Nonlinear Programming- Theory and Algorithms*, (2rd edition), Wiley, New York, 1993.
[38] M. P. Kennedy, L. O. Chua, *Neural networks for nonlinear programming*, IEEE Trans. Circuits Syst. **35** (1988), 554–562.
[39] A. Hosseini, J. Wang, S. Mohamma, *A recurrent neural network for solving a class of generalized convex optimization problems*, Neural Netw. **44** (2013), 78–86.
[40] A. Nazemi, *A dynamic system model for solving convex nonlinear optimization problems*, Commun. Nonlinear Sci. Numer. Simul. **17** (2011), 1696–1705.
[41] A. Nazemi, *A dynamical model for solving degenerate quadratic minimax problems with constraints*, J. Comput. Appl. Math. **236** (2012), 1282-1295.
[42] A. Nazemi, *A neural network model for solving convex quadratic programming problems with some applications*, Eng. Appl. Artif. Intell. **32** (2014), 54—62.
[43] A. Nikseresht, A. Nazemi, *A novel neural network for solving semidefinite programming problems with some applications*, J. Comput. Appl. Math. **350** (2019), 309–323.
[44] A. Nikseresht, A. Nazemi, *A novel neural network model for solving a class of nonlinear semidefinite programming problems*, J. Comput. Appl. Math. **338** (2018), 69–79.
[45] D. Karbasi, A. Nazemi, M. Rabiei, *An optimization technique for solving a class of ridge fuzzy regression problems*, Neural Process. Lett. **53** (2021), 3307–3338.
[46] A. Feizi, A. Nazemi, *Solving the stochastic support vector regression with probabilistic constraints by a high-performance neural network model*, Eng. Comput. **38** (2022), 1005–1020.
[47] D. Karbasi, A. Nazemi, M. Rabiei, *A parametric recurrent neural network scheme for solving a class of fuzzy regression models with some*

real-world applications, Soft Comput. **24** (2020), 11159–11187.
[48] A. Nazemi, A. Sabeghi, *A novel gradient-based neural network for solving convex second-order cone constrained variational inequality problems*, J. Comput. Appl. Math. **347** (2019), 343–356.
[49] A. Nazemi, M. Mortezaee, *A new gradient-based neural dynamic framework for solving constrained min-max optimization problems with an application in portfolio selection models*, Appl. Intell. **49** (2019), 396–419.
[50] A. Nazemi, *A new collaborate neuro-dynamic framework for solving convex second order cone programming problems with an application in multi-fingered robotic hands*, Appl. Intell. **49** (2019), 3512–3523.
[51] Z. Arjmandzadeh, A. Nazemi, M. Safi, *Solving multiobjective random interval programming problems by a capable neural network framework*, Appl. Intell. **49** (2019), 1566–1579.
[52] A. Nazemi, *A dynamic system model for solving convex nonlinear optimization problems*, Commun. Nonlinear Sci. Numer. Simul. **17** (2013), 1696–1705.
[53] A. Nazemi, F. Omidi, *A capable neural network model for solving the maximum flow problem*, J. Comput. Appl. Math. **236** (2012), 3498–3513.
[54] A. Nazemi, *A capable neural network framework for solving degenerate quadratic optimization problems with an application in image fusion*, Neural Process. Lett. **47** (2018), 167–192.
[55] S. Giove, S. Funari, C. Nardelli, *An interval portfolio selection problems based on regret function*, Eur. J. Oper. Res. **170** (2006), 253–264.
[56] W. G. Zhang, Z. K. Nie, *On admissible efficient portfolio selection problem*, Appl. Math. Comput. **159** (2004), 357–371.
[57] W. G. Zhang, W. A. Liu, Y. L. Wang, *On admissible efficient portfolio selection problem: Models and algorithms*, Appl. Math. Comput. **176** (2006), 208–218.
[58] Y. Leung, K. Z. Chen, Y. C. Jiao, X. B. Gao, B. Xing, K. S. Leung, *A new gradient-based neural network for solving linear and quadratic programming problems*, IEEE trans. neural netw. **12** (2001), 1074–1083,
[59] W. Chen, W.-G. Zhang, *The admissible portfolio selection problem with transaction costs and an improved PSO algorithm*, Phys. A: Stat. Mech. Appl. **389** (2010), 2070–2076.
[60] S. Effati, M. Ranjbar, *A novel recurrent nonlinear neural network for solving quadratic programming problems*, Appl. Math. Model. **35** (2011), 1688–1695.
[61] Q. Tao, J. Cao, D. Sun, *A simple and high performance neural network for quadratic programming problems*, Appl. Math. Comput. **124** (2001), 251–260.
[62] A. Nazemi, M. Nazemi, *A gradient-based neural network method for solving strictly convex quadratic programming problems*, Cognit. Comput. **6** (2014), 484–495.
[63] R. K. Miller and A. N. Michel, *Ordinary Differential Equations*, Academic Press, NewYork, 1982.
[64] X. Huang, *Portfolio Analysis: From Probabilistic to Credibilistic and Uncertain Approaches*, Studies in Fuzziness and Soft Computing, Springer, Berlin, 2010.